CIRM Summer School 2022: EPIT 22 Machine Learning, CS 000

Lecture Notes

Instructors: Prof.Vandergheysnt

Prof. Bach, Prof. Yvon, Prof. Agrawal, Prof. Cummings

Attendees

Editors: Aurélien Garivier, Claire Vernade

May 2022

Contents

1 Optimization

Note takers: Achddou R., Al Marjani, Brogat-Motte, Foucault, Graziani, Le Corre, Pierrot, Sentenac, Yang

Instructor: Francis Bach (Inria)

1.1 Convex Optimization

1.1.1 Setup and notation

Given a dataset: $(x_i, y_i)_{1\le i\le n}$ and a predictor function $f_\theta : \mathcal{X} \longrightarrow \mathbb{R}$, our goal is to minimize

$$F(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f_\theta(x_i)) + \Omega(\theta), \quad \text{where}$$

- The loss function ℓ is convex in its second argument (typically quadratic or logistic).
- $f_\theta : \mathcal{X} \longrightarrow \mathbb{R}$ can be linear $\theta^T\varphi(x)$, where $\theta, \varphi(x) \in \mathbb{R}^d$ or non-linear (neural network).
- The regularizer $\Omega(\theta)$ is typically the squared L_2 or L_1 norm.

GOAL:

1. Minimize the training error $F(\theta)$, where $(x_i, y_i)_{1\le i\le n}$ are i.i.d from an unknown distribution $\mathbb{P}$.
2. Control the testing error $\mathbb{E}_{(X,Y)\sim\mathbb{P}}\left[l(Y, f_{\widehat{\theta}}(X))\right]$.
3. Do this efficiently, i.e. in $o(n)$ time.

Throughout this lecture, we restrict our attention to the case where

$$\ell(y, f_\theta(x)) = \frac{1}{2}|y - \theta^\top\varphi(x)|^2.$$

This is not a limitation, as the same techniques can be used to analyze the general case of convex optimization under smoothness assumptions.

1.1.2 Gradient Descent

We can rewrite $F(\theta) = \frac{1}{2n}\|Y - \Phi\theta\|_2^2$, where $Y \in \mathbb{R}^n, \theta \in \mathbb{R}^d$ and $\Phi \in \mathbb{R}^{n\times d}$. The gradient of F is given by

$$F'(\theta) = \frac{1}{n}\Phi^\top(\Phi\theta - Y) \in \mathbb{R}^d.$$

Let's look for a critical point (which is also a global minimum since the loss is strongly convex):

$$\begin{aligned} F'(\theta^\star) = 0 &\iff \Phi^\top\Phi\theta^\star = \Phi^\top Y, \\ &\iff \theta^\star = (\Phi^\top\Phi)^{-1}\Phi^\top Y, \end{aligned} \tag{1.1}$$

where we assumed that $\Phi^\top\Phi$ is invertible.

Gradient descent: Define $H := \frac{1}{n}\Phi^\top\Phi$. Then GD starts at $\theta_0 = 0$ and performs the update

$$\begin{aligned} \theta_t &= \theta_{t-1} - \gamma F'(\theta_{t-1}) \\ &= \theta_{t-1} - \gamma\left(\frac{\Phi^\top\Phi\theta_{t-1}}{n} - \frac{\Phi^\top Y}{n}\right) \end{aligned} \tag{1.2}$$

Combining equations (1.1) and (1.2) we get

$$\theta_t - \theta^\star = (I_d - \gamma H)^\top(\theta_{t-1} - \theta^\star),$$

Now we study the convergence speed of GD.
Criterion: Using the above, we have

$$\begin{aligned} F(\theta_t) - F(\theta^\star) &= \frac{1}{2}(\theta_t - \theta^\star)^\top H(\theta_t - \theta^\star) \\ &= \frac{1}{2}(\theta_0 - \theta^\star)^\top(I_d - \gamma H)^{2t}H(\theta_0 - \theta^\star) \end{aligned}$$

Wlog (it suffices to change the basis in the ambiant space by a rotation), we may assume that H is diagonal. We let $\lambda_1, \dots, \lambda_d$ be the eigenvalues of H, $\mu := \min_i \lambda_i$ and $L := \max_i \lambda_i$.

Theorem 1.1

Assume that $\mu > 0$ (and bounded away from zero) and that $\gamma L \leq 1$. Then we have a linear convergence (i.e exponential decay of the error) of Gradient Descent :

$$F(\theta_t) - F(\theta^\star) \leq (1 - \gamma\mu)^{2t}[F(\theta_0) - F(\theta^\star)].$$

Clearly, the optimal stepsize is $\gamma^\star = \frac{1}{L}$, which gives a convergence speed of:

$$F(\theta_t) - F(\theta^\star) \leq (1 - \frac{\mu}{L})^{2t}[F(\theta_0) - F(\theta^\star)].$$

We denote by $\kappa = \frac{L}{\mu}$ the condition number, which controls the convergence speed of GD.

Question: What if μ is very small ? Then one can write

$$H(1-2\gamma H)^{2t} = \mathrm{Diag}\left((\lambda_i(1-\gamma\lambda_i)^{2t})_{1\le i\le d} \right).$$

Lemma 1.1. *$\lambda(1-\gamma\lambda)^{2t} \le \lambda\exp(-\gamma\lambda 2t) \le \frac{1}{4t\gamma}$, as soon as $\gamma\lambda 2t \le 1$.*

Theorem 1.2

Assume that $2\gamma L \le 1$. Then we have for all $t \ge 1$, $F(\theta_t) - F(\theta^\star) \le \frac{\|\theta_t-\theta^\star\|_2^2}{4t\gamma}$.

Remark.
1. *If F is convex, smooth and with bounded Hessians then a similar result holds.*
2. *For non-constant stepsize γ, there are also guarantees, provided that "we do not let γ be too small or too large".*
3. *GD is adaptive to the curvature of the function.*
4. *Problem 1: The complexity is not $o(n)$.*
5. *Problem 2: The convergence speed is not minimax optimal.*

1.1.3 Acceleration

Let's look at what happens when we add (Heavy ball) momentum

$$\theta_t = \theta_{t-1} - \gamma F'(\theta_{t-1}) + \delta(\theta_{t-1} - \theta_{t-2})$$

Denote $\eta_t = \theta_t - \theta^\star$. Then we have

$$\eta_t = \eta_{t-1} - \gamma H\eta_{t-1} + \delta(\eta_{t-1} - \eta_{t-2}),$$

aka a second order recursion. Based on our previous life as undergrad students, we solve the characteristic equation $r^2 = (1-\gamma\lambda)r + \delta(r-1)$. We need to make sure that the solutions are complex conjugates $\rho\exp(i\varphi)$ and $\rho\exp(-i\varphi)$. In other words, we need to ensure that

$$\begin{aligned}\Delta &= [(1-\gamma\lambda)+\delta]^2 - 4\delta \\ &= \delta^2 - 2(1+\gamma\lambda)\delta + \gamma^2\lambda^2 + 1 \le 0.\end{aligned}$$

The roots of Δ are $\delta_{1,2} = 1 \pm \sqrt{\gamma\lambda}$. Therefore, to guarantee convergence of the sequence $(\eta_t)_{t\ge 1}$, we need to set $\delta \in [1-\sqrt{\gamma\lambda}, 1+\sqrt{\gamma\lambda}]$. We obtain convergence in $\frac{1}{t^2}$.

Convex function Nesterov Acceleration

$$\begin{aligned}\theta_t &= \eta_{t-1} - \gamma F'(\eta_{t-1}) \\ \eta_t &= \theta_t + \delta(\theta_{t-1} - \theta_t)\end{aligned}$$

Remark. *Checking homogeneity in the formulas is a quick way to make sure that the result is not awfully wrong.*

1.1.4 Stochastic Gradient Descent

We want to minimize

$$\mathbf{E}_{p(x,y)}\left[l(y, f_\theta(x))\right]$$

At each iteration, we compute

$$\theta_k = \theta_{k-1} - \gamma_k \frac{\partial l}{\partial \theta}(y_k, f_\theta(x_k)) \quad with \quad x_k, y_k \sim p(x,y)\,.$$

There are two settings:

- single pass with n iid pairs and $F(\theta)$ being the test error
- Multiple passes on finite data set with p, the empirical distance and where $F(\theta)$ is the training error

1.1.5 SGD: Least squares example

We will consider the same example of least squares in which $l(y, f_\theta(x)) = \frac{1}{2}\|y - \theta^\top \varphi(x)\|^2$, thus

$$\frac{\partial l}{\partial \theta} = \varphi(x)(\varphi(x)^\top \theta - y)$$

Assumption $y = \varphi^\top \theta_\star + \varepsilon$ with $\mathbf{E}[\varepsilon\varphi(x)] = 0$ and $\varepsilon^2 \leq \sigma^2$ a.s. and $\gamma H \leq I$

The iteration becomes with Least Squares:

$$\theta_n = \theta_{n-1} - \gamma_n \varphi(x_n)(\varphi^\top \theta - y)$$

By using $\eta_n = (\theta_n - \theta_\star)$ which is the same as GD:

$$\eta_n = \eta_{n-1} - \gamma_n \underbrace{\varphi(x_n)\varphi(x_n)^T}_{\mathbf{E}[\ldots]=H} \eta_{n-1} + \gamma_n \underbrace{\varepsilon_n \varphi(x_n)}_{\mathbf{E}[\ldots]=0}$$

Study:

$$\eta_n = \eta_{n-1} - \gamma H \eta_{n-1} + \gamma \varepsilon_n \varphi(x_n)$$

$$\eta_n = (1-\gamma H)\eta_{n-1} + \gamma \underbrace{\varepsilon_n \varphi(x_n)}_{\mathbb{E}[\varepsilon_n \varphi(x_n)(\varepsilon_n \varphi(x_n))^T)] \leq \sigma^2 H}$$

Lemma 1.2.

$$\eta_n = \underbrace{(1-\gamma H)^n \eta_0}_{determinstic} + \underbrace{\gamma \sum_{k=1}^{n} (1-\gamma H)^{n-k} \varepsilon_k \varphi(x_k)}_{zero\ mean}$$

With this lemma, we can decompose $\mathbb{E}[\eta_n \eta_n^T]$ in two parts

$$\mathbb{E}[\eta_n \eta_n^T] = \underbrace{(1-\gamma H)^n \eta_0 \eta_0^T (1-\gamma H)^n}_{\text{Same as GD}} + \underbrace{\sum_{k=1}^{n} \gamma^2 (1-\gamma H)^{n-k} \mathbb{E}[\varepsilon_k^2 \varphi(x_k)\varphi(x_k)^T](1-\gamma H)^{n-k}}_{\text{SGD extra term}}$$

Our goal is to consider,

$$\mathbb{E}[F(\theta_n) - F(\theta_\star)] = \frac{1}{2}\mathbb{E}[\eta_n^T H \eta_n] = \frac{1}{2} tr(H \mathbb{E}[\eta_n \eta_n^T])$$

By replacing $\mathbb{E}[\eta_n \eta_n^T]$, we have **an extra term compared to SGD**:

$$\begin{aligned} tr(H \sum_{k=1}^{n-1} \gamma^2 \sigma^2 H (1-\gamma H)^{2k}) &\leq \sigma^2 \gamma^2 tr(H^2 (I - (I-\gamma H)^2)^{-1}) \\ &\leq \sigma^2 \gamma^2 tr(H^2 (\gamma H)^{-1}) \\ &\leq \sigma^2 \gamma tr(H) \end{aligned}$$

Summary:

$$\mathbb{E}[F(\theta_n) - F(\theta_\star)] \leq \frac{1}{n\gamma_n} \|\eta_0\|_2^2 + \gamma_n \sigma^2 tr(H)$$

Question: How to get a convergent algo?

- γ decreasing: Brute force way or lazy way with a constant step size depending on the horizon.
- Averaging

Sweet spot: $\gamma_n = \frac{1}{\sqrt{n}}$ is a nice idea, to get $\mathbf{E}[F(\theta_n) - F(\theta_\star)] \leq \frac{1}{\sqrt{n}}(...)$ But it is not homogeneous.

1.2 SGD with averaging (beginning of lecture 2)

Recall: $l_i(\theta) = (y_i - \varphi(x_i)^T\theta)^2$

Algorithm. The algorithm is the same as the classical SGD, but the final estimate is the average of the previous θ_i.

- initialize θ_0
- $\theta_n = \theta_{n-1} - \gamma \frac{\partial l_n}{\partial \theta}(\theta_{n-1})$.
- $\bar{\theta}_n = \frac{1}{n+1}\sum_{i=0}^n \theta_i$

Assumption.

$$y = \theta_*^T\varphi(x) + \varepsilon, \|\varphi(x)\|_2 \leq R, |\varepsilon| \leq \sigma$$

Let us prove the convergence of this algorithm.

1.2.1 Convergence proof for Least-squares

We note $\bar{\eta}_n = \bar{\theta}_n - \theta^* = \frac{1}{n+1}\sum_{k=0}^n \eta_k$ with $\eta_n = \theta_n - \theta^*$.

As previously:

$$\eta_n = (1-\gamma H)^n\eta_0 + \gamma\sum_{k=1}^n (1-\gamma H)^{n-k}\varepsilon_k\varphi(x_k)$$

so

$$\bar{\eta}_n = \underbrace{\frac{1}{n+1}\sum_{k=0}^n (1-\gamma H)^k\eta_0}_{\text{A: deterministic part}} + \underbrace{\frac{\gamma}{n+1}\sum_{k=1}^n\sum_{j=1}^k (1-\gamma H)^{n-k}\varepsilon_k\varphi(x_k)}_{\text{B: noise part}}.$$

The ε_k being independent with $\mathbb{E}[\varepsilon] = 0$, we have

$$\mathbb{E}[F(\bar{\theta}_n) - F(\theta_*)] = \frac{1}{2}\mathbb{E}[\bar{\eta}_n^T H\bar{\eta}_n] = \frac{1}{2}\left(\mathbb{E}[A^T HA] + \mathbb{E}[B^T HB]\right)$$

Here, we study the deterministic and noisy term separately.

Deterministic part.

We have

$$A = \frac{1}{n+1}\sum_{k=0}^{n}(I-\gamma H)^k\eta_0 = \frac{1}{n+1}\frac{I-(I-\gamma H)^{n+1}}{I-(I-\gamma H)}\eta_0$$
$$= \frac{(\gamma H)^{-1}}{n+1}(I-(I-\gamma H)^{n+1})\eta_0. \tag{1.3}$$

So using $0 \preceq (I-\gamma H)^{n+1} \preceq I$ from $\gamma H \preceq I$, we get

$$\mathbb{E}[A^T HA] \leq \frac{\eta_0^T H^{-1}\eta_0}{\gamma^2 n^2}. \tag{1.4}$$

Noise part.

We have

$$B = \frac{\gamma}{n+1}\sum_{k=1}^{n}\sum_{j=1}^{k}(1-\gamma H)^{k-j}\varepsilon_j\varphi(x_j)$$
$$= \frac{\gamma}{n+1}\sum_{j=1}^{n}\Big(\sum_{k=j}^{n}(1-\gamma H)^{k-j}\Big)\varepsilon_j\varphi(x_j)$$
$$\approx \frac{\gamma}{n+1}\sum_{j=1}^{n}(\gamma H)^{-1}\varepsilon_j\varphi(x_j) \tag{1.5}$$

So, we get

$$\mathbb{E}[B^T HB] \leq \frac{1}{n^2}\sum_{j=1}^{n}\mathbb{E}[\varphi(x_j)^T\varepsilon_j^T H^{-1}\varepsilon_j\varphi(x_j)]$$
$$\leq \frac{\sigma^2}{n^2}\sum_{j=1}^{n}Tr(H^{-1}\mathbb{E}[\varphi(x_j)\varphi(x_j)^T])$$
$$\leq \frac{\sigma^2 d}{n} \tag{1.6}$$

Summing the two parts.

From equations 1.4 and 1.6 we obtain :

$$\mathbb{E}[F(\bar{\theta}_n) - F(\theta_*)] \leq \frac{\eta_0^T H^{-1}\eta_0}{\gamma^2 n^2} + \frac{\sigma^2 d}{n} \tag{1.7}$$

1.2.2 General case SGD: averaging

In this section we prove the convergence in a more global setting than least squares.

We have :

$$\|\theta_n - \theta_*\|^2 = \|\theta_{n-1} - \theta_*\|^2 - 2\gamma(\theta_{n-1} - \theta_*)^T \frac{\partial l_n}{\partial \theta} + \gamma^2 \|\frac{\partial l_n}{\partial \theta}\|^2$$

with

$$\frac{\partial l_n}{\partial \theta} = \varphi(x_n)\varphi(x_n)^T(\theta_{n-1} - \theta_*) + \varepsilon_n \varphi(x_n)$$

and

$$\|\frac{\partial l_n}{\partial \theta}\|^2 \leq 2\left(\|\varphi(x_n)\varphi(x_n)^T(\theta_{n-1} - \theta_*)\|^2 + \|\varepsilon_n \varphi(x_n)\|^2\right).$$

So

$$\begin{aligned}\mathbb{E}(\|\theta_n - \theta_*\|^2|\mathcal{F}_{n-1}) &\leq \|\theta_{n-1} - \theta_*\|^2 - 2\gamma(\theta_{n-1} - \theta_*)^T H(\theta_{n-1} - \theta_*) + 2\gamma^2\sigma^2 R^2 && (1.8)\\ &+ 2\gamma^2(\theta_{n-1} - \theta_*)^T R^2 H(\theta_{n-1} - \theta_*) \\ &\leq \|\theta_{n-1} - \theta_*\|^2 + 2\gamma^2\sigma^2R^2 - 2\gamma(1-\gamma R^2)((\theta_{n-1} - \theta_*)^T H(\theta_{n-1} - \theta_*)) && (1.9)\end{aligned}$$

Moreover,

$$-2\gamma(1-\gamma R^2) \leq -\gamma$$

(from $\gamma R^2 \leq 1/2$) and

$$((\theta_{n-1} - \theta_*)^T H(\theta_{n-1} - \theta_*)) = 2(F(\theta_{n-1}) - F(\theta_*))$$

so

$$\mathbb{E}(F(\theta_{n-1}) - F(\theta_*)) \leq \frac{1}{2\gamma}(\mathbb{E}(\|\theta_{n-1} - \theta_*\|^2) - \mathbb{E}(\|\theta_n - \theta_*\|^2)) + \gamma\sigma^2R^2$$

then from Jensen inequality :

$$\mathbb{E}(F(\bar{\theta}_{n-1}) - F(\theta_*)) \leq 1/N \sum_{n=1}^{N} \mathbb{E}(F(\theta_{n-1}) - F(\theta_*)) \leq \frac{\|\theta_0 - \theta_*\|^2}{2\gamma N} + \gamma\sigma^2 R \tag{1.10}$$

Shown in Bach and Moulines [2013] : $\gamma\sigma^2 R$ can be replaced by $\frac{\sigma^2 d}{n}$.

1.2.3 Dual coordinate ascent

Ref Shalev-Shwartz and Zhang [2012] **Finite sum set up.**

$F(\theta) = \frac{1}{2n}\|y - \Phi\theta\|_2^2 + \lambda/2\|\theta\|_2^2$, $y \in R^n, \Phi \in R^{n\times d}$.

Closed-form solution derivation.

The problem is stated as follows :

$$\min_{u=\Phi\theta\in R^n, \theta\in R^d} \frac{1}{2n}\|y-u\|_2^2 + \lambda/2\|\theta\|_2^2,$$

and can be re-written as :

$$\min_{u,\theta}\max_{\alpha\in R^n} \frac{1}{2n}\|y-u\|_2^2 + \lambda/2\|\theta\|_2^2 + \lambda\alpha^T(u-\Phi\theta).$$

Assuming that we have $\theta^* = \Phi^T\alpha$, we have :

$$\max_{\alpha\in R^n} -\lambda/2\alpha^T\Phi\Phi^T\alpha + \min_u \frac{1}{2n}(\|y\|^2 + \|u\|^2 - 2y^Tu) + \lambda\alpha^Tu \tag{1.11}$$

We begin by minimizing the term depending in u :

$$\begin{aligned}\min_u \frac{1}{2n}(\|y\|^2 + \|u\|^2 - 2y^Tu) + \lambda\alpha^Tu &= \min_u \frac{1}{2n}\|y\|^2 + \frac{1}{2n}\|u\|^2 - u^T(y/n - \lambda\alpha)\\ &= -\frac{n}{2}\|y/n - \lambda\alpha\|_2^2 + \frac{1}{2n}\|y\|^2\\ &= -n\lambda^2/2\|\alpha\|_2^2 + \lambda\alpha^Ty \end{aligned}\tag{1.12}$$

as $u^* = y - n\lambda\alpha$. Replacing this term in equation 1.11, we get :

$$\max_{\alpha\in R^n} -\frac{\lambda}{2}\alpha^T\Phi\Phi^T\alpha - \frac{n\lambda^2}{2}\|\alpha\|_2^2 + \lambda\alpha^Ty = \min_\alpha G(\alpha)\lambda \tag{1.13}$$

where we define G as $G(\alpha) = -\frac{1}{2}\alpha^T\Phi\Phi^T\alpha - \frac{n\lambda}{2}\|\alpha\|_2^2 + \alpha^Ty$.

From equation 1.13 we get that :

$$\alpha_* = (\Phi\Phi^T + n\lambda I)^{-1}y$$

$\theta = \Phi^T\alpha = \sum_{i=1}^n \varphi(x_i)\alpha_i$

$\nabla^2 G(\alpha) = \Phi\Phi^T$

$\mathrm{Diag}\nabla^2 G(\alpha) \leq R^2$

Coordinate ascent algorithm.

- Choose coordinate at random in $1, \dots, n$
- Optimize with respect to α_i

Lemma 1.3:

if $h(\beta)$ quadractic then $\inf_\beta h = h(\beta_0) - \frac{1}{2}\frac{h'(\beta_0)^2}{h''(\beta_0)}$.

What is the convergence rate?

$\mathbb{E}(G(\alpha^t) - G(\alpha_*)) = \frac{1}{n}\sum_{i=1}^n G(\alpha) - \frac{1}{2R^2}\nabla G(\alpha)_i^2 - G(\alpha_*) = G(\alpha) - \frac{1}{2nR^2}\|\nabla G(\alpha)\|^2 - G(\alpha_*)$

Losajevich condition.

If G is λ-strongly convex $G(\alpha) - G(\alpha_*) \leq \frac{1}{2\lambda}\|\nabla G(\alpha)\|$, then

$\mathbb{E}(G(\alpha^t) - G(\alpha_*)) \leq (G(\alpha) - G(\alpha_*))(1 - \frac{\lambda}{R^2+n\lambda})$

to reach ε precision

$t \approx \frac{R^2+n\lambda}{\lambda}\log 1/\varepsilon$.

1.3 Global Optimization

1.3.1 Gradient Descent for a single hidden layer: Intro

The predictor is

$$h(x) := \frac{1}{m}\theta_2^T\sigma(\theta_1^T x) = \frac{1}{m}\sum_{j=1}^m \theta_2(j)\sigma\left(\theta_1(.,j)^T x\right).$$

It is rewritten

$$h(x) = \frac{1}{m}\sum_{j=1}^m \Psi(w_j) \text{ with } \Psi(w_j)(x) := \theta_2(j)\sigma\left(\theta_1(.,j)^T x\right).$$

Goal: Minimize

$$R(h) := \mathbb{E}_{p(x,y)}\ell(y, h(x)) \text{ with } R \text{ convex.}$$

Main insight:

$$h = \int_{\mathcal{W}} \Psi(w)d\mu(w) \text{ with } d\mu(w) = \frac{1}{m}\sum_{j=1}^m \delta_{w_j}.$$

Overparametrized regime $\implies$ mean fields limit, measure μ with densities.

We want to minimize with respect to measure μ:

$$R\left(\int_{\mathcal{W}} \Psi(w)d\mu(w)\right).$$

μ is represented by a finite set of m particules, gradient descent on $(w_1, \ldots, w_m)$.

Three main questions:

- Algorithm limit when m gets large
- Global conv. to a global minimum
- Prediction performance.

1.3.2 Derivations

Goal Find

$$\min_{\mu} F(\mu) = R\left(\int_{\mathcal{W}} \Psi(w) d\mu(w)\right).$$

We study Gradient flows instead of GD:

$$\dot{W} = -m\nabla F(W) \text{ with } W = (w_1, \ldots, w_m).$$

It's an idealisation of this GD with small steps:

$$W_{k+1} = W_k - \gamma \nabla F(W_k).$$

Reindexing:

$$\overline{W}_{k\gamma+\gamma} = \overline{W}_{\gamma k} - \gamma \nabla F(\overline{W}_{\gamma k}).$$

Assimilating $k\gamma \leftarrow t$, $\gamma \leftarrow dt$ gives:

$$\overline{W}_{t+dt} = \overline{W}_t - \gamma \nabla F(\overline{W}_t).$$

This is an Euler scheme of the gradient flow differential equation.

What's the bound on the deviation between the gradient flow and the GD ? (This problem is not treated, we do computations at the limit in the course).

Note: SGD is not a Langevin diffusion in the limit as the noise is multiplied by γ. It would be if the noise term were multiplied by $\gamma^{1/2}$.

First derivation on linear Networks (no activation function).

Loss function $R : \mathcal{R}^{d\times d} \longrightarrow \mathcal{R}$.

The particles' parameters $w_1, \ldots, w_m$ lie in $\mathcal{R}^d$, $W = (w_1, \ldots, w_m) \in \mathcal{R}^{d\times m}$.

The function to be minimized

$$F(W) = R\left(\frac{1}{m}\sum_{j=1}^{m} w_j w_j^T\right) = R\left(\frac{1}{m} W W^T\right)$$

is the composition of a convex function with a quadratic function of the parameters.

Proposition 1. *The function $M := \frac{1}{m}WW^T$ is positive semi-definite.*

The goal is to minimize a convex function over the set of SDP matrices.

Lemma 1.3. *M is optimal iff:*

$$\begin{cases} M \succcurlyeq 0 \\ \nabla R(M) \succcurlyeq 0 \\ tr(M\nabla R(M)) = 0. \end{cases}$$

Proof. **1. The condition is necessary**. The first condition is true by construction. Also, if M is optimal then:

$$\begin{aligned} \forall M + \varepsilon\Delta \succcurlyeq 0,\ R(M+\varepsilon\Delta) \geq R(M), & \\ \implies & R(M) + \varepsilon \mathrm{tr}\Delta\nabla R(M) + o(\varepsilon) \geq R(M) \\ \implies & \mathrm{tr}\Delta\nabla R(M) \geq 0. \end{aligned}$$

For any $u \in \mathbb{R}^d$, pick $\Delta = uu^T$, this implies $u^T\nabla R(M)u \geq 0$, which implies $\nabla R(M) \succcurlyeq 0$.

Pick $\Delta = \pm M$, $\implies$ $tr(M)\nabla R(M) = 0$. Note that the convexity of R is not used to proof the necessary condition.
2. The condition is sufficient. For any matrix $N \succcurlyeq 0$:

$$\begin{aligned} R(N) &\geq R(M) + \mathrm{tr}\nabla R(M)(N-M),\ \text{(by convexity of R)} \\ &\geq R(M) + \underbrace{\mathrm{tr}\nabla R(M)N}_{\geq 0} - \underbrace{\mathrm{tr}\nabla R(M)M}_{=0} \\ &\geq R(M). \end{aligned}$$

□

Formulation of the problem We study the gradient flow of

$$F(W) = R\left(\frac{1}{m}WW^T\right).$$

It respects equation:

$$\dot{W} = -\frac{m}{2}\nabla F(w).$$

The speed of the gradient flow is tuned for elegance of computations. Let's compute the gradient explicitly.

$$\begin{aligned} F(W+\Delta) &= R\left(\frac{1}{m}W\Delta^T + \frac{1}{m}\Delta W^T + O(\|\Delta\|^2) + \frac{1}{m}WW^T\right) \\ &= R(\frac{1}{m}WW^T) + \mathrm{tr}(\nabla R(M)(\frac{1}{m}(W\Delta^T + \Delta W^T)))) + +O(\|\Delta\|^2) \\ &= R(\frac{1}{m}WW^T) + \mathrm{tr}(\Delta^T\frac{2}{m}\nabla R(M)W) + O(\|\Delta\|^2). \end{aligned}$$

Identifying the gradient we get:

$$\dot{W} = -\frac{m}{2}\nabla F(w) = -\nabla R\left(\frac{1}{m}WW^T\right)W.$$

Goal: Running gradient flow ends on a point satisfying the conditions of optimality.

Fact 1: If W has rank d at time 0 then $W(t)$ remains full rank.

Proof. We study $r := \log\det(M)$.

Let's write the ODE for $M = \frac{1}{m}WW^T$.

$$\begin{aligned}\dot{M} &= \frac{1}{m}\dot{W}W^T + \frac{1}{m}W\dot{W}^T \\ &= \frac{1}{m}\left[-\nabla R(M)WW^T - WW^T\nabla R(M)\right] \\ &= -\nabla R(M)M - M\nabla R(M).\end{aligned}$$

Note that it depends only on M. Now for the ODE of r:

$$\begin{aligned}\dot{r} &= \operatorname{tr}(M^{-1}\dot{M}) \\ &= -\operatorname{tr}(M^{-1}\nabla R(M)M + M\nabla R(M)) \\ &= -2\operatorname{tr}\left(\nabla R(M)\right)\end{aligned}$$

This implies that if $r(0)$ is defined, then r is always defined, thus M remains full rank if it is at $t = 0$. □

Assume that $M(t)$ converges:

$$M(t) \longrightarrow M_\infty.$$

For general 2 layer networks, this has not been shown, in the simpler case we are considering (Linear Networks), it has been done, we skip it for this course.

Proposition 2. *M_∞ is optimal.*

Proof. The stationarity condition gives:

$$\begin{aligned}&\dot{M} = -\nabla R(M)M - M\nabla R(M) \\ \implies& -\nabla R(M_\infty)M_\infty - M_\infty\nabla R(M_\infty) = 0 \\ \implies& \operatorname{tr}(M_\infty\nabla R(M_\infty)) = 0.\end{aligned}$$

The tricky part is to show $\nabla R(M_\infty) \succcurlyeq 0$.

Assume it's not the case, i.e. $\lambda_{\min}\left(\nabla R(M_\infty)\right) < 0$.

Define set

$$K := \{z \text{ s.t. } z^T\nabla R(M_\infty)z < 0\}.$$

If there is at least one negative eigenvalue, K has a non empty interior.

We assumed that $M(t)$ converges to M_∞, there thus exists some t_0 s.t.:

$$\|M(t_0) - M_\infty\| \leq \varepsilon.$$

Consider any $y_0 \in K$. Since $W(t_0)$ has full rank, there exists some $\alpha_0 \in \mathbb{R}^m$ s.t. $y_0 = W(t_0)\alpha_0$. Define

$$z(t) := W(t)\alpha_0.$$

Note that $z(t_0) = y_0$, therefore it belongs to K by construction. This definition implies the following ODE on z:

$$\dot{z}(t) = \dot{W}(t)\alpha_0 = -\nabla R(M) W(t)\alpha_0 = -\nabla R(M) z(t).$$

This ODE implies that $z(t)$ is always well defined if $z(0)$ is. Define the shorthand $A := \nabla R(M_\infty)$.

$$\begin{aligned}
\frac{d}{dt}\left[\frac{z^T A z}{z^T z}\right] &= \frac{\dot{z}^T A z}{z^T z} - \frac{z^T A \dot{z}}{z^T z} - 2\frac{z^T A z \dot{z}^T z}{(z^T z)^2} \\
&= -2\frac{z^T \nabla R(M) A z}{z^T z} + 2\frac{z^T A z z^T \nabla R(M) z}{(z^T z)^2} \\
&=_{t\to\infty} -2\frac{z^T A^2 z}{z^T z} + 2\frac{(z^T A z)^2}{(z^T z)^2} \leq 0.
\end{aligned}$$

The last expression is non positive due to the Cauchy-Schwarz inequality. This implies that if z enters K, it never leaves. We also have:

$$\frac{d}{dt}\|z(t)\|^2 = 2\dot{z}^T z = -2z^T \nabla R(M) z =_{t\to\infty} -2z^T A z.$$

" $\implies$ "$\|z(t)\|^2$ diverges. (The last computations are not rigorous because made in the limit, they can be made "true" involving technicalities and ε.) Divergence is impossible, as $z(t) := W(t)\alpha_0$ and $W(t)$ converges, so the hypothesis $\lambda_{\min} < 0$ is false. □

2 Machine Learning on Graphs

Note takers: Bardou, Brandao, Even, Gonon, Mitarchuk, Natura, Le Q-T, Pic, Shilov, Weber

Instructor: Pierre Vandergheynst (EPFL)

2.1 Hour 1 (Monday 10-11 am)

2.1.1 Introduction

Graphs are natural structures to represent data in many applications, such as biology (e.g. proteins) or networks (e.g. transportation, social, energy). On such a structure, building, learning and processing meaningful features can be done at different scales:

- at node scale: to classify a node, to predict an edge,
- at a (sub)graph scale: to detect and classify structures (e.g. communities)

More generally, we want to be able to process a (sample of) a graph signal, to be able to classify information over a fixed network. These types of tasks require adapted tools and methods. The following sections describe some of them.

2.1.2 Some elements of Spectral Graph Theory

First of all, let us formally define what a graph is and some basic related notions.

Definition 1. *A graph $G = (V, E)$ is a pair, composed of $V = \{v_1, \cdots, v_N\}$ a set of vertices and $E = \{e_1, \cdots, e_M\}$ a set of edges.*

A graph has some basic descriptors, gathering information about its structure. The degree of a vertex v, denoted $d(v)$ or d_v is the number of nodes connected to it.

$$d(v) = |\{u \in V \text{ s.t. } (u, v) \in E \textbf{ or } (v, u) \in E\}|$$

The degrees of each node are gathered in the degree matrix $\mathbf{D}(G) = \mathrm{D}iag(d_1, \cdots, d_N)$.

Definition 2. *(Diameter, volume of a graph) The diameter $d(G)$ of a graph G is the longest shortest path between a pair of vertices. The volume $vol(G)$ of a graph G is the sum of the degree of its vertices, that is $vol(G) = \sum_{v \in V} d_v = tr(\mathbf{D})$.*

The incidence matrix S is a $N \times M$ matrix defined as:

$$\mathrm{S}_{i,j} = \begin{cases} +1 & \text{if } e_j = (v_i, v_k) \text{ for some } k \\ -1 & \text{if } e_j = (v_k, v_i) \text{ for some } k \\ 0 & \text{otherwise} \end{cases}$$

We can represent G with the adjacency matrix $\mathbf{A}$, a $N \times N$ matrix:

$$\mathbf{A}_{i,j} = \begin{cases} +1 & \text{if } (v_i, v_j) \in E \textbf{ or } (v_j, v_i) \in E \\ 0 & \text{otherwise} \end{cases}$$

We can extend the definition of the degree of a node to weighted graphs, for which the adjacency matrix becomes a weight matrix $\mathbf{W}$, with $\mathbf{W}_{i,j} \geq 0$: $d(v_i) = \sum_{j=1}^{N} \mathbf{W}_{i,j}$.

With these definitions, we can eventually define a slightly more complex matrix:

$$\mathrm{S}\mathrm{S}^T = \mathbf{D} - \mathbf{A}$$

Definition 3. *The (unnormalized) Laplacian of G, denoted $\mathbf{L}$ is defined as:*

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

$\mathbf{L}$ does not depend on the orientation of the edges, therefore it can be defined for undirected graph too. For weighted graphs, we have the extension $\mathbf{L} = \mathbf{D} - \mathbf{W}$. By definition, $\mathbf{L}$ is symmetric, but also positive semi-definite.

Proof. For any weight matrix $\mathbf{W}$, we can show through simple calculations that $\forall x \in \mathbb{R}^N$:

$$x^T \mathbf{L} x = \frac{1}{2} \sum_{j \sim i} \mathbf{W}_{i,j} (x[i] - x[j])^2 \geq 0$$

$\square$

Considering a graph signal $f \in \mathbb{R}^N$ (node attribute), we can observe the differential nature of the incidence matrix S by noting that

$$(\mathrm{S}^T f)[j] = f[i] - f[k]$$

which can be considered as the derivative of f along edge j. Here, edge j connects vertex v_i to vertex v_k. Extending this observation to the whole signal f, we can think of $F = \mathrm{S}^T f \in \mathbb{R}^M$ as the gradient of f. Note that F is an *edge-based* signal. Similarly, for an edge-based signal G, $g = \mathrm{S}G \in \mathbb{R}^N$ is the divergence of G. Note that g is a *vertex-based* signal.

This insight can help us understand the nature of the Laplacian $\mathbf{L} = \mathrm{SS}^T$. Considering a graph (vertex-based) signal, we have:

$$\begin{aligned} f^T \mathbf{L} f &= f^T \mathrm{SS}^T f \\ &= ||\mathrm{S}^T f||_2^2 \\ &= \sum_{i \sim k} (f[i] - f[k])^2 \end{aligned}$$

Therefore, for a weighted graph, we can consider the quadratic form of a signal involving the Laplacian $\mathbf{L}$:

$$f^T \mathbf{L} f = \sum_{i \sim k} \mathbf{W}_{i,k} (f[i] - f[k])^2$$

as a measure of how *smooth* the signal is.

Since $\mathbf{L}$ is a real, symmetric and positive semi-definite matrix, it has an eigendecomposition into real eigenvalues and eigenvectors λ_i, u_i, with non-negative eigenvalues ($0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$). Given the properties of the Laplacian $\mathbf{L}$, its eigendecomposition may give us information about the structure of the graph.

Proposition. The number of connected components c of G is the dimension of the nullspace of $\mathbf{L}$. Furthermore, the null space of $\mathbf{L}$ has a basis of indicator vectors of the connected components of G. An indicator of a subset H of V is

$$x \in \mathbb{R}^N \text{ s.t. } \begin{cases} x[i] = 1 & \text{if } \in H \\ x[i] = 0 & \text{otherwise} \end{cases}$$

Definition 4. *(Algebraic connectivity, Fielder Vector) The algebraic connectivity of a graph is the second smallest eigenvalue of* $\mathbf{L}$, λ_2, *which is positive if and only if the graph is connected. Its associated eigenvector,* u_2 *is called the Fielder vector.*

The value of λ_2 gradually increases with the connectedness of the graph. In particular, we can show that $\lambda_2 \geq \frac{1}{vol(G)\mathrm{d}(G)}$.

Definition 5. *The Cheeger Constant of a graph is denoted* $h(G)$ *and is defined as:*

$$h(G) = \min_{A \subset V} \left\{ \frac{|\partial A|}{\min\left(vol(A), vol(\bar{A})\right)} \ s.t.\ 0 < |A| < \frac{1}{2}|V| \right\}$$

with $\bar{A}$ *the complementary of* A, $vol(A) = \sum_{v \in A} d(v)$ *and* $\partial A = \{(u,v) \in E \ s.t.\ u \in A, v \in \bar{A}\}$ *gathering the edges between* A *and* $\bar{A}$.

The Cheeger Constant is able to measure the presence of "bottlenecks" which are formed by strongly connected components loosely connected with each others.

We can relate the Cheeger Constant with algebraic connectivity by the Cheeger inequalities. A simple example is given below:

Theorem 2.1. *For a general graph* G,

$$2h(G) \geq \lambda_2 \geq \frac{h^2(G)}{2}$$

2.2 Hour 2 (Monday 11 am -12 pm)

2.2.1 Cut and Cluster

WIP

We can associate a cost to cutting a graph in subsets $A_1, \ldots, A_n$ inspired by the Cheeger constant, which is the minimal surface to volume ratio among graph cuts. To do so, begin by defining the total weight of the edges between $A, B \subset V$:

$$C(A,B) := \sum_{i \in A, j \in B} \mathbf{W}\left[i,j\right], \tag{2.1}$$

which can be regarded as a proxy for surface between A and B and

$$\mathrm{cut}(A_1, \ldots, A_k) := \frac{1}{2}\sum_{i=1}^{k} C(A_i, \bar{A}_i) \tag{2.2}$$

For two different definitions of subset volume we obtain

$$\mathrm{RatioCut}(A, \bar{A}) = \frac{\mathrm{cut}(A, \bar{A})}{|A|} + \frac{\mathrm{cut}(\bar{A}, A)}{|\bar{A}|} \tag{2.3}$$

$$\text{NormalizedCut}(A, \bar{A}) = \frac{\text{cut}(A, \bar{A})}{\text{vol}(A)} + \frac{\text{cut}(\bar{A}, A)}{\text{vol}(\bar{A})} \tag{2.4}$$

We shall see that there is a relationship between

$$\min_{A \subset V} \text{RatioCut}(A, \bar{A}) \tag{2.5}$$

and the second eigenvalue of the Laplacian, which will allow us to approximate RatioCut. To do so, first define the indicator function

$$f[i] = \begin{cases} \sqrt{|\bar{A}|/|A|} & \text{if} i \in A \\ -\sqrt{|A|/|\bar{A}|} & \text{if} i \in \bar{A} \end{cases} \tag{2.6}$$

Since we have

$$\begin{aligned} f^\top \mathbf{L} f &= f^\top (D) f - f^\top W f && (2.7) \\ &= \sum_i f_i \sum_j w_{ji} f_i - \sum_i f_i \sum_j w_{ij} f_j && (2.8) \\ &= \sum_{i,j} w_{ij} (f_i^2 - f_i f_j) && (2.9) \\ &= \frac{1}{2} \sum_{i,j} w_{ij} (f_i^2 - 2 f_i f_j + f_j^2) && (2.10) \\ &= \frac{1}{2} \sum_{i,j} w_{ij} (f_i - f_j)^2 && (2.11) \\ &= \sum_{i \sim j} \mathbf{W}(i,j)(f[i] - f[j])^2 && (2.12) \end{aligned}$$

replacing this above we can show that (assuming $A, \bar{A} \neq \emptyset$)

$$\begin{aligned}
f^\top \mathbf{L} f &= \frac{1}{2} \sum_{i,j} w_{ij} (f_i - f_j)^2 && (2.13)\\
&= \frac{1}{2} \left(\sum_{i,j \in A} w_{ij} (f_i - f_j)^2 + \sum_{i,j \in \bar{A}} w_{ij} (f_i - f_j)^2 + \sum_{i \in A, j \in \bar{A}} w_{ij} (f_i - f_j)^2 + \sum_{i \in \bar{A}, j \in A} w_{ij} (f_i - f_j)^2 \right) && (2.14)\\
&= \frac{1}{2} \left(\sum_{i \in A, j \in \bar{A}} w_{ij} (f_i - f_j)^2 + \sum_{i \in \bar{A}, j \in A} w_{ij} (f_i - f_j)^2 \right) && (2.15)\\
&= \frac{1}{2} \left(\sum_{i \in A, j \in \bar{A}} w_{ij} \left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 + \sum_{i \in \bar{A}, j \in A} w_{ij} \left(-\sqrt{\frac{|A|}{|\bar{A}|}} - \sqrt{\frac{|\bar{A}|}{|A|}} \right)^2 \right) && (2.16)\\
&= \frac{1}{2} \left(C(A, \bar{A}) \left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 + C(\bar{A}, A) \left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 \right) && (2.17)\\
&= C(A, \bar{A}) \left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2 \right) && (2.18)\\
&= C(A, \bar{A}) \left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + \frac{|A|}{|A|} + \frac{|\bar{A}|}{|\bar{A}|} \right) && (2.19)\\
&= C(A, \bar{A}) \left(\frac{|V|}{|A|} + \frac{|V|}{|\bar{A}|} \right) && (2.20)\\
&= |V| \left(\frac{\mathrm{cut}(\bar{A}, A)}{|A|} + \frac{\mathrm{cut}(A, \bar{A})}{|\bar{A}|} \right) && (2.21)\\
&= |V| \, \mathrm{RatioCut}(A, \bar{A}) && (2.22)
\end{aligned}$$

which implies

$$\frac{f^\top \mathbf{L} f}{|V|} = \mathrm{RatioCut}(A, \bar{A})$$

Note that

$$\begin{aligned}
f^\top f &= \sum_i f_i^2 \\
&= \sum_{i \in A} f_i^2 + \sum_{i \in \bar{A}} f_i^2 \\
&= \sum_{i \in A} \frac{|\bar{A}|}{|A|} + \sum_{i \in \bar{A}} \frac{|A|}{|\bar{A}|} \\
&= |\bar{A}| + |A| = |V|
\end{aligned}$$

so we can replace the left-hand side of the expression above by the Rayleigh quotient of f and $\mathbf{L}$

$$\frac{f^\top \mathbf{L} f}{f^\top f} = \operatorname{RatioCut}(A, \bar{A})$$

Solving

$$\min_{A \neq \emptyset \subset V, f \text{ indicator of } A} \frac{f^\top \mathbf{L} f}{f^\top f}$$

is thus equivalent to solving Problem (2.5). Note that the indicator function is a constant iff $A, \bar{A} \neq \emptyset$ and so

$$\min_{A \subset V, f \text{ indicator of } A, f^\top \mathbf{1} = 0} \frac{f^\top \mathbf{L} f}{f^\top f}$$

If we relax the requirement that f is the indicator above, by the Courant–Fischer–Weyl min-max principle, minimizing the Rayleigh quotient over the space orthogonal to the first eigenvalue yields the second eigenvalue. The f for which this happens is called the *Fiedler vector* u_f. From (2.1), we see that minimizing this quotient with these constraints corresponds to finding the *smoothest* vector (in the sense that the differences over adjacent nodes are small — and more so for strongly connected nodes), that is not constant. To recover the partition we choose clusters according to the sign of the eigenvector components, nodes i, j being in the same cluster if $\operatorname{sign}(u_f[i]) = \operatorname{sign}(u_f[j])$. Components with different signs are guaranteed to exist because the second eigenvalue is orthogonal to constants.

2.2.2 Generalizing to $k > 2$

We begin by generalizing the definition of RatioCut given above:

$$\operatorname{RatioCut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\operatorname{cut}(A_k, \bar{A}_k)}{|A_k|} \tag{2.23}$$

We define $F \in \mathbb{R}^{N \times k}$ as

$$F[i, j] = f_{ji} = \begin{cases} 1/\sqrt{|A_j|} \text{ if } v_i \in A_j \\ 0 \text{ otherwise} \end{cases} \tag{2.24}$$

We observe that

$$
\begin{aligned}
\left(F^\top F\right)_{ij} &= \sum_k f_{ik}^\top f_{kj} \\
&= \sum_k f_{ki} f_{kj} \\
&= \sum_{k \in A_i \wedge k \in A_j} f_{ki} f_{kj} + \sum_{k \notin A_i \vee k \notin A_j} f_{ki} f_{kj} \\
&= \sum_{k \in A_i \wedge k \in A_j} \frac{1}{|A_j|} \\
&= \delta_{ij}
\end{aligned}
$$

where the last step follows from noting that the same vertex cannot be in more than one element of the partition and that we are summing over all elements of that partition.

Observe that using Eq. (2.6) and (2.7)

$$
\begin{aligned}
f_k^T \mathbf{L} f_k &= \frac{1}{2} \sum_{i,j} w_{ij} (f_{ki} - f_{kj})^2 \\
&= \frac{1}{2} \left(\sum_{i,j \in A_k} w_{ij} (f_{ki} - f_{kj})^2 + \sum_{i \notin A_k \vee j \notin A_k} w_{ij} (f_{ki} - f_{kj})^2 \right) \\
&= \frac{1}{2} \left(\sum_{i,j \in A_k} w_{ij} (f_{ki} - f_{kj})^2 + \sum_{j \in A_k, i \notin A_k} w_{ij} (f_{ki} - f_{kj})^2 + \sum_{i \in A_k, j \notin A_k} w_{ij} (f_{ki} - f_{kj})^2 + 0 \right) \\
&= \frac{1}{2} \left(\sum_{i \notin A_k, j \in A_k} w_{ij} \frac{1}{|A_k|} + \sum_{i \in A_k, j \notin A_k} w_{ij} \frac{1}{|A_k|} \right) \\
&= \frac{1}{2} \left(\frac{C(\bar{A_k}, A_k)}{|A_k|} + \frac{C(A_k, \bar{A_k})}{|A_k|} \right) \\
&= \frac{\operatorname{cut}(A_k, \bar{A_k})}{|A_k|}
\end{aligned}
$$

Which implies that

$$
\begin{aligned}
\operatorname{Tr}(F^\top \mathbf{L} F) &= \sum_k f_k^T \mathbf{L} f_k \\
&= \sum_k \frac{\operatorname{cut}(A_k, \bar{A_k})}{|A_k|}
\end{aligned}
$$

which, as in the case $k = 2$ suggests solving a relaxed problem.

2.3 Hour 3 (Wednesday 11-12 am)

2.3.1 Signal Processing on Graphs

Goal 1: create models of smooth graph signals and operators to manipulate them (GSP)

For L a symmetric matrix ($L = u^\top \mathrm{D}iag(\lambda_1, \dots, \lambda_N)u$ for $u = (u_1, \dots, u_N)$ an orthonormal matrix), a Laplacian matrix in the sequel, and f a real-valued function, let $f(L)$ be defined as $L = u^\top \mathrm{D}iag(f(\lambda_1), \dots, f(\lambda_N))u$: f is applied on the eigenvalues. Typical example: diffusion on graphs, defined as

$$\partial_t f = -Lf,$$

where L is a graph Laplacian, yielding $f_t = e^{-tL} f_0$ and making appear the graph Fourier, defined as $\hat{f}(\ell) = \sum_j f(j)u_\ell(j)$, and the graph Fourier inverse.

Graph Fourier transform of f a signal: $\hat{f}(\ell) = \sum_j f(j)u_\ell(j)$, and its Fourier inverse $f(i) = \sum_k \hat{f}(k)u_k(i)$.

f is a M-smooth signal on a graph if ($\|S^\top M\|_2^2 \leq M$), or equivalently if $|\hat{f}(\ell)| \leq M\lambda_\ell$.

Definition 6. *A graph filter is an operator acting on graph signals f_{in}, represented as a function of the Laplacian: $g(L)$, leading to $f_{out} = g(L)f_{in}$.*

We have $\widehat{g(L)f}(\ell) = g(\lambda_\ell)\hat{f}(\ell)$.

Spectral kernels:

Localization: define $(T_i g)(n) = g(L)_{i,n}$ (is that it?)

Polynomial localization: g polynomial of degree less than $K \implies$ if the distance between i and n in the graph is more than K, then $(T_i g)(n) = 0$.

Efficient implementation of a graph filter Hammond et al. [2011] The image $g(L)f$ of a graph signal $f \in \mathbb{R}^N$ by a graph filter $g(L)$ may be approximately computed without having to compute spectral decomposition of the Laplacian matrix ($O(N^3)$ operations with SVD). Indeed, if g has some kind of smoothness, it can be approximated by a polynomial $g(L) \simeq \sum_{k=1}^{K-1} a_k L^k$. Computing Lf, then $L^2 f = L(Lf)$ and so on until $L^{K-1}f$ can be done in $O(KN^2)$ operations. If L is sparse, using a sparse matrix representation allows to actually use only $O(KM)$ operations (recall that M is the number of edges while N is the number of vertices)?

Summary of this section. We have a controlled way to define a graph filter g as localized as we want: the smoother it is, the better it can be approximated with polynomials and hence well localized. Moreover, we have an efficient way to evaluate the filter on a signal.

2.3.2 Designing and Processing Graph Features with Graph Signal Processing

Graph wavelets. Let us now first show how to design an analog of wavelets on graphs Hammond et al. [2011], wavelets being a classical powerful signal processing tool on Euclidean domains. The main motivation for wavelets is that it allows to define a filter localized both in time/space and frequency. Recall that the Fourier transform of a graph signal $f \in \mathbb{R}^N$ is defined by $\hat{f}(\ell) = \langle u_\ell, f \rangle$ and a graph filter $g(L)$ operates as $g(L)f(n) := \sum_{\ell=1}^{N} g(\lambda_\ell)\hat{f}(\ell)u_\ell(n)$. As for classical wavelets, one can translate and dilate a "mother wavelet" g in order to define wavelets with different localizations in time/spatial and frequency domains. In order to control the localization in the frequency domain, we scale g with a scale parameter $s \in \mathbb{R}$:

$$D_s g := g(sL).$$

For instance, when $s < 1$, it dilates g in the frequency domain ($\lambda \mapsto g(\lambda)$ is replaced by $\lambda \mapsto g(s\lambda)$) likely resulting in a smoother function than g in the sense that it has smaller variations than g within "communities" of the graphs hence it is less spread on the nodes of the graph. Decreasing the scale reduces the spread of the wavelet, see Figure 2.1.

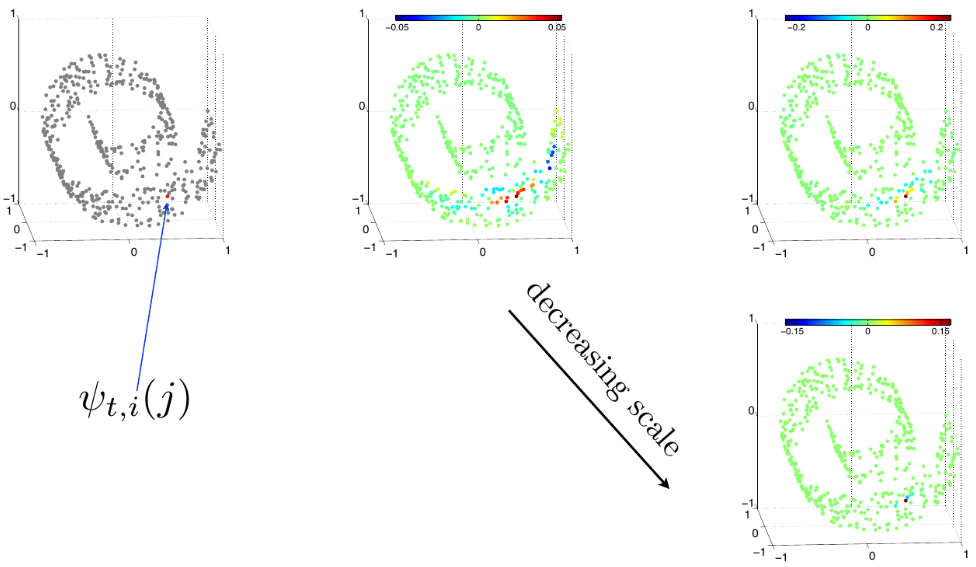

Figure 2.1: Illustration of the phenomenon of decreasing scale on a toy spiral dataset.

In order to control the localization in the time/spatial domain, we apply $g(L)$ on the impulse δ_n localized on a single vertex $n \in V$. This defines an analog of the translation operator but for graph:

$$T_n g(i) := \sum_{\ell=1}^{N} \hat{g}(\lambda_\ell) u_\ell(n) u_\ell(i).$$

Definition 7. *(Spectral Graph Wavelets) Let $g(L)$ be a graph filter. The associated wavelet operator scaled by s and centered at vertex n is defined as:*

$$\psi_{s,n}(i) := (T_n D_s g)(i) = \sum_{\ell=1}^{N} \hat{g}(s\lambda_\ell) u_\ell^*(n) u_\ell(i).$$

Neighbourhoods-dependent features based on wavelets. Let $g(L)$ be a graph filter. For every vertex n, the feature $\|\psi_{1,n}\|_2^2 = \|T_n g\|_2^2 = \sum_{\ell=1}^{N}(g(\lambda_\ell))^2|u_\ell(n)|^2$ depends on the structure of the neighbourhood. For instance, if g is K-localized, then by definition $T_n g$ only depends on the k-nearest neighbours of vertex n. In order to change the spread of the filter g, one can more generally consider a scale $s \in \mathbb{R}$ and associate to every vertex n the feature $\|\psi_{s,n}\|_2^2$. Note that if the scale is too small, these features will not discriminate the vertices since $\psi_{s,n}$ is then localized at the vertex n for every n. Similarly, when s is too large, $\psi_{s,n}$ is localized on the whole graph and the features are non-discriminant.

A linear interpolation of features Shuman et al. [2016]. Let us now describe a simple interpolation example where we recover missing features corresponding to vertices in $V_1^c := V \setminus V_1$ based on the known features f_{V_1} associated with the vertices in V_1. What follows is a simplified version of Pesenson's variational spline interpolation. We consider $\varepsilon > 0$, a regularized version of the Laplacian $\bar{L} := L + \varepsilon I$ and we define $\varphi_j := T_j g$ with g that acts on the spectrum of $\bar{L}$ as $g(\lambda) = \frac{1}{\lambda}$. Note that the functions φ_j are regularized Green's functions since they satisfy $\bar{L}\varphi_j = \delta_j$. Define α_* as:

$$\alpha_* := [\bar{L}_{V_1,V_1} - \bar{L}_{V_1,V_1^c}(\bar{L}_{V_1^c,V_1^c})^{-1}\bar{L}_{V_1^c,V_1}]f_{V_1}.$$

Upsample α_* to define a signal on the whole graph:

$$f_{\text{upsample}} = \sum_{n \in V_1} \alpha_*(n)\delta_n.$$

We now interpolate the original signal f as follows:

$$f_{\text{interp}} := g(\bar{L})f_{\text{upsample}}.$$

One can check that this coincides with f on V_1 where the signal was known. Moreover, this choice of interpolation minimizes, over all $\hat{f}$ that coincides with f on V_1, some kind of energy (typically $\hat{f}^T\bar{L}\hat{f}$).

2.4 Hour 4 (Thursday 2-3 pm)

In the previous lectures on Graph Representation Learning we:

- learned that the eigendecomposition of the Laplacian of graph reveals a lot about its structure
- leveraged the smoothness of Eigenvectors for partition vectors / signals
- did dimensionality reduction unsupervised
- came up with the idea of operators that can be applied to graph filters
- set up an algorithm that allowed us to apply these operators in a computational efficient way to signals, which can be used to craft specific features for specific applications.

This can be summarized as **LOVE** - “**L**aplacian **O**rthogonal eigen**VE**ctors” (see Figure 2.2).

Another topic we discussed and that we will now explore further is Spectral Clustering.

Figure 2.2: Beatles (1967) and Vandergheynst (2022) - “All You Need Is LOVE”

2.4.1 More Spectral Clustering

Let G be a large graph, which is guaranteed to have many communities. The purpose of this section is to show tools that allow to find the communities of G with a low computational budget (in this regime even performing spectral clustering would be too expensive). The idea is to sample (in a good way) m nodes on the graph with the hope that this sub-graph will be a good representation of G, in the sense that it reveals the k clusters of the graph.

The immediate challenge is to estimate k. This quantity is usually not given as input, in particular in unsupervised learning.

To address this problem, the main idea is to heuristically perceive a gap in the eigenvalues of the Laplacian L of G. The integer k is the number of the greatest eigenvalues of L before the gap. These k will help us to determine m. The search for k can be done in a stochastic approach. More details on these techniques can be found in Napoli et al. [2016].

From now on assume that a reasonable estimate of k has been computed and let us get back to the sampling of m vertices. A trivial lower bound on m to recover all k clusters of G would be $m \geq k$. We will not be able to get away with only $m = O(k)$ samples. But we will show that up to some parameters, with high probability $m = O(k \log(k))$ many samples suffice.

Let $L = U\Lambda U^\top$ be the eigendecomposition of L such that diagonal entries of Λ are ordered in increasing order: $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

Then, we sample m nodes independently, where node i is sampled with probability

$$p_i^* := \frac{\|U_k^\top \delta_i\|_2^2}{k}, \tag{2.25}$$

where U_k is the submatrix consisting of the first k columns of U and δ_i is the i-th vector of the standard basis in $\mathbb{R}^n$.

If we represent the action of sampling $m \leq n$ nodes by a matrix $M \in \{0,1\}^{m \times n}$, and grouping the the probability of a node to be selected in a diagonal matrix P we can derive guaranties on the compression-decompression of the graph if M is chosen wisely. More formally:

Lemma 2.2. *There exists a distribution p^* over the nodes of G, such that if $M \sim p^*$ then for all $0 < \varepsilon, \delta < 1, \forall x \in \operatorname{span}(U_k)$*

$$\mathbb{P}\left[(1-\delta)\|x\|_2^2 \leq \frac{1}{m}\|MP^{-\frac{1}{2}}x\|_2^2 \leq (1+\delta)\|x\|_2^2\right] \geq 1-\varepsilon$$

for $m = \Theta\left(\frac{1}{\delta^2} k \log\left(\frac{k}{\varepsilon}\right)\right)$.

However, recall that we mentioned at the beginning of this section that computing such a decomposition would computationally be too expensive. Fortunately, approximating the p_i is sufficient for our purposes and such an approximation can be obtained efficiently with high probability:

What is interesting is that we can estimate the optimal distribution of the above assertion by applying random filters and taking the mean.

The recovering procedure of the graph is given by:

$$\min_{z \in \mathbb{R}^n} \|P_\omega^{-\frac{1}{2}}(Mz - y)\|_2^2 + \gamma \underbrace{z^T g(L) z}_{A}$$

Where $y = Mx$ and A is soft constraint of frequencies.

The experiments conducted show the difference in reconstruction performance depending on the distribution chosen on the graph. In particular, the uniform distribution is generally a bad idea this task. Another experiment carried out on the graph of proximity of pixels of an image (for a given image we associate a graph weighted by the proximity between two pixels) shows that we can reconstruct the image quite faithfully by sampling 7% of the nodes.

2.5 Hour 5 (Thursday 3-4 pm)

2.5.1 Graph Neural Networks

Good recipe for Neural Networks : Convolutional NN shared weights: parallelization, shift invariance.

Adapting it to graphs

Idea 1 : work at the node level to compute local aggregator *Idea 2 :* make computation scalable by weight sharing ("convolutional net")

Look at neighbors to aggregate and compute the features.

Generalization of CNN to Graph : see article ChebNet

- ChebNet : At each layer use a graph filter and learn parameters of the filter : polynomial filter for computations
- **GCN - Simplified architecture :** Only use linear filters because using multiple layers has "similar" effect Use same weights for nodes and their neighbors Rescale for stability and have a more uniform effect across the graph

Some examples of the matrix weights of GNNs:

1. $W = UD_\theta(\Lambda)U^\top$.
2. $W = UG_\theta(\Lambda)U^\top$.
3. (Chebnet?) $W = P_\theta(L)$.
4. Simplification: $W = \theta_1 \mathbf{I} - \theta_2 A$ (can be simplified even more by setting $\theta = \theta_1 = -\theta_2$).

where the next features (neurons) are computed as: $F_i(x) = \sigma([W]_i^\top x + b)$.

Message Passing GNNs in GNN : Compute local features, transfer message to the neighbors, re-scale the neighbors, receive message from the neighbors to agglomerate/update your state. Repeating the operation is similar to receiving signals from further and further neighbors. This is what ChebNet does: it tries to learn how to transmit the message through the polynomial filters.

This approach can be generalized to non-polynomial filters. Three main steps :

- Compose message: $m^{t+1}(i \longrightarrow j) = \mathcal{M}(f_t(x_i), f_t(x_j), a_{ij})$.
- Aggregate messages: $\mathcal{A}(m^{t+1}(j \longrightarrow i), j \in \mathcal{N})$.
- Update states: $f_{t+1}(x_i) = \mathcal{U}(f_t(x_i), \mathcal{A}(m^{t+1}(j \longrightarrow i), j \in \mathcal{N}))$.

To finalize this approach, depending on the task at hand : node or graph.

Attention-based GNNs : Graph Attention Networks (GATs), Non-Local Neural Networks (NLNNs) Use a self-attention mechanism t weight node features The attention can be generalized : there are weights to decide if at a certain level of the network we use the information at any graph even if they are not connected directly.

Unifying view of GNNs : Message Passing General properties arise from this general idea : symmetries see slides

Possibility to add constraints justified by the problem setting (eg: node equivariant message passing functions)

Applications :

- E-commerce
- Anonymize Networks
- Biology : Classify sites on proteins with their respective nodes (Pocket classification, interface prediction, Ultra-fast PPI search : learn proteins interactions and compatibility)

2.5.2 Analyzing GNNs

Check how expressive GNNs are : how good they can distinguish non-isomorphic graphs. Weisfeiler-Leman test : assign a color to nodes of each graph (same color for every one at beginning), update the color via an injective hash based on the previous color and the color of the neighbors of each node. Finally, compare color histograms.

2.6 Hour 6 (Friday 2-3 pm)

2.6.1 Understanding GNNs

WL Failure There are non-isomorphic graphs that cannot be distinguished by 1-WL (e.g. Decalin & Bencyclopentyl).

How does GNN compare to WL? Both are equivalent. If a GNN can distinguish two graphs, 1-WL can distinguish them as well. Conversely, if 1-WL can distinguish two graphs, with sufficient depth a GNN can distinguish them as well.

Takeaway The paper *How Powerful are Graphs Neural Networks* characterizes the kind of injective multisite functions.

2.6.2 Beyond GNNs: exploiting higher order structures

To distinguish two graphs, we can look for motives like triangles. This is what the k-WL test does. Question: what is the neighbourhood of a set of nodes? (k+1)-WL is more expressive than k-WL.

Simplicial Neural Nets We want to exploit higher-order structure in a more simplified way. The idea is to create a simplex and then keep increasing the structure. But the subsets of the higher structure must be contained by the lower level.

Definitions:

1. boundary (e.g. the boundary of a triangle are its edges)
2. boundary incidence
3. signed boundary matrices
4. k-th Hodge Laplacian ... We can prove that L_k is positive semi-definite.

Simplicial Weisfeiler-Leman We exploit richer structures of **adjacencies in simplicial complexes**:

- boundary adjacency
- co-boundary adjacency
- lower adjacency
- upper adjacency

We use aggregation over adjacent structures in WL fashion. It is not less powerful than 3-WL

2.6.3 Conclusion

There are limitations of GNNs that we do not understand yet. First problem: we can make it more expressive by adding more layers. Since we average the layers, the last layers, that are the most important to take meaningful decisions, get inexpressive.

Second problem: over-squashing.

3 Multi-armed bandits and beyond

Note takers: Achddou J., Barrier, Chapuis, Ganassali, Haddouche, Marthe, Saad, Tarrade, Van Assel

Instructor: Shipra Agrawal (Columbia University)

3.1 Introduction

When learning from sequential interactions, there is a tradeoff between:

- information and rewards:
- learning and optimization,
- exploration and exploitation.

In a nutshell, your goal is to get a maximal reward at each round and you have to choose between:

- **exploitation**: choosing an option that ensures you a good reward using information that you gathered in the past,
- **exploration**: choosing an option that has been less profitable in the past but on which you did not collect enough information, which might cost you a bad immediate reward but will allow you to understand better the system and thus obtain better rewards in the future.

We will start with introducing the basic Multi-armed Bandits problem. Multi-armed Bandits and Reinforcement Learning problems deal with the tradeoffs hereabove, gathered under the term *exploration/exploitation dilemma.*

Examples:

- Learn from customer's feedback to improve what products to show on internet. In this case, the outcome is whether or not the customer purchased the product,
- Machines at the casino : which one to put money on?

  Given an amount of money, you are free to choose in which machine(s) you will put it. Your goal is to maximize the money you will earn in total.

  Are you gonna try only one machine? Or all of them uniformly at the beginning, and then stop to use the bad ones?

  When to stop trying (exploration) and start playing (exploitation)?

Stochastic Multi-armed Bandit problem (MAB). We consider a setting of online decisions: at every round $t \in [T] = \{1, \dots, T\}$, we pull one arm $i_t \in \{1, \dots, N\}$ out of N arms using past information.

As a feedback, for each arm $i \in [N]$, a reward $r_{i,t}$ is generated i.i.d. from a **fixed but unknown distribution** with support in $[0, 1]$ and mean μ_i. The learner only observes the reward $r_t = r_{i_t,t}$ of the pulled arm i_t. The mean of the reward at time t (knowing that arm i_t has been selected) is thus μ_{i_t}.

The goal is to minimize the *regret* compared to the best arm $i^* = \arg\max_i \mu_i$:

$$\mathrm{Regret}(T) = \mathbb{E}\left[\sum_{t \in [T]} \mu^* - \mu_{i_t}\right] = T\mu^* - \sum_{t \in [T]} \mathbb{E}[\mu_{i_t}]$$

where $\mu^* = \mu_{i^*} = \max_i \mu_i$ (note that i_t is a random variable, hence the expectation in the definition of $R(T)$).

If we know the best arm i^*, we can play $i_t = i^*$ at every round and get the optimal reward $T\mu^*$ (and regret 0): the regret is defined as the difference between what one could ideally have obtained and what we actually got at the end.

For each arm i, denoting by $\Delta_i = \mu^* - \mu_i$ its gap and by $k_i(T)$ its number of pulls up to time T, the expected regret can be rewritten as

$$\mathrm{Regret}(T) = \sum_{i \neq i^*} \Delta_i \mathbb{E}[k_i(T)]$$

Of course, a strategy that learns something from the data will diminishes the observation frequency of any sub-optimal arm, hence its regret might be sublinear. At the opposite, if the proportions of pulls of sub-optimal arms do not evolve with time, the strategy does not learn anything and the regret is linear.

Outline. We will first cover basic algorithms involving UCB and Thompson Sampling. Then we will see useful generalizations: contextual bandits, bandits with constraints, assortment of bandits. We will finally study bandit techniques for MDP/RL.

The need for exploration. We will use a toy example to highlight how randomness forces the use of exploration.

We consider $N = 2$ arms: blue and red, with respective means $\mu_1 = 1.1$ and $\mu_2 = 1$. The optimal expected reward in this case is $1.1 \times T$.

A first natural strategy consists to simply pull the arm with the current best estimate (MLE/empirical mean) of unknown mean. This strategy is called Follow-The-Leader.

But we will see that initial trials can be misleading. Assume for instance that arm red is a Dirac distribution at 1: the associated sequence of rewards will be $1, 1, 1, \ldots$. If arm blue has quite a large variance, there is a positive probability that its empirical mean after a few trials will be lower than 1.

On this event denoted by E, you will then pull the red arm at every time step (as it empirical mean 1 will never go down under the empirical mean of the blue arm. Thus your expected regret after T steps will satisfy

$$\text{Regret}(T) \gtrsim \mathbb{P}(E) \times 0.1T$$

hence the regret will be linear in T: the strategy fails to learn anything on event E. To correct this misbehaviour you have to **pay attention to exploration**!

As already explained, a good algorithm will have to balance between:

- **exploitation**: play the empirical mean reward maximizer,
- **exploration**: play less explored actions to ensure the convergence of empirical estimates.

3.2 Lower bounds

An algorithm has no way to know whether an arm is sub-optimal before it plays it. Thus it will have to observe sub-optimal arms at least a few times, leading to a non-negative regret. This has been quantified in the literature by Lai and Robbins (1985): for *any* given instance $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_N)$ of the MAB problem, any "reasonable" algorithm will play a sub-optimal arm i at least $\Omega(\frac{\log(T)}{\Delta_i^2})$ times for large T, hence a minimal regret of

$$\text{Regret}(T) \gtrsim \log(T) \sum_{i \neq i^*} \frac{1}{\Delta_i} .$$

This bound is instance-dependant: it depends on the distributions of the bandit parameter $\boldsymbol{\mu}$, more specifically through the gaps Δ_i.

Remark. *Imagine a strategy that always selects $i_t = 1$: it will have a 0 regret among all bandit parameters such that $i^* = 1$, but a linear regret among all other bandit parameters. Then this is not a good strategy, and this is the kind of strategy we remove when considering only "reasonable" strategies.*

On the other hand, there also exists a worst case bound: for every algorithm, there exists a bandit parameter for which $R(T) = \Omega(\sqrt{NT})$.

3.3 The Upper Confidence Bound algorithm (Auer 2002)

The strategy. We define the empirical mean at time t for am i as follows:

$$\hat{\mu}_{i,t} = \frac{\sum_{s\in[t]} r_s \mathbf{1}_{i_s=i}}{k_i(t)} .$$

As we already discussed, the empirical mean is not sufficient to capture the need for exploration. The idea of the UCB algorithm is to combine at each time t and for each arm i both an exploitation term (the empirical mean) and an exploration term (which is a bonus that decreases with the number of pulls) into an index denoted by $\text{UCB}_{i,t}$:

$$\text{UCB}_{i,t} = \underbrace{\hat{\mu}_{i,t}}_{\text{exploitation term}} + \underbrace{2\sqrt{\frac{\log t}{k_i(t)}}}_{\text{exploration term}}$$

The strategy is optimistic: we know that μ_i belongs to the confidence interval $[\hat{\mu}_{i,t} \pm 2\sqrt{\frac{\log t}{k_i(t)}}]$ w.h.p. and we take the highest value of this interval as basis: $\text{UCB}_{i,t}$ overestimates μ_i. While increasing the number of observations this confidence region will shrink to $\{\mu_i\}$.

The UCB algorithm plays at time t the arm with the best optimistic estimates, as explained in Algorithm 1.

Regret analysis of UCB. Recall the expression of regret:

$$\text{Regret}(T) = \sum_{i\neq i^*} \Delta_i \mathbb{E}[k_i(T)]$$

We assume optimistically that for any i, $UCB_{i,t} > \mu_i$.

First we bound the number of mistakes $\mathbb{E}[k_i(T)]$ for all suboptimal arms $i \neq i^*$.

A bound of $\mathbb{E}[k_i(T)] \leq \frac{C \log T}{\Delta_i^2}$.

Arm i will be played at time t only if $\text{UCB}_{i,t} > \text{UCB}_{i^*,t}$.

```
Input: number of arms $N$, number of steps $T$
Observe each arm once
$t \leftarrow N$ while $t < T$ do
    for each arm $i$ do
        Compute $\mathrm{UCB}_{i,t} = \hat{\mu}_{i,t} + \sqrt{\frac{4 \log t}{k_i(t)}}$
    end
    $i_{t+1} \leftarrow \arg\max_i \mathrm{UCB}_{i,t}$
    Observe $r_{t+1} = r_{i_{t+1},t}$
    $t \leftarrow t + 1$
end
```

Algorithm 1: Upper Confidence Bound

If $n_{i,t} > \frac{16 \log T}{\Delta_i^2}$, we get $|\hat{\mu}_{i,t} - \mu_i| \leq \frac{\Delta_i}{2}$ with probability $1 - \frac{1}{T^2}$ using Azuma-Hoeffding inequality, and then

$$\mathrm{UCB}_{i,t} - \hat{\mu}_{i,t} = \sqrt{\frac{4 \log t}{n_{i,t}}} \leq \frac{\Delta_i}{2}$$

With high probability, arm i will not be pulled more than $\frac{16 \log(T)}{\Delta_i^2}$ (bound on expected number of mistakes), thus with high probability

$$\mathrm{Regret}(T) \leq 16 \log(T) \sum_{i \neq i^*} \frac{1}{\Delta_i}$$

3.3.1 Thompson Sampling (Thompson 1933)

Thompson Sampling is a Bayesian algorithm. The general idea is to maintain belief about parameters (*e.g.* mean reward) of each arm. Then observe the feedback, update the belief of pulled arm in a Bayesian manner. Belief update is performed using Bayes rule: the posterior is proportional to the product of likelihood and prior. Importantly, we don't try to estimate the parameters in this setting.

We pull the arm by sampling from the posterior probability of being the best arm. Note that this is different than choosing the arm that is the most likely to be the best.

The main intuition of maintaining Bayesian posteriors is the following:

- When the number of trials increases, the posterior concentrates on the true parameters. This phenomenon enables exploitation, as the mode of the posterior captures the maximum likelihood estimate.
- Moreover, uncertainty is high when the number of trials is small. This variance captures the uncertainty about the arms and enables exploration.

Example of Bernoulli rewards with Beta priors. In the case of Bernoulli rewards, we pick the Beta distribution since it is *conjugate* (it is important because drawing a point according to updated Bayesian posterior may be costly in the general case and often requires MCMC methods). If you take $\mathrm{Beta}(\alpha, \beta)$ as a prior, then the posterior is updated as follows:

- $\mathrm{Beta}(\alpha + 1, \beta)$ if you observe 1.
- $\mathrm{Beta}(\alpha, \beta + 1)$ if you observe 0.

Note that every time you observe a sample, the variance decreases[1].

We start with a $\mathrm{Beta}(1, 1)$ distribution as prior belief for every arm. Then in round t:

- for every arm i, sample $\theta_{i,t}$ independently from current posterior $\mathrm{Beta}(S_{i,t} + 1, F_{i,t} + 1)$, where:
$$S_{i,t} = \sum_{s\in[t-1]} \mathbf{1}_{r_s=1}\mathbf{1}_{i_s=i} \quad \text{and} \quad F_{i,t} = \sum_{s\in[t-1]} \mathbf{1}_{r_s=0}\mathbf{1}_{i_s=i}\,,$$
- play arm $i_{t+1} = \arg\max_i \theta_{i,t}$,
- observe reward and update the Beta posteriors
$$F_{i,t+1} = \begin{cases} F_{i,t} & \text{if } i \neq i_{t+1} \\ F_{i,t} + \mathbf{1}_{r_{t+1}=0} & \text{if } i = i_{t+1} \end{cases} \quad \text{and} \quad S_{i,t+1} = \begin{cases} S_{i,t} & \text{if } i \neq i_{t+1} \\ S_{i,t} + \mathbf{1}_{r_{t+1}=1} & \text{if } i = i_{t+1} \end{cases}$$

Example of continuous rewards with Gaussian priors. We take a standard $\mathcal{N}(0, 1)$ prior. The reward likelihood is $\mathcal{N}(\hat{\mu}, 1)$ such that the posterior after n independent observations simply takes the form $\mathcal{N}(\hat{\mu}, \frac{1}{n+1})$ where $\hat{\mu}$ is the empirical mean.

Start with $\mathcal{N}(0, v^2)$ prior belief for every arm. Then in round t:

- for every arm i, sample $\theta_{i,t}$ independently from current posterior $\mathcal{N}(\hat{\mu}_{i,t-1}, \frac{v^2}{n_i(t-1)+1})$,
- play arm $i_t = \arg\max_i \theta_{i,t}$,
- observe reward and update the empirical mean $\hat{\mu}_{i,t}$.

Remark: In practice Thompson sampling seems to be more efficient in general than UCB since UCB involves the optimistic assumption of the overestimation of the mean which may not be realistic.

Why does it work? For the sake of simplicity, we come back to the two arms example: we consider two arms with $\mu_1 \geq \mu_2$, $\Delta = \mu_1 - \mu_2$. In this case we directly have that if arm 2 is pulled, the regret is Δ.

We want to bound the number of pulls of arm 2 by $\frac{\log T}{\Delta^2}$ to get a $\frac{\log T}{\Delta}$ regret bound.

[1]the variance of $\mathrm{Beta}(\alpha, \beta)$ is $\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$

How many pulls of arm 2 are actually needed?

After $n \geq \frac{16\log(T)}{\Delta^2}$ pulls of arm 2 and arm 1.

Using Azuma-Hoeffding, one has : $|\hat{\mu_i} - \mu_i| \leq \sqrt{\frac{\log(T)}{n}} \leq \frac{\Delta}{4}$ with high probability. So the arms are well separated.

Beta posteriors are well separates: their mean is $\frac{\alpha_i}{\alpha_i+\beta_i} = \hat{\mu}_i$ and standard deviation about

$$\frac{1}{\sqrt{\alpha+\beta}} = \frac{1}{\sqrt{n}} \leq \frac{\Delta}{4}$$

Thus the 2 arms can be distinguished and arm 2 will not be pulled anymore. Hence the importance of verifying both arms have been pulled enough to ensure the consistency of our result.

Extension to multiple arms. One has the following kind of results:

$$\mathbb{P}(a_t = a^*|F_{t-1}) \geq \frac{p}{1-p}\mathbb{P}(a_t = a|F_{t-1})$$

where p is the probability of anti-concentration of posterior sample for the best arm.

Best arm gets played roughly every $1/p$ plays of arm a.

- p can be lower bounded by Δ_a in general but it actually goes to 1 exponentially fast with increase in number of trials of best arm,
- cannot accumulate from arm a without playing a^* sufficiently.

3.4 Useful generalizations of the basic MAB problem

Different generalizations could be useful depending on the application:

- pulling more than one arm at a time
- having unknown distributions
- changing the feedback (having it censored for instance)
- having a goal different than reward maximization

3.4.1 Handling context in MAB

In this part, we will only consider linear contextual bandit. They make sense in a lot of application, for instance in content based recommendation, where customers and product can be described by their features. It allows for an easier way of dealing with a large amount of products and customer types, and the features will allow to make profit of similarities across product or users.

Linear contextual bandits :

- N arms (possibly very large),
- a d-dimensional context (feature vector) $x_{i,t}$ for every arm i, time t
- Linear parametric model, with parameter θ
- algorithm picks $x_t \in \{x_{1,t}, \dots, x_{N,t}\}$, observe $r_t = x_t \cdot \theta + \eta_t$
- Optimal arm depends on context: $x_t^* = \arg\max_i x_{i,t} \cdot \theta$
- Goal : minimize regret $\sum_t (x_t^* - x_t) \cdot \theta$

UCB for contextual bandits Linear regression is used to approximate the parameter :

- least square solution $\hat{\theta}_t$ of set of equation $x_s \dot{\theta} = r_s$
- $\hat{\theta}_t \simeq B_t^{-1} \left(\sum_{s=1}^{t-1} x_s r_s\right)$ where $B_t = I + \sum_{s=1}^{t-1} x_s x_s^\top$

With high probability, we have the bound : $\|\theta - \hat{\theta}_t\|_{B_t} \leq C\sqrt{d \log(Td)}$

Remark. *The bound doesn't depend on the number of arm, only on the dimension.*

The algorithm proceeds as follows. At time t:

- Observe the context $x_{i,t}$ for different arms $i = 1, \dots, N$
- Compute optimistic parameter estimates and confidence intervals for every arm
- Choose the best arm according to the most optimistic estimates

$$\arg\max_i \max_\theta \theta^\top x_{i,t} \text{ such that } \left\| \theta - \hat{\theta} \right\|_{B_t} \leq C\sqrt{d \log(Td)}$$

```
for each $t = 1, \dots, T$ do
    Observe set $A_t \subseteq [N]$, and context $x_{i,t}$ for all $i \in A_t$
    Play arm $I_t = \arg\max_{i \in A_t} \max_{z \in C_t} z^\top x_{i,t}$ with $C_t$ as defined
    Observe $r_t$. Compute $C_{t+1}$
end
```

Algorithm 2: LinUCB algorithm

Proof.

$$\text{Regret}(T) = \sum_{t=1}^{T} (x_t^* \cdot \theta - x_t \cdot \theta)$$

with $x_t^* = x_{i^*,t}$, $i^* = \arg\max_i x_{i,t}^\top \theta$, and $x_t = x_{i_t,t}$

For the first part, we always make use optimism :

$$\begin{aligned}
R(T) &\leq \sum_{t=1}^{T} (x_t^* \cdot \theta - x_t \cdot \theta) \qquad \text{with high P} \\
&= \sum_{t=1}^{T} x_t(\tilde{\theta}_t - \theta) \\
&\leq \sum_{t=1}^{T} \sqrt{x_t^\top B_t^\top x} \sqrt{(\tilde{\theta}_t - \theta)^\top B_t (\tilde{\theta}_t - \theta)} \\
&= \sum_{t=1}^{T} \|x_t\|_{B_t^{-1}} \underbrace{\left\|\tilde{\theta} - \theta\right\|_{B_t}}_{\leq \sqrt{Cd\log(Td)}} \\
&= \sum_{t=1}^{T} \|x_t\|_{B_t^{-1}} \sqrt{Cd\log(Td)}
\end{aligned}$$

Moreover, by the Elliptical Potential Lemma (see e.g. Lattimore and Szepesvári [2020], Chapter 20) :

$$\sum_t \|x_t\|_{B_t^{-1}} = x_t^\top B_t^{-1} x_t = \tilde{O}(\sqrt{d})$$

which finishes the proof. □

Remark. *Thompson Sampling for linear contextual bandits uses the (Gaussian) Bayesian Linear Regression to sequentially maintain a posterior distribution over the unknown parameter θ. Regret guarantees currently show a slight suboptimality: $R(T) = O(d^{3/2}\sqrt{T})$ but it is still unclear whether this is due to an artefact in the proof or if that extra $\sqrt{d}$ should be here for more fundamental reasons.*

3.4.2 Assortement selection as multi-armed bandit

The customer response to the recommended assortment **may depend** on the combination of items and not just the marginal utility of each item, in the assortment.

Ex: An assortment combining 3 types of cell phones might push the user to go for the cheapest.

Setting:

- selecting a subset $S_t \in [N]$ in each of the sequential rounds $t = 1, \dots, T$.
- On selecting a subset S_t, reward r_t is observed with expected value $\mathbb{E}[r_t|S_t] = f(St)$ where the function $f : R^N \mapsto [0, 1]$ is unknown

Different possible structural assumptions on f (e.g. Lipschitz). But also multinomial logit choice model(MNL).

The multinomial logit choice model (Luce 1959, MfFadden 1978) is the following:

the probability that a consumer purchases product i at time t when offered an assortment S is

$$p_i(S) = \frac{e^{\theta_i}}{1 + \sum_i e^{\theta_i}} \text{ if } i \in S \cap \{0\} \text{ or } 0 \text{ otherwise} \tag{3.1}$$

i can be 0 meaning that there is no purchase.

The idea is to take into account the distribution of other arms. Pulling an arm no longer depends on its marginal distribution only. The above is a simple model to do so.

MNL bandit problem. In this setting we have N products, unknown θ_i. At every step t, we recommend an assortment S_t of size $\leq K$, observe customer response i_t, revenue r_{i_t}, and update parameter estimates. The customer's behavior is modeled by 3.1.

The goal is to optimize the total revenue $\mathbb{E}\left[\sum_t r_{i_t}\right]$, or minimize the regret compared to the optimal assortment

$$\mathcal{R}(T) := Tf(S^*) - \mathbb{E}\left[\sum_t r_t\right] = \sum_t (f(S^*) - f(S_t))$$

where $S^* = \max_S f(S)$. In many cases, even if the expected value $f(S)$ is known for all S, computing S may be intractable. Therefore, for this problem to be tractable some structural assumptions on f will be made.

Main challenges Censored feedback: feedback of product i is effected by other products in a given assortment (combinatorial: N^K choices). In other words, the response observed on offering a product i (as part of an assortment S) is not independent of other products in the assortment.

Technique to get unbiaised estimate of *individual parameters*: offer a given assortment S until no purchase: the number of times $n_S(i)$ that i is purchased in S on this process is an unbiased estimator of e^{θ_i}. Indeed, one has

$$\mathbb{E}[n_S(i)] = \frac{p_i(S)}{p_0(S)} = e^{\theta_i}.$$

Concretely, if at any round t a purchase of any item in the offered set S_t is observed, then the algorithm continues to offer the same assortment in round $t+1$, *i.e.* $S_{t+1} = S_t$. If a no-purchase is observed in round t, then the algorithm updates the parameter estimates and makes a new assortment selection for round $t+1$.

Then, having established confidence intervals for the parameters θ_i, we can run UCB and Thompson Sampling techniques.

UCB based algortihm Agrawal et al. [2019] acheives $O(\sqrt{NT})$ regret.

3.4.3 Bandits with constraints and non-linear aggregate utility

Generalizing MAB: we observe a non-negative reward r_t ans a cost vector c_t. The problem now becomes:

$$\max \sum_t r_t \text{ s.t. } \forall j, \ \sum_t c_{t,j} \leq B.$$

-> Bandits with Knapsacks Badanidiyuru et al. [2013]

A generalization of this is Bandits with convex knapsacks and concave rewards (BwCR), with convex constraints domains and concave rewards.

Bandits with convex knapsacks and concave rewards (BwCR) Agrawal and Devanur [2014] Pulling an arm i_t generates $v_t \in \mathbb{R}^d$ with unknown mean V_{i_t}.

Total number of pull constrained by T + arbitrary convex global contraints of the form $\frac{1}{n}\sum_t v_t \in S$, with S a convex set.

The goal is to maximize $f\left(\frac{1}{n}\sum_t v_t\right)$, for f an arbitrary concave function, of minimise $d\left(\frac{1}{n}\sum_t v_t, S\right)$ as one has to assure $\frac{1}{n}\sum_t v_t \in S$.

Results for UCB-like optimistic algo for BwCR We need to estimate for every arm i and coordinate j.

We are interested in the following problem, where $H_t = \{\overline{V} : \overline{V}_{ij} \in [\text{LCB}_{t,ij}, \text{UCB}_{t,ij}]\}$,

$$p_t = \arg_p \max_{\overline{V} \in H_t} f\left(\sum_i p_i \overline{V}_i\right) \tag{3.2}$$

$$\text{s.t.} \quad \min_{\overline{U} \in H_t} \text{dist}\left(\sum_i p_i \overline{U}_i, S\right) \leq 0 \tag{3.3}$$

For non-decreasing f, the inner maximizer in the objective of 3.3 will be simply the UCB estimate, therefore for the classic MAB problem this algorithm reduces to the UCB algorithm.

3.5 Bandit techniques for Markov Decision Processes

General formulation The general problem is as follows: the reward on pulling an arm (action) depends on the *current* state of the system. Each round $t = 1, \dots, T$ consists in observing the current state, taking an action, observing the reward and a new state. The solution concept is referred to as a *policy*. The general goal is to learn the state transition dynamics and the reward distributions, while optimizing the policy.

(Application: inventory management, autonomous vehicle control, robot navigation, personalized medical treatments...)

3.5.1 MDPs

at round t, the player observes state s_t, take action a_t, observe reward $r_t \in [0, 1]$ and next state s_{t+1}.

The system dynamics is given by a MDP (S, A, r, P, s_0) such that

$\mathbb{E}[r_t|s_t, a_t, H_t]\mathbb{E}[r_t|s_t, a_t] =: r_{s_t,a_t}$, and $\mathbb{P}[s_{t+1}|s_t, a_t, H_t]\mathbb{P}[s_{t+1}|s_t, a_t] =: P_{s_t,a_t}(s_{t+1})$.

Due to Markov property, there is an optimal policy $\pi : S \longrightarrow A$. The goal is to minimize expected regret compared to the best stationary policy π^*, defined as follows

$$\text{Regret}(M, T) := \sum_{t=1}^{T} \left[r(s_t^*, \pi^*(s_t^*)) - r(s_t, a_t)\right] .$$

We want to learn the MDP model parameters (r, P) from observations $(s_t, a_t, r_t, s_{t+1})_{1 \le t \le T}$, while optimizing the policy for total expected reward.

In these models, regret bounds will be of the form $O(S\sqrt{TA})$.

Need for exploration Let us look at a single state MDP: the situation boils down to the classical MDP problem, for which the exploit only policy may mislead into playing bad action forever.

[example of a two-states MDP] Let us illustrate the fact that exploiting the seemingly best policy is not the optimal choice. In the above example, initializing at state 1 and playing red action forever would avoid the best action (state 2, black action), which needs a bit of 'faith' in order to be discovered! In MDPs, the exploitation is thus even more important that in classical MAB.

Communicating MDPs Caveat: MDP can get stuck on bad states for a long time, depending on the underlying graph structure.

Let us define *communicating MDPs*, which are MDPs for which there is always a way to get out of a bad state in finite time. Namely, for every pair of states s, s', there is a policy π (*that* *depends on s, s')* such that using this policy starting from s, the expected time to reach s' is finite and bounded by D, called the *diameter* of the MDP.

Some useful properties:

- the optimal asymptotic average reward is independent of the starting state
- the asymptotic average reward (gain) of policy π is defined by

$$\lambda^{\pi}(s) := \mathbb{E}\left[\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T} r(s_t, \pi(s_t))\middle| s_1 = s\right]$$

  There is a single policy π^* achieving the optimal infinite average reward

$$\forall s,\ \max_{p} i\lambda^{\pi}(s) = \lambda^{\pi^*}(s) =: \lambda^*$$

- We define the regret as the gain compared to asymptotic normal:

$$\text{Regret}(M, T) := T\lambda^* - \sum_{t=1}^{T} r(s_t, a_t).$$

Some bounds Upper confidence bounds based algorithms Auer et al. [2008], Bartlett and Tewari [2012] : worst-case regret bound $O(DS\sqrt{AT})$, lower bound $\Omega(\sqrt{DSAT})$.

Optimistic Posteriori Sampling Agrawal and Jia [2017]: worst-case regret bound in $O(DS\sqrt{AT})$.

3.5.2 UCRL: Upper confidence bound based algorithm for RL

Expected reward $R(s, a)$ for all $s \in S, a \in A$, as well as $P(s, a)$ a distribution on S. At each step, we can use r_t to update an estimator of $R(s_t, a_t)$ and s_{t+1} for $P(s_t, a_t)$

UCRL algorithm Model-based approach: maintain estimates $\hat{P}, \hat{R}$, and occasionally solve the MDP $(S, A, \hat{P}, \hat{R}, s_1)$ to find a policy, run this policy for some time to get samples, update the estimates, and iterate. We proceed in epochs:

At every epoch k, use samples to compute an optimistic MDP $(S, A, \tilde{R}, \tilde{P}, s_1)$, solve it to find an optimal policy $\tilde{\pi}$. Then, execute $\tilde{\pi}$ in epoch k, and observe samples s_t, a_t, r_t, s_{t+1}. Then, go to next epoch if $n_k(s, a) \geq 2n_{k-1}(s, a)$ for some s, a.

[missing: more involved description of the algorithm]

Theorem 3.1. *For any communicating MDP with unknown diameter D, we have with high probability*

$$\text{Regret}(M, T) \leq \tilde{O}(DS\sqrt{AT}),$$

where $\tilde{O}$ hides logarithmic factors in S, A, T.

Proof sketch. By the communicating property, we have that w.h.p., the extended MDP in UCRL is communicating. For this extended extended MDP, the optimal average reward $\tilde{\lambda}(s)$ is independent of s.

The average regret in an epoch k is

$$\lambda^* - \frac{1}{T_k}\sum_{t\in[T_k]} r_{s_t,a_t} = (\lambda^* - \tilde{\lambda}) + (\tilde{\lambda} - \frac{1}{T_k}\sum_{t\in[T_k]} r_{s_t,a_t}).$$

The above first term is non-postive by construction of UCRL. For the second term, we follow the same policy but on a different MDP. The bounds are obtained using concentration of transition probability vector samples from the posterior.

Bellman equation Define the value functions

$$v_\gamma^M(s) = \mathbb{E}_M\left[\sum_{k\geq 1}\gamma^{k-1} r_k\right].$$

We have

$$v_\gamma^M(s) = R(s, M(s)) + \gamma\mathbb{E}_{s'\sim P^M(s,\cdot)}\left[v_\gamma^M(s')\right],$$

which also writes

$$v_\gamma^M = R^M + \gamma P^M \cdot v_\gamma^M$$

We can show that

$$\lambda^M(s) = \lim_{\gamma\to 1}(1-\gamma)v_\gamma^M(s)$$

Define the bias vector $h^M(s) := \mathbb{E}\left[\lim_{T\to\infty}\sum_{t=1}^T (r_t - \lambda^M(s_t))|s_1 = s\right]$. We actually have that

$$h^M(s_1) - h^M(s_2) = \lim_{\gamma\to 1}(v_\gamma^M(s_1) - v_\gamma^M(s_2)).$$

These two equations, together with the fixed point equation satisfied by $v_\gamma^M(s)$ give that for $\gamma \in (0,1)$,

$$(1-\gamma)v_\gamma^M(s) = R^M(s) + \gamma\sum_{s'} P^M(s,s')(v_\gamma^M(s') - \gamma v_\gamma^M(s)).$$

Then, sending $\gamma \longrightarrow 1$, one gets the Bellman equation:

$$\lambda^M(s) = R^M(s) + \sum_{s'} P^M(s,s')h^M(s') - h^M(s).$$

Bounding the difference In our context Bellman equation writes $\tilde{\lambda} - r_{s,\pi(s)} = \tilde{P}_{s,\pi(s)}\cdot\tilde{h} - \tilde{h}_s$, where $\tilde{h}$ is the bias vector of samples and satisfies $|\tilde{h}_i - \tilde{h}_j| \leq D$ for all $i, j \in S$. Thus

$$\begin{aligned}\tilde{\lambda} - \frac{1}{T_k} \sum_{t \in [T_k]} r_{s_t,a_t} &= \frac{1}{T_k} \sum_{t \in T_k} \left(\tilde{P}_{s_t,a_t} \cdot \tilde{h} - \tilde{h}_{s_t} \right) \\ &= \frac{1}{T_k} \sum_{t \in T_k} \left(\tilde{P}_{s_t,a_t} \cdot \tilde{h} - P_{s_t,a_t} \cdot \tilde{h} + P_{s_t,a_t} \cdot \tilde{h} - \tilde{h}_{s_t} \right).\end{aligned}$$

By martingale property, $P_{s_t,a_t} \cdot \tilde{h} - \tilde{h}_{s_t} = 0$. Then, w bound the deviation of posterior sample from the true model $(\tilde{P}_{s_t,a_t} - P_{s_t,a_t}) \cdot \tilde{h}$ (posteriori variance, sample error...). Since here $\tilde{h}$ is not fixed, we need a union bound, giving a bound in $\tilde{O}(D\sqrt{S}/\sqrt{N_{s,a}})$. □

3.5.3 Posterior sampling algorithm for MDPs

A more intuitive algorithm with different techniques for regret bound proofs.

Finite state, finite action S states, A actions.

Prior: Dirichlet $(\alpha_1, \alpha_2, ..., \alpha_i + 1, ..., \alpha_s)$ on $P_{s,a}$

After $n_{s,a} = \sum \alpha_i$ observations for a state-action pair s, a one computes the posterior $\hat{p}_{s,a}(i) = \frac{\alpha_i}{\sum_j \alpha_j} = \frac{\alpha_i}{n_{s,a}}$.

The variance is bounded by $\frac{1}{n_{s,a}}$: the more we have trials, the more the posterior concentrated around true probability.

Learning phase One maintains a Dirichlet posterior for $P_{s,a}$ for any (s, a). We start with an uninformative prior Dirichlet $(1, 1, \dots, 1)$.

Deciding phase We first sample $\tilde{P}_{s,a}$ for any (s, a). Then the optimal policy $\tilde{\pi}$ is computed for the MDP $(S, \varphi A, \tilde{P}, r, s_0)$

Our algorithm

- For any (s, a), generate multiple $\psi = \tilde{O}(S)$ independent samples from a Dirichlet posterior for $P_{s,a}$.
- Form extended sample MDP $(S, \psi A, \tilde{P}, r, s_0)$.
- Form optimal policy $\tilde{\pi}$ and use through the epoch.

Further initial exploration: For (s, a) with very small $N_{s,a} < \sqrt{\frac{TS}{A}}$ use simple optimistic sampling that provides extra exploration.

Regret bound analysis Assumption True MDP is communicating with diameter D

- For UCRL: with high probability, extended MDP is communicating
- For posterior sampling whp extended MDP is a communicating MDP with diameter at most $2D$.

We recall that a useful property of communicating MDP is that optimal asymptotic average reward does not depend on the initial state.

The averaged regret in an epoch k is

$$\lambda^* - \frac{1}{T_k}\sum_{t\in[T_k]} r_{s_t,a_t} = (\lambda^* - \tilde{\lambda}) + (\tilde{\lambda} - \frac{1}{T_k}\sum_{t\in[T_k]} r_{s_t,a_t}).$$

Two main results: First the optimism of transition matrix on a projection is sufficient $\geq \lambda^*$ if for every s, a, $\tilde{P}_{s,a}.h^* \geq P_{s,a}.h$ if a set of samples satisfy optimism on projection to unknown bias vector h^*.

Second For any fixed bounded vector h a sample satisfies above with probability $1/S$. There is no need to know h^*! But there is a need of $O(S\log(\frac{SA}{\rho}))$ samples whp.

3.6 Learning to manage inventory

It gives a general recipe for a loose class of problems.

Overview:

You start the inventory at time t, you observe inv_t, you gather new o_t and old o_{t-L} orders. Then you have to deal with the demand d_t after dealing a new on hand inventory $I_t = inv_t + o_{t-L}$. You then observe $y_t = mon(I_t, d_t)$.

You finally incur holding and lost sales cost $h(I_t, d_t)$...

Learning an MDP In each round $t = 1..T$:

- Observe inventory I_t, past $L-1$ orders $(o_{t-L+1}, ..., o_{t-1})$.
- Decide new order $o_t \in [0, U]$
- Observe sales $y_t = \min(d_t, I_t)$ where $d_t \sim F$.
- Incur cost $\bar{C}_t = h(I_t - y_t) + p(d_t - y_t)$
- Start new inventory $I_{t+1} = I_t - y_t + o_{t-L+1}$.

Holding unobserved lost sales The actual cost is $\bar{C}_t = h(I_t - y_t) + p(d_t - y_t)$ but d_t is unknown. We then use the surrogate $C_t = h(I_t - y_t) + p(-y_t)$.

Challenges and relaxation The past L orders are dependent, this implies combinatorial states dependance

Base stock policies We generate π^x: order the minimum amount required to bring new inventory position to base-stock level x.

Some properties: Let g^x be an asymptotic long-run average of a base stock policy π^x starting in state s. (the expectation of the limit of the averaged sum of the costs when time T goes to infinity.)

$$g^x(s) = \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} C_t \middle| s\right]$$

As the ratio of lost sales penalty to holding cost increase to infinity, the ratio of the cost of the best base stock policy to the optimal cost converges to 1.

Furthermore, if one assumes that demand distribution F is such that $F(0) > 0$ then $g^x(s)$ is independent of s and convex in x.

Analogy with convex bandit There are similarities between the inventory problem and convex bandits. In stochastic convex bandit, you want to minimise the regret between our observed costs and the optimal expected strategy while in inventory control, you minimised the expected cost wrt the optimal averaged asymptotic strategy.

Both cost function are convex.

To complete

4 Natural Language Processing

Note takers: Blanke, Daoud, Duchemin, Gourru, Jhuboo, Jourdan, Lauga, Mercklé, Sandberg, Terreau

Instructor: Franęis Yvon (LIMSI/CNRS)

[Video on the IBM project debater] Today, machines are capable of amazing, human-level performances.

4.1 Language: a hard nut to crack

Natural Language Processing (NLP) was studied for three main reasons:

- language processing as computation,
- computational psycholinguistics,
- automated processing tools and applications.

A classical approach to automatic speech recognition consisted in the following **pipeline model**.

- Lexical decoding: from a continuous-time audio signal to a discrete sequence of phonetic symbols.
- Orthographic decoding: create words and a sentence out of the phonetic symbols.
- Text normalisation: identify categories of words in the sentence.
- Structure identification: find the dependencies between the words.

The way back is even harder.

An example of tricky sentence "Le cousin de Paul se piquait de bien connaître la ville". It is tricky for several reasons. "Connaître" doesn't have a subject. "Cousin" can mean "cousin" but also a tipula. "Se piquer" could be confused with an insect biting. This is called word sense disambiguation.

More than processing one sequence, it is even harder to handle a span of text of several sentences with coreferences between them. Typically, different mentions of one single entity (*e.g.* one person) make an automatic processing very challenging.

Nowadays, there are a number of NLP tasks that go beyond isolated sentences. Classifying sentences, for instance by tense, mood, polarity etc. Typically, classifying tweets. Finding the structure of a text.

The pipeline model mentioned above often struggles because of arbitrarily long, multi-layered dependencies across the pipeline.

Another typical case of reason of failure is the ambiguity of some words. In politics for example, the chair (organizer) of a conference would sometimes be understood as a chair as the piece of furniture. In French, just think of the word "et" which can easily be confused with "et" or even "hait".

Pipeline model does not work. "It's like building a compiler, but you have only part of the syntax."

- Errors accumulate down the pipe.
- Early decisions require deep analysis.
- Ambiguity is a feature, not a defect (puns!).
- Segmentation ambiguities
  - *gardes* plural or second person
- Lexical ambiguities
- Syntactic ambiguities
  - "N. H. Defends Laconia Law Barring Female Nudity In Supreme Court Ruling"
- Semantic Ambiguities
- Pragmatic Ambiguities
  - Understand that "I'm cold" means "Close the window"

Language is always evolving.

- Phonetic changes and reconfigurations
- New spellings and grammatical constructs
- Lexicalization of new derivatives and compounds

- New senses appearing
-

4.2 The great paradigm shift : towards statistical NLP

With more resources and a more fine grained description of languages, we could get to this nice pipeline scenario. Around 92-93, statistics have progressively been incorporated into NLP. This transition has officially started with a publication in computation linguistic in the context of special issues on corpora based approaches.

Switch from grammar to corpora patterns : the hypothesis is to find ways to process large facets of languages. 2011, Norvig

The first ingredient to moving to statistical language models. Arguments for graduality in language: grammatical rules and judgement can be gradual : for instance house is a noun but home is a "better" noun than house, that has a larger combinatorial power. Similarly, "grièvement" only applies to specific contexts, while "gravement" can have much more applications and would be preferred as both mean the same. Finally, Human brain appears to be sensible to frequency : we recognize frequent word quicker. These are arguments to go toward statistical treatment of natural language.

A second ingredient is the collection of a large corpus of relevant data. Linguistic Data Consortium - LDC () : catalog of corpora, resources for annotation, rare languages, see LREC.

The third ingredient is the development of NLP challenges by funding agencies. They focused on having strong methodological construction of tasks.

- describe the task exactly
- what is given to participants (computational resources)
- what is the metric
- distribute test data for final evaluation

(Repository to track the progress in Natural Language Processing (NLP)). People tend to participate to these challenges for : access to data, and access funding. These have been highly influential to move to statistical methods that were, most of the time, the most accurate approaches.

4.3 Discovery of statistical method : the effectiveness of simple models

The simplicity of those models comes from the fact that they do not need to know any of the rules of a language and simply works from statistic measure (for example how likely some words will be written close together). In the following, we describe three important applications in NLP of this statistical viewpoint.

Speech recognition : a recurrent problem is how to decide the correct sentence to write for a given recorded sample, e.g "danser, dansés, dansé, dansée". How to compare sentences ?

Language models, simple yet effective (n-grams), with L the length of a sentence, V the vocabulary space.

$$P(w_1, ., w_L) = \prod_i^L p(w_i|w_1...w_{i-1}) = \prod_i^L p(w_i|w_{i-n+1}...w_{i-1}) \tag{4.1}$$

This technique allows to process omre than words such as letters, speech

Information Retrieval: Bag-of-words
→ Main idea: *"Turn a document into a vector."*
Each document is embedded as a vector $d \in \mathbb{R}^{|\mathcal{V}|}$ with $d^\top = (x_1, \dots, x_{|\mathcal{V}})$. A typical choice for x_i is

- $x_i = \frac{N(w_i \in d)}{l_d}$ where $N(w_i \in d)$ is the number of times the word w_i appears in the document d and l_d is the number of words in the document d.
- $x_i = TF - IDF(w_i)$ (Term Frequency(TF) — Inverse Dense Frequency(IDF)).

This embedding method allows to compare two documents d and d' using several measures as scalar product, cosine sim, standard distances.

Computational lexicography
→ Main idea: *"You shall know a word by the company it keeps."*
We compute semantic relationship from distributional observations: shared contexts imply semantic relatedness.

Considering a fixed vocabulary, $\mathcal{V} = \{v_1, \dots, v_{|\mathcal{V}|}\}$. For any word w, $r(w) \in \mathbb{R}^{|\mathcal{V}|}$ is the vector where the i-th entry counts the number of time the word v_i is a neighbor of the word w in the corpus. Then, the distance between two words w and w' is given by

$$\text{dist}(w, w') \propto r(w)^\top r(w').$$

4.4 From empirical methods to machine learning techniques

Supervised classification Resolving ambiguities by building trees. This can be turned to a simpler problem : finding dependencies, which are binary decision that can be solved using ML methods. Difficult step : find the good features to describe the data and the problem (context

Bag of Words, position in the sentence, sentence type). Other examples that can also be solved using classification tools are:

- word sense disambiguation,
- sentence segmentation
- co-reference resolution
- sentiment analysis
- ...

Results in from 1993-2010 can be summed up as: find a problem (word sense ambiguity,..), formulate it as a classification problem and use ML tools.

More ML related topics also emerged, such as high dimension, metrics, high number of classes that can even be organized in hierarchies.

ML was also successful in more applications cases than simple classification, e.g. parsing trees. These can be learnt step by step. Most exactly, what is learned is the sequence of actions needed (analogous to robotic movements). Action are not observed, some different sets of actions can lead to the same parsing tree. Dependency parsing build acyclic set of arcs between words.

No crossing arcs = projectivity(= easier to solve. Non-projectivity is rare in French and english. This allows some fast algorithms: greedy left-right decoding.

Transition-based projective dependency parsing: guaranteed to have an acyclic graph.

Remark: Punctuation is treated as words. But there are markers for the start and the end of a sentence, so that we know when words are usually used at the end (like punctuation). The main task of modeling structure, syntax of the language is to define a way do decide if a word or a sentence is better than an other. If one notes A and B two sentences, one can introduce the equivalence between A being better than B and a probability $P(A)$ being higher than a probability $P(B)$. This probability $P(sentence)$ is a language model. Such probability can easily be derived from a simple Markov assumption to predict the likelihood of one word based on the preceding words. This Markov assumption is very naïve but is extremely efficient computationally. One call this type of models n-grams and for a sequence of words $(w_1, ..., w_L)$ we define :

$$P\left(w_1, ..., w_L\right) = \prod_{i=1}^{L} P\left(w_i | w_1, \dots, w_{i-1}\right)$$

New architectures(transformers) are trained to learn which words matter in the history. In the past the importance was fixed. Feed-forward: fixed number of words in history. RNN: older words are gradually forgotten.

4.5 Transformers and self-attention

Compute with heads Compute linear weights:

$$\tilde{D} = \text{softmax}(D/\sqrt{d}) \in [0,1]^T \times [0,1]^T. \tag{4.2}$$

4.6 Evaluating language models

4.6.1 Large Language Models are *very* powerful

Originally used as scoring model for disambiguation tasks.

Will now cover use as text generators. Can use language models for any natural language tasks (Radford et al 2019).

Tasks: Give your model a prompt, and generate next word. Probability over possible next words. Can apply this to lots of different domains, such as translation, or even arithmetic. In the latter case they even are often correct.

The language models are evaluated with a measure called perplexity.

$$\text{PPL}(M) = 2^{\frac{1}{T}} \log_2 P(w[1:T]|M) \tag{4.3}$$

Cross entropy between source and model.

Before NNs language models had fairly bad perplexity (120 nats), now we reach around 6 times lower for models trained on English texts.

Evaluation with linguistic probes. How to evaluate if we learn long-range structural dependencies. Ex: Subject verb agreement. Subject must agree in number with the object, but they can bee far apart in the text. "The keys to the cabinet (are|is) on the table."

Linzen et al, (2016), traine an LM-RNN to predict the verb number. Performance good (1% error rate). Drops slowly with subject-verb distance. Drops slowly with intervening distractors (eg singular words between subject and verb). If instead train a NN to predict next word we get a 10-fold loss in performance. In complex cases, more direct form of training signal is needed to learn the correct structure.

4.6.2 Algorithms for text generation

Greedy Search At each step, the most likely word given the past is chosen.

Ancestral sampling

$$w_0 =< s > \tag{4.4}$$
$$w_t \sim P(w|w(t' \leq t)) \tag{4.5}$$

Nucleus Sampling

Language model (de)generation In practice, these text generation algorithms end up generating loops, even though they are syntaxically consistent.

High probability sentences do not resemble human productions.

- Too many repetitions.
- High-frequency tokens are over-represented, and low-frequency ones underrepresented.
- Lack of diversity.
- Lack of global consistency.
- Poorly calibrated posterior distribution.

Action takes place in very high-dimensional space. Predict next step from current vector in this space. Easy to go to "nonsensical" parts of the space. Easy to get caught in loops.

Beam search [with histogram pruning] More improved search algorithms (Wiher et al, 2022).

Better learning losses. - Use label smoothing.

4.6.3 Evaluating LMs with distributional properties

Evaluating Zero-shot/ few-shot behavior.

- Zero-shot learning, No demonstrations. "translate English to French: cheese -> ?"
- One-shot learning, one demonstration,...
-

Current challenges for language modeling.

- Text generation is still difficult.
- Improve efficiency and scalability.
- How to update models as language changes.
- How to avoid models learning hate-speech, and how to remove e.g. private information without having to retrain model, etc. (Stochastic Parrots)

4.7 Transfer learning

4.7.1 Multi-task learning and pretraining

Learning representations for NLP in an unsupervised way (Collobert 2011). Instead of having one benchmark for each task, they say we want to have one system for all tasks, and to learn it in an unsupervised manner.

In 2018 people started to implement this program at scale. Recipe:

1. take huge corpus to train embeddings (unsupervised).
2. Use this representation as features for supervised training on domain specific task.

Popular models include ELMO (Peters et al, 2018) and BERT (Devlin et al 2019).

Elmo is a network made by several stack of bidirectional RNNs. Pass sentence through RNNs back and forth, then passed to next layer. All layers are combined to yield the final representation.

Bert is a transformer, but it is non-causal. It can see the full sentence, and is trained by masking some of the words. In the last layer the goal is to predict the masked words.

BERT, and similar pre-trained encoders, typically give significantly improved performances. A lot of encoders these days are in the form of encoder-decoder pairs, using next word prediction, deshuffling, denoising, or similar techniques to avoid the need for labelling.

Benefits of LM pretraining: - Leverage large corpuses of text in an almost unsupervised way. - Allows for knowledge transfer between domains.

4.8 Multilingual NLP

4.8.1 Introduction

Diversity of languages around the world (see https://www.ethnologue.com/guides). These are divided in language families, which are not equality distributed around the world. The top 25 languages only covers half of the world population. Countries with only one language are the exception, bilingualism (or more) is the norm. Many languages are endangered due to their lack of use in the population from some economics point of view or else. New languages have also been created. This diversity of languages is particularly surprising given that languages have the same origin and humans have the same brain structure. Nevertheless there is a wide variety of linguistic systems.

NLPers should care for several reasons (https://ruder.io/nlp-beyond-english/) such as political/societal, economic, linguistic, Machine Learning, cultural/historical, cognitive. Motivation for using NLP in multilingual setting : usual publications don't even give the language that are studied, as English turned to be the standard ML language.

The typical methods of multilingual NLP are machine translation, multilingual models (mGPT, the multilingual version of GPT, mBERT) and cross-lingual representations and transfer. The available resources are parallel and comparable corpora (https://opus.nlpl.eu, wikipedia), bilingual dictionaries (https://panlex.org), comparative/typological language documentation (https://wals.info).

Main challenges Multilingual NLP suffers from a large resource unbalance (Joshi et al., 2020, https://arxiv.org/abs/2004.09095). Languages can be clustered into classes having different scales of available (labeled or not) data. While those resources are high for seven languages (0.27%, spoken by 2.5 billions), 2191 languages (88.38%, spoken by 1.2 billions) have no existing resources, such as annotated data for supervised settings. Therefore those languages can't benefit from recent technologies using NLP, e.g. voice command (phone, car, ...).

Nowadays some informal language sentences can mix two languages, e.g. bilingual speaker. New interesting problems arise such as language contact and code-switching. This lead to new tasks: language identification, language transcription and analysis, language translation, CS generation. See for example (Sitaram et al., 2019, https://arxiv.org/abs/1904.00784).

For moderation problem, hateful speeches are sometimes not recovered for low resources languages.

4.8.2 ML models for Machine Translation (ML)

An attempt to handle multilingual data consists in using machine translation.

The first approach used vanilla RNN (Recurrent Neural Network). Encoder decoder systems (seq2seq), go through a first phase of sentence encoding, then recursively generate the translated sentence (the target sentence). The main issue is that all the information of the source sentence need to be encoded in a (memory less) hidden vector. As this is not enough to store all the necessary information, this nice and simple approach fails.

To circumvent this problem, attention mechanisms were proposed (Bahdanau https://arxiv.org/abs/1409.0473, Luong https://arxiv.org/abs/1508.04025). The hidden representation is now a linear combination of the latent representations of the source sentence words. The network is modifying the representation of the whole sentence representation for the current word generation in the target sentence. Additionally, the attention matrix provides, for each generated word in the target language, the relative importance of each word in the source sentence.

This further led to the Transformer, that was initially proposed as a seq2seq model for language translation. The encoder is used for language modeling tasks. In a seq2seq setting, the decoder is using cross attention, i.e. computing attention between target and source sentences, which is not done in the Transformer encoder-only architecture (such as BERT). One main advantage of the multi head attention is the possibility to compute in parallel. See also Popel et al. (2020, https://www.nature.com/articles/s41467-020-18073-9.pdf).

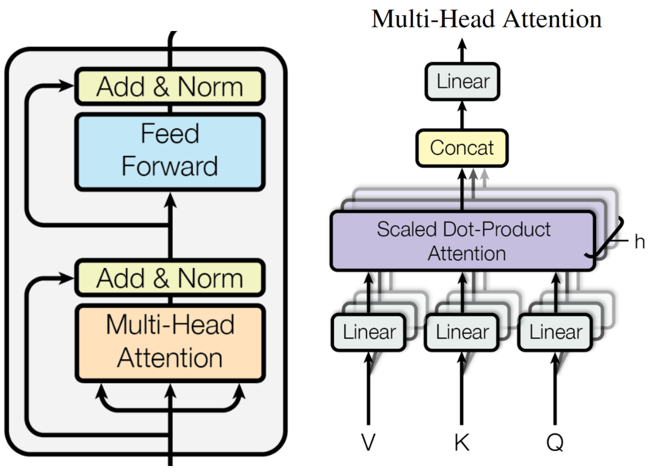

Figure 4.1: Bloc Transformer and multihead attention https://arxiv.org/abs/1706.03762

Multilingual models The usual approach took pairs of language. Now multilingual models translate from various languages to various languages with a single model (Firat et al., 2016, https://aclanthology.org/N16-1101.pdf. Johnson et al., 2017 https://arxiv.org/abs/1611.04558). This needs to build some joint representations of word pieces. A lot of studies are conducted on multilingual representations, i.e. words in several languages represented in the same low dimension continuous space. This assume that there is a stable notion of word across languages. This can be done using a Transformer (Conneau and Lample 2018, https://arxiv.org/abs/1901.07291)

Some crucial properties of Neural MT: segmentation in sentences and words, spelling and grammatical correction/normalization, grammatical parsing, sentence simplification.

Even bad MT is more useful than you think. MT translates artificial training data (text+labels) into other languages.

Universality of languages: X-lingual transfer learning (Yarowsky et al., 2001, https://aclanthology.org/N01-1026.pdf). The four main steps are:

- Automatic word alignment of parallel sentences
- PoS tag source data
- Project tags via alignment links
- Use of a PoS tagger with projected data

To obtain multilingual representations one should compute embeddings such that mutual translations nearest neighbours

- bilingual skip-gram
- X-lingual word space alignment with bilingual dictionary

- multilingual sentence representation via multilingual translation
- joint encoding/decoding with round-trip-translation

XLM (Lample and Conneau, 2019, https://arxiv.org/abs/1901.07291) learns multilingual contextual embeddings.

Conclusion Toward deep language understanding ? The language models are currently scaling to enormous datasets, thanks to more resources in term of materials, money and working force, that are dedicated to the field. A lot of what is done is based on many heuristics : it requires to go toward better optimizations for these huge Language models. Additionally, these languages do not incorporate knowledge. Finally, evaluation system are not properly built, and might prevent from getting the limitations of existing approaches. This is particularly difficult with text : how to evaluate if a sentence is “good” ?

5 Privacy in Machine Learning

Note takers: Ahmadipour, El Ahmad, Jose, Lachi, Lalanne, Oukfir, Nesterenko, Ogier, Siviero, Valla

Instructor: Rachel Cummings (Columbia University)

5.1 Introduction

Privacy considerations arise as soon data is collected on individuals, on group on individuals, on moral personas, $\ldots$. More specifically, we look at the setup where one processes data D through a mechanism $\mathcal{M}$ which can be anything from data publication, basic statistics computation, decision rule learning, complex machine learning tasks, $\ldots$, and wants the result $\mathcal{M}(D)$ to be made public. The natural question on a privacy standpoint is whether the mechanism $\mathcal{M}$ can be "reverted" in order to learn sensitive information from D. For instance, if $\mathcal{M}$ is the identity function, the publication of $\mathcal{M}(D)$ leaks full information about D and even though the notion of privacy is not rigorously defined yet, we can intuitively qualify such mechanism as "non-private".

This manuscript is a transcription of Prof. Rachel Cummings' lecture titled *Privacy in Machine Learning* that was given at the 2022 Spring School of Theoretical Computer Science at the CIRM, Marseille, France. Any error in this document may be due to its transcription and cannot be imputed to Prof. Cummings.

The lecture organizes as follows:

5.2 Defining privacy - Lecture 1

Even though the notion of privacy might seem natural at first, it is important to give it a good definition. We will start by trying to answer the question *What is privacy ?*

Attempt 1. *Privacy is about protecting identities.* This definition is natural. Something is private if it doesn't allow identifying you. As a result, it might be natural to consider that an algorithm is private if and only if it doesn't leak personally identifiable information (PPI). In practice, it is the main definition of privacy on a legal point of view. For instance, the French RGPD regulation instances (CNIL) consider that a mechanism is private when it makes a *sufficient effort* in hiding the identities. However, the more we look into it, the less convincing this definition becomes. First because it is extremely subjective, but mainly because it only shifts the problem. Indeed, what could be considered PPI or not? For instance, the last names and first names of people from a database seem to be natural PPI's. But what about their sum? Their encoding on a different alphabet? The application of any function on them? What about the correlation with other information such as the zip code, the income or the number of children? As a result, this definition has shown many failures in the past. For instance, research has shown that the search history of people can fully identify them, even with anything considered PPI's at the time removed (https://www.nytimes.com/2006/08/09/technology/09aol.html). On the other hand, it has also been shown that removing the PPI's can block inference and learning (Dwork et al.) and can only result in noise. As a result, this definition is better than nothing, but it is far from being future-proof, it both isn't really *private* while still partially blocking learning, and it requires a lot of legal effort in order to classify what is identifiable.

Attempt 2. *Privacy is about protecting people's freedoms from harm.* This definition is much stronger than the previous one. However, by the absolute aspect of this promise, it forces $\mathcal{M}(D)$ being independent of D. For instance, if researchers were to find correlations between smoking and lung cancer while not being able to learn if their patient smoked or not (in order to protect them from loosing their insurance), it would be a hard task. This definition of privacy thus has the drawback of completely blocking inference and learning.

Attempt 3. *Privacy is when almost no more information can be obtained with an analysis on the same dataset without a person's data.* This definition of privacy is interesting. Indeed, one can deduce the private information on an individual of D from $\mathcal{M}(D)$ if this individual has a huge impact on the result, i.e. when the result would have been significantly different without its information. As a result, privacy is obtained when $\mathcal{M}(D)$ is relatively invariant up to the addition or removal of any element of D. This definition of privacy will be adopted and rigorously defined through the concept of *differential privacy* in the rest of this lecture because it is the most future-proof and usable (even if it is still not clear for now) definition of privacy that research has come up with up to this day.

5.3 Differential Privacy - Lecture 1

The privacy of the mechanism $\mathcal{M}$ is achieved through randomization of its output. Formally, for $\varepsilon, \delta \geq 0$,

Definition. [DMNS '06] An algorithm $\mathcal{M} : T^n \to R$ is (ε, δ)-differentially private if $\forall$ neighboring $D, D' \in T^n$ and $\forall S \subseteq R$,

$$P[\mathcal{M}(D) \in S] \leq e^{\varepsilon} P\left[\mathcal{M}\left(D'\right) \in S\right] + \delta$$

where the randomness is taken on the coin tosses of $\mathcal{M}$.

Note that this definition bounds the "max amount" that one person's data can change the output of a computation. Furthermore, it is a worst case over all pairs of neighboring datasets. In particular,

- it doesn't matter what everyone else's data are,
- it doesn't matter what data you have,
- it doesn't depend on the future usage of $\mathcal{M}(D)$,
- if your data has huge influence, it will be hard to distinguish from your neighbors.

Furthermore, differential privacy does not block learning "DP addresses the paradox of learning nothing about an individual while learning useful information about a population. It is a definition, not an algorithm."- The Algorithmic Foundations of Differential Privacy, Dwork and Roth.

5.3.1 The role of the privacy parameters

This definition of privacy relies on two *privacy parameters*, ε and δ. They both impact how private the resulting mechanism is, but they do not play a symmetric role.

The role of ε. If a mechanism is (ε, δ)-DP, it is also (ε', δ)-DP if $\varepsilon' > \varepsilon$. As a result, the smaller ε, the stronger the constraint on privacy. The two following limit behaviors arise:

- $\varepsilon = 0$: Perfect privacy, where the result cannot depend at all on the data. As a result, no learning is possible.
- $\varepsilon = +\infty$: No privacy since the constraint vanishes. Privacy is no longer implied by the definition.

We want to be somewhere in the middle and the "correct "choice of ε is an open question depending on the sensitivity of the data.

The role of δ. Similarly, we can observe that the smaller δ, the stronger the privacy guarantees. δ differs from ε because:

- It gives a small additive slack in the privacy guarantee.

- It allows for a family of output distributions that are not all absolutely continuous with respect to each other. Imagine D, D' are neighboring databases and say $P[M(D) \in S] > 0$ and $P\left[M\left(D'\right) \in S\right] = 0$. Without δ :

$$0 < P[M(D) \in S] \leq e^{\varepsilon} P\left[M\left(D'\right) \in S\right] \leq e^{\varepsilon} \cdot 0 = 0$$

- Even with uniform support, it allows for an easier mechanism design.

In order to tune δ, we can fall back on the following observations and interpretations of this parameter:

- δ may be viewed as the probability under which the output mechanism does not respect the ε-DP guarantee.
- Hence, δ may be viewed as a *relaxation* term.
- If $\delta = 1$ then we're back to no privacy, even for $\varepsilon = 0$:

$$P[M(D) \in S] \leq e^{0} P\left[M\left(D'\right) \in S\right] + 1$$

- We have to take $\delta \ll \frac{1}{n}$. Indeed, when $\varepsilon = 0$ (which should give full privacy when $\delta = 0$), one can easily check that the mechanism that picks a random person from the database and output their data is $(0, 1/n)$-DP.

Remark: One might think that the definition of differential privacy is arbitrary, and it is. However, it is becoming increasingly adopted because this is the best that has been proposed to this date. Indeed, it ensures strong privacy guarantees while allowing for a nice algebra of private mechanisms (as we will see later). As a consequence, it is both conceptually powerful and handy, in a way that wasn't matched by previous definitions (such as k-anonymity).

Remark: The randomization of the output of the mechanism is at the core of this definition. Besides, one can easily check that trying to obtain privacy with a mechanism that is pointwise almost surely constant under $(\varepsilon, 0)$-DP results in a mechanism that is constant on all databases. Hence, one must be willing to pay a pointwise variance in order to obtain privacy.

5.3.2 Algebra of private mechanisms:

Private mechanisms come with three handy properties of post-processing, composition and group privacy that make them usable in practice.

Post-processing DP is immune to post-processing: If $M(D)$ is (ε, δ)-differentially private and f is any function (possibly stochastic), then $f(M(D))$ is (ε, δ)-differentially private. To put it simply, it is impossible to compute a function of the output of the private algorithm and make it "less" private. “No adversary (function f) can break the privacy guarantee ”

Composition DP is robust under composition: If $M_1, \ldots, M_k$ are (ε, δ)-differentially private, then: $M(D) \equiv (M_1(D), \ldots, M_k(D))$ is $(k\varepsilon, k\delta)$-differentially private.

If multiple analyses are performed on the same data, as long as each one satisfies DP, all the information released taken together will still satisfy DP (albeit with a degradation in the parameters) This result quantifies the common heuristic: Privacy degrades gracefully as more computations are performed on the same dataset. The linear scaling in both ε and δ can be further improved via advanced composition: If $M_1, \ldots, M_k$ are (ε, δ)-differentially private and adaptively chosen, then: $M(D) \equiv (M_1(D), \ldots, M_k(D))$ is $(\varepsilon', k\delta + \delta')$-differentially private for

$$\varepsilon' = \varepsilon\sqrt{2k \log \frac{1}{\delta'}} + k\varepsilon\left(e^{\varepsilon} - 1\right) = \theta(\sqrt{k}\varepsilon)$$

Composition allows composing *simple* private procedures in order to obtain *complex* private algorithms.

Group Privacy Privacy guarantee depends on the group size: If two datasets D, D' differ in k entries and M is (ε, δ)-differentially private, then for all outputs S :

$$\mathbb{P}[M(D) \in S] \leq e^{k\varepsilon}\mathbb{P}\left[M\left(D'\right) \in S\right] + ke^{\varepsilon(k-1)}\delta.$$

In other words, DP guarantees for *individuals* generalizes to DP guarantees for *communities.*

5.3.3 Neighboring databases

Note that for now, we did not properly define the notion of neighboring databases. Usually, we say that two databases are neighbors iff their content differs on at most one person's data. This informal definition can take multiple forms depending on the structure of the database.

- If the databases $\boldsymbol{x}$ and $\boldsymbol{y}$ are order-sensitive and of fixed size n, we usually say that x and y are neighbors when $\|\boldsymbol{x} - \boldsymbol{y}\|_0 \leq 1$. Databases are then compared according to their order sensitive Hamming distance.
- If the databases $\boldsymbol{x}$ and $\boldsymbol{y}$ are order-insensitive and of fixed size n, we usually say that x and y are neighbors when $\inf_\sigma \|\boldsymbol{x} - \sigma(\boldsymbol{y})\|_0 \leq 1$ where σ is any permutation that permutes the entries of $\boldsymbol{y}$. Not that if those databases are built on a countable set, this definition is equivalent to $\|h(\boldsymbol{x}) - h(\boldsymbol{y})\|_1 \leq 2$ where the function h transforms a database into its histogram (i.e. the vector counting the occurrences of the elements). Databases are then compared according to their order insensitive Hamming distance.
- If the databases $\boldsymbol{x}$ and $\boldsymbol{y}$ are order-insensitive and of possibly arbitrary sizes $n_{\boldsymbol{x}}$ and $n_{\boldsymbol{y}}$, we usually say that x and y are neighbors when $\|h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{x}) - h_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{y})\|_1 \leq 1$ where $h_{\boldsymbol{x},\boldsymbol{y}}$ refers to the histogram function that builds on the supports of $\boldsymbol{x}$ and $\boldsymbol{y}$ (which is countable). Databases are then compared according to their size insensitive Hamming distance.

Independently of the definition, we write $\boldsymbol{x} \sim \boldsymbol{y}$ when $\boldsymbol{x}$ and $\boldsymbol{y}$ are neighbors. All those definitions are not equivalent, but it is often clear which one to use depending on the setup.

Most of the results do not depend on the definition of neighboring databases, but when they do, it will be specified.

5.4 Private Mechanism Design - Lecture 1

This section presents simple building blocks for designing private mechanisms.

5.4.1 Laplace Mechanism

Given a unction f defined on a set of databases and valued in a real vector space, how can one mimic the behavior of f with a private mechanism? The Laplace mechanism gives a simple answer to this question by adding Laplace noise to the expected result scaled to the *sensitivity* of f.

Definition. The sensitivity of a function f is defined as

$$\Delta f = \max_{\boldsymbol{x} \sim \boldsymbol{y}} \|f(\boldsymbol{x}) - f(\boldsymbol{y})\|_1 .$$

Examples.

- If f counts the number of people with blue eyes, $\Delta f = 1$.
- If f is a histogram function built on a finite quantization of the data space, $\Delta f = 1$ with size-insensitive neighboring definition and $\Delta f = 2$ otherwise.
- If f is an averaging function, $\Delta f = \infty$ generally. however, if the data points live in set of l_1 diameter D, $\Delta f = D/n$ with the size-sensitive neighboring definitions and $\Delta f = D$ with the size-insensitive neighboring definition.

Laplace Mechanism - Definition The Laplace mechanism for f with privacy parameter ε is defined as

$$\mathcal{M}_L(\boldsymbol{x}, f, \varepsilon) = f(\boldsymbol{x}) + [\mathrm{Lap}(0, \Delta f/\varepsilon)]$$

where $[\mathrm{Lap}(0, \Delta f/\varepsilon)]$ refers to a vector (of size the output dimension) of i.i.d. random variables with centered Laplace distributions of standard derivation $\Delta f/\varepsilon$.

The structure of the noise allows for *pure* differential privacy (i.e. $\delta = 0$).

Theorem 5.1: Laplace Mechanism - Privacy

$\mathcal{M}_L(\cdot, f, \varepsilon)$ is $(\varepsilon, 0)$-differentially private.

Proof. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two neighboring databases. $\mathcal{M}_L(\boldsymbol{x}, f, \varepsilon)$ and $\mathcal{M}_L(\boldsymbol{y}, f, \varepsilon)$ have distributions that are absolutely continuous with respect to Lebesgue measure that are strictly positive almost everywhere. We may compare the ratio of these densities.

$$\begin{aligned}
\frac{\mathbb{P}[\mathcal{M}_L(\boldsymbol{x}, f, \varepsilon) = z]}{\mathbb{P}[\mathcal{M}_L(\boldsymbol{y}, f, \varepsilon) = z]} &= \frac{\mathbb{P}[[\mathrm{Lap}(0, \Delta f/\varepsilon)] = z - f(\boldsymbol{x})]}{\mathbb{P}[[\mathrm{Lap}(0, \Delta f/\varepsilon)] = z - f(\boldsymbol{y})]} \\
&= \frac{\Pi_i \mathbb{P}[\mathrm{Lap}(0, \Delta f/\varepsilon) = z_i - f(\boldsymbol{x})_i]}{\Pi_i \mathbb{P}[\mathrm{Lap}(0, \Delta f/\varepsilon) = z_i - f(\boldsymbol{y})_i]} \\
&= \frac{\Pi_i \frac{\varepsilon}{2\Delta f} e^{-\frac{\varepsilon |f(\boldsymbol{x})_i - z_i|}{\Delta f}}}{\Pi_i \frac{\varepsilon}{2\Delta f} e^{-\frac{\varepsilon |f(\boldsymbol{y})_i - z_i|}{\Delta f}}} = \Pi_i e^{-\frac{\varepsilon(|f(\boldsymbol{x})_i - z_i| - |f(\boldsymbol{y})_i - z_i|)}{\Delta f}} \\
&\leq \Pi_i e^{-\frac{\varepsilon |f(\boldsymbol{x})_i - f(\boldsymbol{y})_i|}{\Delta f}} = e^{-\frac{\varepsilon \sum_i |f(\boldsymbol{x})_i - f(\boldsymbol{y})_i|}{\Delta f}} \\
&= e^{-\frac{\varepsilon \|f(\boldsymbol{x}) - f(\boldsymbol{y})\|_1}{\Delta f}} \leq e^{\varepsilon} \,.
\end{aligned}$$

So for any Borel set S,

$$\begin{aligned}
\mathbb{P}[\mathcal{M}_L(\boldsymbol{x}, f, \varepsilon) \in S] &= \int \mathbb{P}[\mathcal{M}_L(\boldsymbol{x}, f, \varepsilon) = z] dz \\
&\leq e^{\varepsilon} \int \mathbb{P}[\mathcal{M}_L(\boldsymbol{y}, f, \varepsilon) = z] dz = e^{\varepsilon} \mathbb{P}[\mathcal{M}_L(\boldsymbol{y}, f, \varepsilon) \in S] \,,
\end{aligned}$$

which concludes the proof. □

Furthermore, the tail bounds of the Laplace distribution give the following utility guarantee:

Theorem 5.2: Laplace Mechanism - Accuracy

$$\mathbb{P}\left[\|f(x) - y\|_1 \leq \log\left(\frac{d}{\beta}\right) \cdot \left(\frac{\Delta f}{\varepsilon}\right)\right] \geq 1 - \beta$$

where d is the output dimension.

This is our first example of a privacy-utility tradeoff. With the Laplace mechanism, the higher the privacy guarantees are, the more degraded the utility is. Also, we can notice that the higher the sensitivity, the lower the utility.

5.4.2 Exponential Mechanism

The Laplace mechanism works great when the output space is a real vector space and when the utility of the output can be measured with the l_1 norm. But what if the output space has a different structure (ex texts) or what if the utility does not depend directly on the l_1 norm? The exponential mechanism solves this problem by allowing mechanism design with an arbitrary utility function.

The exponential mechanism has to assign a numeric score to each possible output

Assign a specific numeric score to each possible output.

Quality of outcome measured by score function: $q : \mathbf{N}^{|X|} x \mathcal{R} \to \mathbf{R}$ where q(x,r) is a measure of how good outcome r would be on database x

Choice of q should depend on application

Reasonable quality score?

Smooth degradation of outputs.

Score function sensitivity

Definition:

The sensitivity of a score function $q : \mathbf{N}^{|X|} x \mathcal{R} \to \mathbf{R}$
$\Delta q = \max r \in \mathcal{R} \max x, y$ neighbors

Exponential Mechanism [MT07]

Definition: Given a quality score q:

Essentially we do a "biased sampling" with an exponential bias.

Example:

Most common eye color? X = brown, blue, green $x \in \mathbf{N}^{|X|}$ database of eye colors $\mathcal{R} = X$ $q(x, r)$ = # people in database x with eye color r $\Delta q = 1$ as each person can have at most one eye color

Theorem 5.3: MT'07

The exponential Mechanism M is ε differentially private

$$\frac{\mathbb{P}[\mathcal{M}_E(x, q, \varepsilon) = r]}{\mathbb{P}[\mathcal{M}_E(y, q, \varepsilon) = r]} \le e^{\varepsilon}$$

Proof.

$$\frac{\mathbb{P}[\mathcal{M}_E(x, q, \varepsilon) = r]}{\mathbb{P}[\mathcal{M}_E(y, q, \varepsilon) = r]} = (definitionofexponentialmech) = (lawofexponents, sameasproofinLaplacemech) = ...firs$$

The first term is similar to what we saw in the Laplace Mechanism, so suing the same techniques we can show that:

This means we can swap x and y at the above cost. So, for the second term,

□

Accuracy:

$$\mathbb{P}[q(x,r) - \max r^{'} \in \mathcal{R} q(x,r^{'} \leq \frac{2\Delta q \cdot ln(|\mathcal{R}|/\beta)}{\varepsilon}] \leq \beta$$

High probability to pick an outcome that is close to the best possible outcome.

Best possible means highest quality score

Close depedns on high probabiity guarantee.

Exponential Privacy Accuracy trade-off

5.5 DP and online/adaptive statistics

A ε-DP algorithm is more noisy, but does this hurt generalization? Training score is worse, but this can also prevent overfitting.

5.5.1 DP and generalization

Theorem 5.1. *An ε-DP algorithm cannot overfit by more than ε*

We want the learning with DP samples to be (almost) as good as with the underlying distribution (not compared to the ground truth).

Reminder (Group Privacy) If S, S' differ in k elements, then $k\varepsilon$-privacy

DP private learners generalize well Notions of generalization:

- DP generalization: "similar samples should have similar output." DP-guarantee are strong worst case guarantee
- Weaker notion: Robust Generalization "no adversary can use the output to find a hypothesis that overfits"
- Stronger notion: Perfect Generalization "output reveals nothing about the sample". (Does not compare against a sample changed by one, but against the true underlying distribution. Means you are perfectly generalizing from the sample)

Why don't we change DP def to include distribution? eg for some rare databases, provide weaker privacy. However rare events are precisely the ones we are trying to protect. This is not an issue here

5.5.2 DP and adaptive analysis

How to do data analysis in a robust way?

What can go wrong? To learn global truth, the agent sends multiple queries sequentially, adapting new queries depending on the previous answer. [DFHPRR15] This can cause overfitting.

Particularly a risk in fields where scientists share one dataset (*eg* astronomy, historical datasets in economy)

`AboveNoisyThreshold` for multiple threshold queries [DNPR '10]

This is a DP algorithm for detecting which queries in a stream have answer above a given threshold.

```
input: database X, query stream $\{f_1, \dots\}$ with sensitivity $\Delta$, privacy parameter $\varepsilon$ and
       threshold $T$.
$\hat{T} := T + \mathrm{Lap}(2\Delta/\varepsilon)$;
for each query $f_i$ do
    $Z_i \sim \mathrm{Lap}(4\Delta/\varepsilon)$ (we add noise twice!);
    if $f_i(x) + Z_i > \hat{T}$ then
        output Above and halt
    else
        output Below
    end
end
```

Remarks This Algorithm compares noisy answer against a noisy threshold (fixed in advance). It can be proven $(\varepsilon, 0)$ DP and satisfies a composition privacy for k queries with only $\varepsilon = \log k$, can answer exponentially many queries (by composition theorem)! (vs composition of queries gives k or $\sqrt{k}$). Finally, ANT halts once it finds a single above threshold query, we need another algorithm if we would like to find multiple above threshold queries.

`SparseVector` to do threshold queries and do something with the results above threshold

Combine ANT and Laplace mechanism to release the answers.

Applicable to many problems

`Reusable Holdout`

Randomly partition D in training D_t and holdout D_h

When training a model, we only test generalization when testing on holdout, but this is true if holdout is used only once! (it needs to be considered as a fresh sample). The idea here is to access holdout only through DP algorithm, then no overfitting on holdout.

```
Input: training set $S_t$, holdout set $S_h$, threshold $T$, tolerance $\tau$, budget $B$.
$\hat{T} := T + \mathrm{Lap}(4\tau)$
For each query $\varphi : X \to [0,1]$:
if $B < 1$ then
    output Below and halt
else
    if $|E_{S_h}(\varphi) - E_{S_t}(\varphi)| > \hat{T} + Lap(8\tau)$ then
        output $E_{S_h}(\varphi) + \mathrm{Lap}(2\tau)$
        $B = B - 1$
        $T = T + \mathrm{Lap}(4\tau)$
    else
        output $E_{S_t}(\varphi)$
    end
end
```

Same algo as SV with

- check if answer on holdout is close to answer on training set (*i.e.* above noisy threshold)

  if no release noisy answer on holdout

  otherwise just release answer on training
- As in SV, we have a privacy budget, counting if we can still access holdout

DP and accuracy are quantified (see slides).

Theorem 5.2. *Thresholdout is $B/(\tau n)$-DP. For all adaptively chosen queries $\{\varphi_1, \dots, \varphi_m\}$, for all i such that a_i isn't "bellow" the threshold, for all $t > 0$:*

$$\Pr[|a_i - \Pr(\varphi_i)| > T + (t+1)\tau] \leq 6\exp(-\tau^2/2) + \exp(-t/8).$$

5.5.3 DP and sequential hypothesis testing

Try to address the replication crisis, how to get meaningful p-values?

As usual, observe $x_1, \cdot, x_n$ and have null hypothesis H_0 and interesting alternative hypothesis H_1. p-value is likelihood of seeing the sample assuming the null (reject H_0 if p is small)

Usual threshold is $p < 0.05$, small but still means that there is 5% chance of this sample occurring under the null. In particular, when testing 20 hypotheses, we can expect around one false discovery.

Controlled by False Discovery Rate (FDR). FDR is a measure capturing rate of false rejection of H_0. We want a post processing to control in an offline (sequence of p-values is known in advance), or online manner.

Offline FDR control Just select the k smallest p-values.

Online FDR control framed as an investment problem (because lots of tools and framework for budget, reward) "online alpha-investing rule" then "generalized alpha-investing rule"

Level based on recent discovery (LORD) and SAFFRON add statefulness to estimate current proportion of true nulls.

SAFFRON

Keep a candidate set, estimate fraction of true null from the size of this set. When new value arrives, estimate value of investing in the hypothesis, and current wealth. Gives alpha-investing value α_t.

PAPRIKA

```
input: p-values $\{p_1, \dots\}$, multiplicative sensitivity parameter $\eta$, target FDR level, initial
       wealth, privacy parameters $(\varepsilon, \delta)$, expected number of rejection $c$
$\hat{Z} \sim \text{Lap}(2\eta c/\varepsilon)$;
count $\leftarrow 0$;
for each p-value $p_t$ do
    $\hat{Z}_t \sim \text{Lap}(4\eta c/\varepsilon)$;
    or candidacy $C_t \leftarrow 1(\log(p_t) < \text{Threshold}_t)$;
    Compute alpha investing rule $\alpha_t$;
    if $C_t = 1$, count $< c$, $\log(p_t) + Z_t < \log(a_t) + \hat{Z}$ then
        output $R_t = 1$;
        count $++$;
        resample $\hat{Z} \sim \text{Lap}(2\eta c/\varepsilon)$
    else
        output $R_t = 0$ (fail to reject)
    end
end
```

Remarks. combine SAFFRON investment estimation with SV

instead of comparing p and α, compare noisy versions of them

looks a lot like ANT but

Multiplicative sensitivity: $\eta(f) := \min\left\{\max \frac{f(x)}{f(y)}, \max\{f(x), f(y)\}\right\}$ (either small ratio, or both are very small anyway)

- keep a candidacy set. In ANT noisy in a symmetric way (fails both way as often), here we want false rejection to be rarer.
- here sensitivity is multiplicative and looking at $\log p$, because sensitivity of p-value can be very high.

Theorem 5.4: PAPRIKA is DP and accurate

PAPRIKA is (ε, δ)-DP **and controls FDR** to below an explicit threshold

Note that for this algorithm $\delta > 0$ (but tiny).

No theoretical guarantee with respect to the power of the method. In experiments, good power requires rather large ε values.

5.5.4 DP and Changepoint Detection

Goal: detect distribution of timeseries changes at $t^\star$

Assume we have offline DP method (reasonable)

How to do it online? DP detect that test statistic is above threshold in the sliding window, and run offline algo on this window

5.6 Online Optimization

summary:

- Private algo for maintaining partial sum
- Private Follow The Approximate Leader

Incoming stream, and we want to adapt the decision based on what was seen before

5.6.1 First idea

Given stream of bits $b_1, \dots, b_T$. At each time t output $\sum_{\tau=1}^{t} b_\tau$

Bad idea 1 At each time t, output $\sum_{\tau=1}^{t} b_\tau + \mathrm{Lap}(1/\varepsilon)$

Then by composition, $\varepsilon = \sqrt{T}\varepsilon' \log 1/\delta$

Accuracy loss $O(1/\varepsilon')$

Good accuracy *or* good privacy. Fix ε and choose $\varepsilon' = \varepsilon/\sqrt{T}$, then we have accuracy loss $O(\sqrt{T}/\varepsilon)$. However this can easily become large

Bad idea 2 Add noise to each b_i : $\hat{b}_i = b_i + \text{Lap}(1/\varepsilon)$

output $\sum_{i=1}^{\tau} \hat{b}_i = \sum_{i=1}^{\tau} b_i + \tau\text{Lap}(1/\varepsilon)$

1. Big noise infrequently 2. Small noise too often

We need a data structure to fix this

Better idea (but really a lie)

Break down into blocks (like a balanced binary tree). Then per sample, add Laplace noise for each block.

$\tau+$ $Lap(1/\varepsilon')$

$\tau/2+$ $Lap(1/\varepsilon')|$ $\tau/2+$

$Lap(1/\varepsilon')$

$\tau/4 + Lap(1/\varepsilon')|$ $\tau4 + Lap(1/\varepsilon')|$ $\tau/4 + Lap(1/\varepsilon')|$

$\tau/4 + Lap(1/\varepsilon')$

$\dots$

$1|2|$ $3|\dots|$

τ

Goals

- Any sum uses only $O(\log(T))$ noise terms
- Any noise term is used only $O(\log(T))$ times

NB: instead of bits b_i, we can think of vectors z_i with $\|z_i\| \leq \Delta$. Then use noise $\text{Lap}(\Delta/\varepsilon')$. And replace with $\sum^t z_i$.

Tree Based Aggregation Protocol (TBAP) [Chanet et al. 2010, Dwork et al. 2010]

```
input  : $z_1, \dots, z_\tau \in \mathbb{R}^d$, $\Delta$ $l_2$-bound on all $z_t$, $\varepsilon$
output : Sequence of noisy partial sums $v_1, \dots, v_\tau \in \mathbb{R}^d$
Initialize binary tree $A$ of size $2^{\lfloor \log_2 T \rfloor} - 1$ with leaves $z_1, \dots, z_\tau$;
for $t = 1, \dots, T$ do
    Accept $z_t$ from data stream;
    Let $P = \{z_t \to \cdots \to \text{root}\}$ be a path from $z_t$ to the root. ;
    *Tree update.*;
    Let $\Lambda$ be the first node in P that is a left-child in A. *We only add noise up to a point (the
      first left-child) then stop* ;
    Let $P_\Lambda = \{z_t \to \cdots \to \Lambda\}$ ;
    for *all nodes $\alpha$ in path $p$* do
        $\alpha \leftarrow \alpha + z_t$;
        if $\alpha \in P_\Lambda$ then $\alpha \leftarrow \alpha + \gamma$ where $\gamma \in \mathbb{R}$ sampled $\propto \exp \frac{-\|\gamma\|_2 \varepsilon}{\Delta \lfloor \log_2 T \rfloor}$;
    end
end
*Output partial sums*;
Initialize $v_t \in \mathbb{R}^d$ to be 0 ;
Let $b$ be a ( $\lfloor \log_2 T \rfloor + 1$)-bit binary representation of $t$. ;
for $i = 1, \dots, \lfloor \log_2 T \rfloor + 1$ do
    if $b_i = 1$ then
        always add something;
        if *$i$-th node in $P$ (denoted $P(i)$) is a left child* then
            $v_t \leftarrow v_t + P(i)$
        else
            $v_t \leftarrow v_t +$ left-sibling($P(i)$)
        end
    end
end
return $v_t$
```

NB: Laplace Mechanism $\mathbb{P}x \propto \exp \frac{-\|x\|\varepsilon}{\Delta}$, here $\mathbb{P}\gamma \propto \exp \frac{-\|\gamma\|\varepsilon}{\Delta floor \log_2 T}$

Private follow the Approximate Leader

```
input  : sequence of cost functions $f_1, \dots, f_\tau$, $H$, $L$, $C$, $\varepsilon$
Initialize $\hat{w}_i \in C$ arbitrarily, output $\hat{w}_i$ for $t = 1, \dots, T$ do
    Pass $\nabla f_t(\hat{w}_i), L, \varepsilon$ into TBAP and recieve current partial sum $\hat{v}_t$
    $\hat{w}_{t+1} = arg \min_{w \in C} < \hat{v}_t, w > + \frac{H}{2} \sum_{\tau=1}^{t} \|w - \hat{w}_\tau\|_2^2$
    Output $\hat{w}_{t+1}$
end
```

5.7 Private Deep Learning

The only thing to to is to adapt a DP gradient descent.

DP-SGD [Abadi et al 2016] (Deep learning with differential privacy, In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security).

```
input  : Dataset $X = (x_1, \dots, x_n)$, loss function $\mathcal{L}_\theta$, learning rate $\eta_t$, batch size $L$, noise
         multiplier $\sigma$, gradient norm bound $c$
Initialize $\theta_0$ randomly;
(Sample a batch) e.g. Poisson random subsample $L_t$ with pre-example prob $L/n$ ;
for each $x_i \in L_t$ do
    compute $g_t(x_i) = \nabla_{\theta t} \mathcal{L}_{\theta t}(x_i)$ ;
    $\bar{g}_t(x_i) = \frac{g_t(x_i)}{\max\{1, \|g_t(x_i)\|_2 / c\}}$ ;
    $g_t = \frac{1}{|L_t|} \sum_{x_i \in L_t} g_t(x_i) + N(0, \sigma^2 C^2 I)$ ;
    $\theta_{t+1} = \theta_t + \eta_t g_t$
end
output : $\theta_T$ and compute overall $(\varepsilon, \delta)$-DP bound via privacy accounting.
```

The question is the correct size of the noise to add. Difficult a priori because we don't know the variations of the gradient (possibly unbounded). We can clip the gradient to always lie in some range.

Note that there are no ε or δ in the algorithm. Could use composition, but here training for thousands of rounds so even a square-root bound is too large. We have special composition rules for learning with gaussian noise.

Definition 8 (Renyi DP [Mir 17]). *A mechanism M is (α, ε)-RDP if for all neighbours x, x'*

$$RDP(\alpha) := D_\alpha(M(x) \| M(x')) \leq \varepsilon$$

where $D_\alpha(P\|Q) = \frac{1}{\alpha-1} \log \left(\mathbb{E}_{x\sim x} \left[\left(\frac{P(x)}{Q(x)} \right)^\alpha \right] \right)$

Privacy accounting of DP-SGD via RDP

1. compute subsample RDP parameters for one step $RDP_{t=1}(\alpha)$
2. RDP composition:

   Proposition 3. *If M_1, M_2 respectively are $(\alpha, \varepsilon_1), (\alpha, \varepsilon_2)$-RDP for $\alpha \geq 1$, then the composition is $(\alpha, \varepsilon_1 + \varepsilon_2)$-RDP.*

3. convert to (ε, δ)-DP

   Proposition 4. *If M is (α, ε)-RDP $\forall \alpha \geq 1$, then M is $\left(\varepsilon(\alpha) + \frac{\log(1/\delta)}{\alpha-1}, \delta \right)$-DP $\forall \delta > 0$.*

   (ε depends on *alpha*).

5.8 Misc.

Who adds noise? Two models:

- *Trusted Curator model:* requires trusted party collects and sees data, add less noise (more accurate)
- *Local Model:* add noise locally (doesn't require trust), more error because can't coordinate noise

Example of local model To give people deniability on a yes/no question, the agent flips two coins and answers truthfully, but if they get two tails they flip their answer. Then still possible to have population level statistics.

DP Synthetic data generation Given a database D, find another database D' that has the same statistical properties as D.

- *Challenges:* datasets are often high dimensional and are required to be correct on many queries:
  - how to measure "accuracy" of a synthetic dataset? (there exist good measures of distance but superpolynomial in the size of the dataset)
  - Computational efficiency of data generation.
- *(Partial) solutions:*

Explaining DP How to communicate to public/policymakers/engineers?

(Partial) solutions: measuring users' privacy expectations from different DP description; finding methods for explaining privacy parameters $\longrightarrow$ **Ongoing work**.

Bibliography

Shipra Agrawal and Nikhil R Devanur. Bandits with concave rewards and convex knapsacks. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 989–1006, 2014.

Shipra Agrawal and Randy Jia. Optimistic posterior sampling for reinforcement learning: worst-case regret bounds. *Advances in Neural Information Processing Systems*, 30, 2017.

Shipra Agrawal, Vashist Avadhanula, Vineet Goyal, and Assaf Zeevi. Mnl-bandit: A dynamic learning approach to assortment selection. *Operations Research*, 67(5):1453–1485, 2019.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

Francis R. Bach and Eric Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate o(1/n). *CoRR*, abs/1306.2119, 2013. URL http://arxiv.org/abs/1306.2119.

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 207–216. IEEE, 2013.

Peter L Bartlett and Ambuj Tewari. Regal: A regularization based algorithm for reinforcement learning in weakly communicating mdps. *arXiv preprint arXiv:1205.2661*, 2012.

David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2010.04.005. URL https://www.sciencedirect.com/science/article/pii/S1063520310000552.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms.* Cambridge University Press, 2020.

Edoardo Di Napoli, Eric Polizzi, and Yousef Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 23(4):674–692, March 2016. doi: 10.1002/nla.2048. URL https://doi.org/10.1002/nla.2048.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *arXiv preprint arXiv:1209.1873*, 2012.

David I. Shuman, Mohammad Javad Faraji, and Pierre Vandergheynst. A multiscale pyramid transform for graph signals. *IEEE Trans. Signal Process.*, 64(8):2119–2134, 2016. doi: 10.1109/TSP.2015.2512529. URL https://doi.org/10.1109/TSP.2015.2512529.