





cnrs formation entreprises

Fundamentals of Machine Learning

Direction de l'innovation et des relations avec les entreprises

Day 3: Supervised learning, regression and classification

Aurélien Garivier, Rémi Gribonval, Mathurin Massias, Titouan Vayer

What you've seen so far

- learn how to recognize a learning problem
- different type of data and variables
- preliminary inspection of data
- illustration on 3 simple datasets
- dimension reduction and clustering

Recap

Machine learning: mapping x to y through some function F_θ

- x: input
- y: output
- F_θ: model
- θ: model parameters
- training samples $(x_i, y_i)_{i \in [n]}$ representative and independent observations

- Goal: predict the value of y for any new x (in particular outside the training data)
- Learning (finding the best value for parameters θ) is done via **loss minimization** on observed data

Today: supervised models

Regression

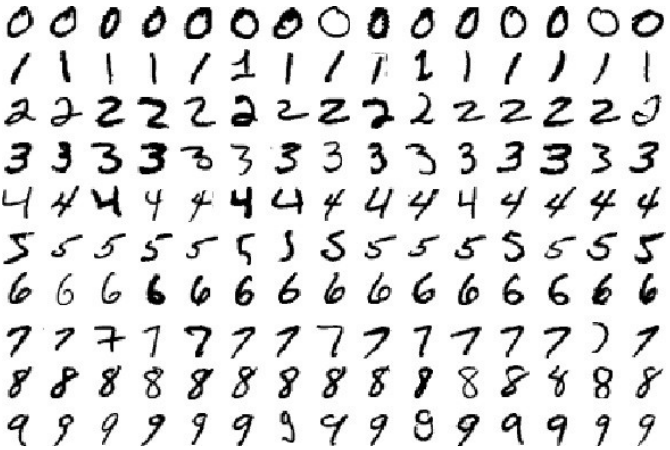
- Observe pairs (x_i, y_i)
- target variable $y \in \mathbb{R}$ or $\mathbb{R}^k$, **continuous**

Classification

- Observe pairs (x_i, y_i)
- target variable $y \in \mathcal{Y}$ **discrete**
- up to relabelling of target values, $\mathcal{Y} = \{0, 1\}$ (binary) or $\{0, 1, \ldots, K\}$ (multiclass)

MNIST: regression or classification?

Difference between regression with output in $[0, K]$ and multiclass classification?



Steps of ML

- gather or obtain data
- explore / visualize (maybe sample)
- cleanup
- transform data
- handle missing data
- train/test split (train/test/validate split)
- **choose and learn model** $\hookrightarrow$ list of models and their properties
- compare models
- exploit

Concepts to take into account

Various reason to favor a model

- interpretability
- confidence (prediction score, confidence interval)
- consistency (optimal when n goes to ∞)
- minimax (stats): this convergence is as fast as possible
- non-asymptotic: guarantees for finite n
- parameter sensitivity: how hard is it to tune hyperparameters?
- computational budget
- handling of online data (update model when gathering new data)
- practical performance

Plan

1 Linear regression

2 Logistic regression

3 k-Nearest neighbors

4 Trees

5 Two ensemble methods

6 Regularization

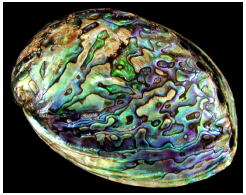
Linear regression

Among the simplest ML models:

- $x \in \mathbb{R}^d$ (collection of information for each sample)
- $y \in \mathbb{R}$ (continuous target value)
- $\theta \in \mathbb{R}^d$ (same dimension as input)
- $F_\theta(x) = \theta^\top x$ (increment in x leads to proportional increment in y)
- classical loss in this framework: $\frac{1}{2n} \sum_1^n (y_i - \theta^\top x_i)^2$ **Mean Squared Error (MSE)**

Example:

$$y = \text{age(abalone)} = \theta^\top x = \theta_1 \times \text{weight} + \theta_2 \times \text{diameter}$$



Intercept in linear models

- intercept = constant term in model
- $y = \theta^\top x + \theta_0$
- can be handled by adding a constant feature: $x \to \tilde{x} = (x, 1) \in \mathbb{R}^{d+1}$
- most scikit-learn linear models fit an intercept by default!
- to change default behavior: `reg = Model(fit_intercept=False)`
- θ is `reg.coef_`, θ_0 is `reg.intercept_` (only exist after fit)

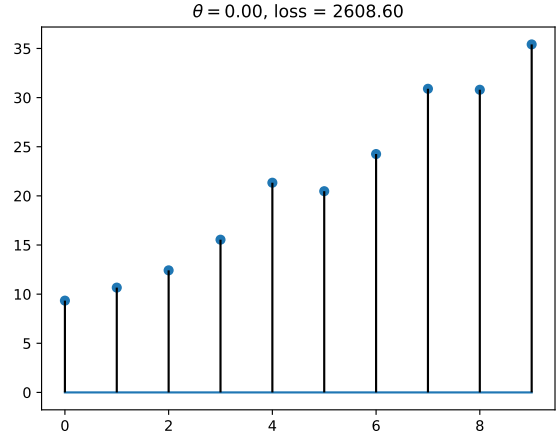
$\hookrightarrow$ notebook illustration

Opening the `LinearRegression` black-box

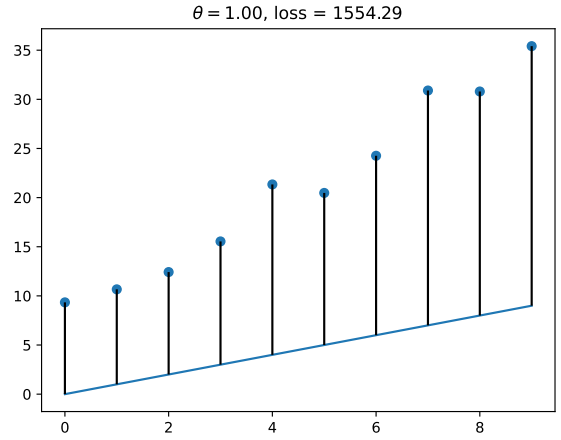
- What happened when we called `reg.fit()`?
- Check in notebook the `coef_` attribute
- Remember the objective function/loss is:

$$\frac{1}{2n}\sum_{i=1}^{n}(y_i - x_i^\top \theta)^2 = \frac{1}{2n}\|y - X\theta\|^2$$

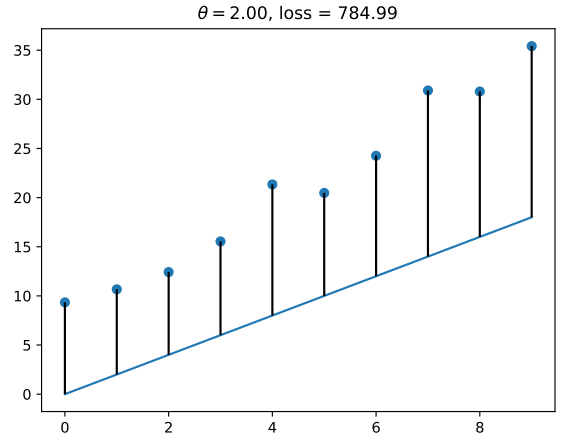
Fitting: “Trial and error” I



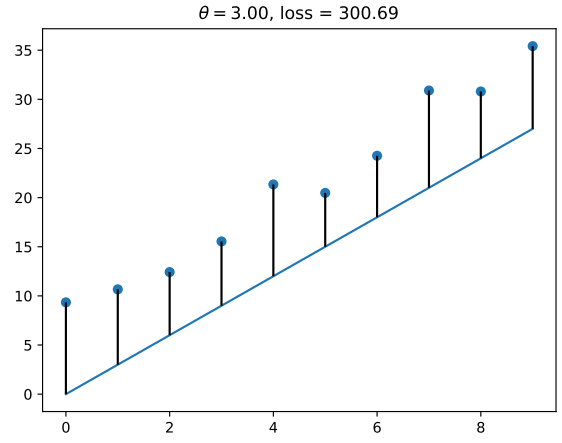
Fitting: “Trial and error” II



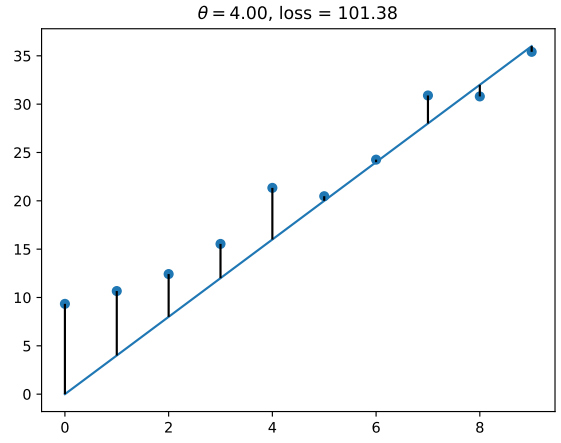
Fitting: "Trial and error" III



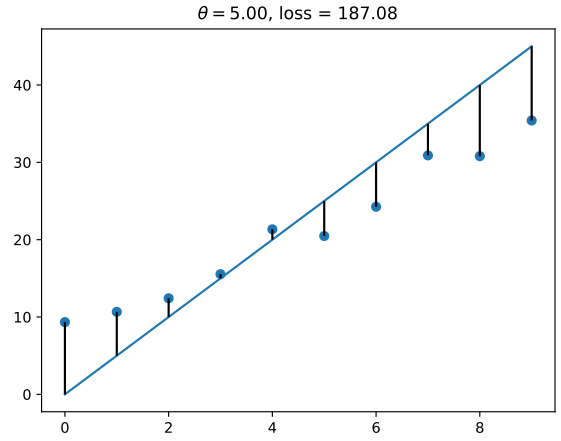
Fitting: “Trial and error” IV



Fitting: “Trial and error” V



Fitting: “Trial and error” VI



Solving least squares: rigorously

Gradient descent for OLS: $\boxed{\theta_{k+1} = \theta_k - \eta \nabla \mathcal{L}(\theta_k) = \theta_k - \eta X^\top (X\theta_k - y)}$

But for ordinary least squares, gradient descent is not needed: we can directly jump to the minimum!

Closed-form solution for least squares

- Recall optimization problem is $\min_\theta \mathcal{L}(\theta) = \frac{1}{2}\|y - X\theta\|^2$
- Objective is differentiable and convex, minimizer satisfies $\nabla \mathcal{L}(\theta^*) = 0$.
- Minimizer satisfies $X^\top X\theta^* = X^\top y \hookrightarrow \theta^* = (X^\top X)^{-1} X^\top y$.
- Works in any dimension, costs $\mathcal{O}(d^2 n)$ for $X \in \mathbb{R}^{n \times d}, d \leq n$.

by the way... this loss is convenient, but where does it come from?

A statistical detour: MLE

- suppose you observe n results of a coin toss, $y_1, \dots, y_n \in \{0, 1\}$ and you want to estimate the probability that it lands on tails.
- Bernoulli model $\mathbb{P}(y = 1) = p \in [0, 1]$.
- likelihood of the observations (probability to observe $(y_1, \dots, y_n)$): $\prod_1^n p^{y_i}(1-p)^{1-y_i}$
- maximize the likelihood = minimize the negative log likelihood = $\sum_1^n y_i \log p + \sum_1^n (1 - y_i) \log(1 - p)$
- closed form: maximum likelihood for $p = \frac{1}{n} \sum_1^n y_i$ (percentage of tails obtained).

OLS = MLE for Gaussian noise

Assume the following generative model:

$$\boxed{\mathbb{P}(Y|X=x) \sim \mathcal{N}(\theta^\top x, \sigma^2)}$$

Then the likelihood is:

$$\prod_1^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_i - x_i^\top \theta)^2\right)$$

and minimizing the negative log likelihood amounts to solving OLS:

$$\theta^* = \arg\min_\theta \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - x_i^\top \theta)^2$$

MLE is the "best" estimator amongst all unbiased estimators (in which sense?)

Plan

1 Linear regression

2 Logistic regression

3 k-Nearest neighbors

4 Trees

5 Two ensemble methods

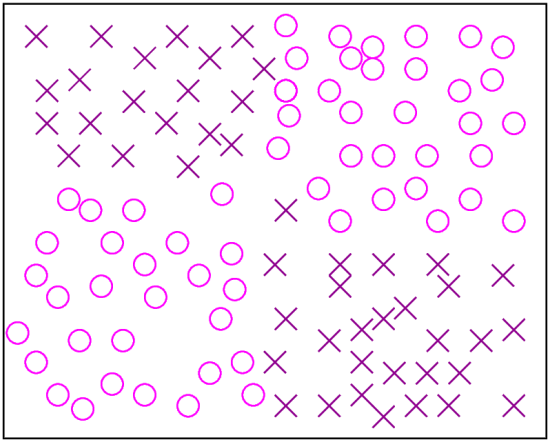
6 Regularization

Logistic regression (bad name!)

- (Generalized) Linear model:
$$F_\theta = \text{sign}(x^\top \theta)$$
- Trained through:
$$\theta^* = \arg\min_\theta \sum_1^n \log\left(1 + \exp(-y_i x_i^\top \theta)\right)$$
- beware: above formulation for $y \in \{-1, 1\}$, not $y \in \{0, 1\}$
- why this loss? $\hookrightarrow$ board
- MLE again: Logreg is MLE for $y \sim \text{Bernoulli}\left(\frac{1}{1+\exp(-x^\top\theta)}\right)$
- logreg provides a score (not a probability): $\frac{1}{1+\exp(-x^\top\theta)}$
- no closed form: needs optimization method (1st or 2nd order)
- limits of linear models $\hookrightarrow$ board

Beyond linearity

How would a Logistic regression classifier deal with this type of data?



Plan

1 Linear regression

2 Logistic regression

3 k-Nearest neighbors

4 Trees

5 Two ensemble methods

6 Regularization

k-Nearest neighbors

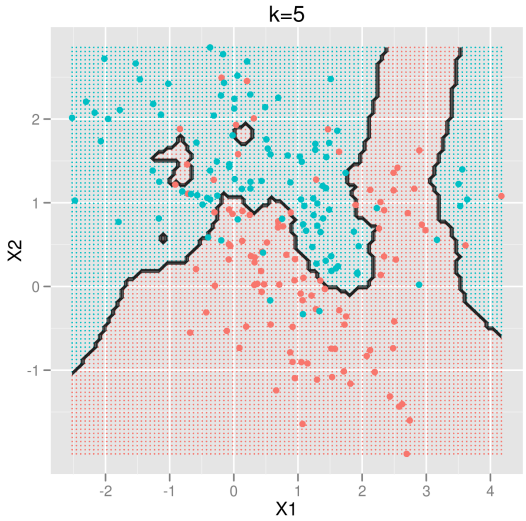
- non parametric model (no assumption on form of data distribution)
- **no parameter** (k is an hyperparameter, it's not learned)
- for new point x, find its k-nearest neighbors in the training data $(x_{(1)}, \dots, x_{(k)})$
- classification: prediction = most frequent class amongst neighbors

$$F_\theta(x) = \arg\max_{y \in \mathcal{Y}} \sum_{j=1}^{k} \mathbb{1}\{y_{(j)} = y\}$$

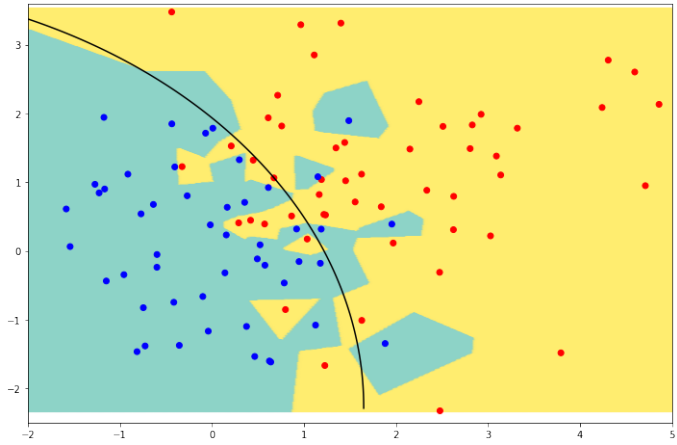
- regression: prediction = average of observed values in neighbors

$$F_\theta(x) = \frac{1}{k} \sum_{j=1}^{k} y_{(j)}$$

Visualisation of k-NN on 2d example

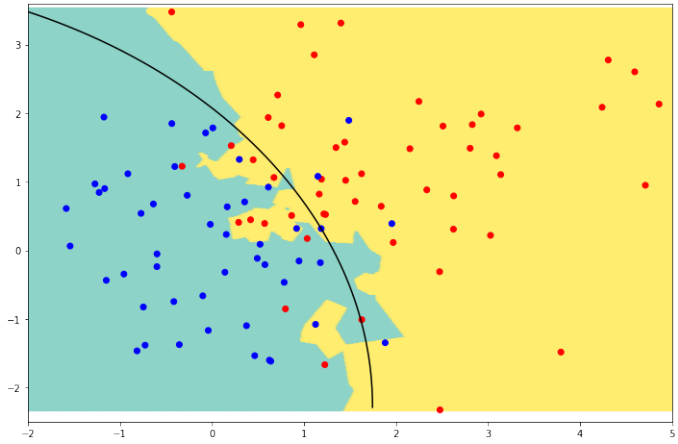


Impact of k on decision function I



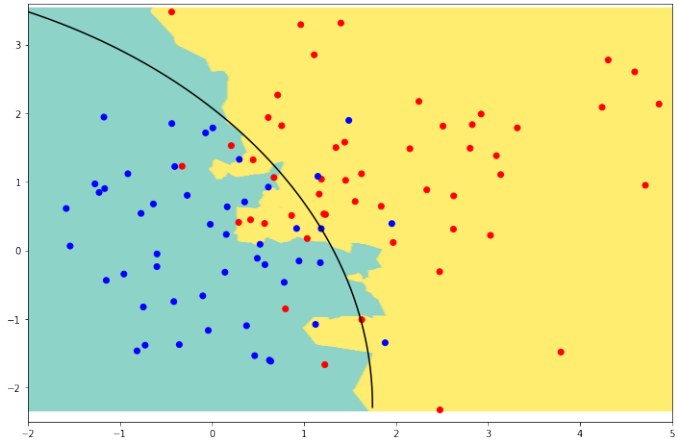
$k = 1$

Impact of k on decision function II



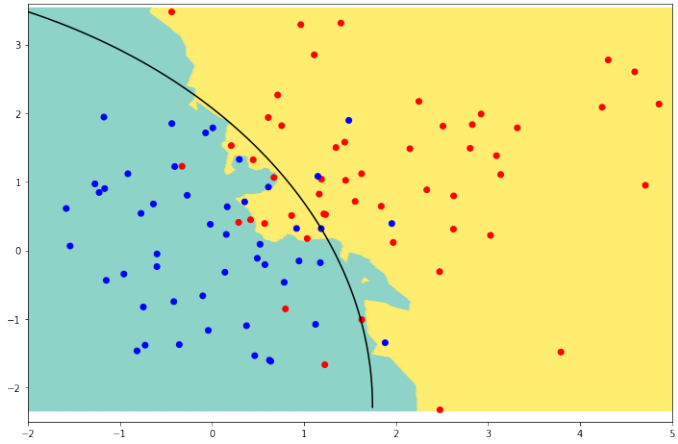
$k = 3$

Impact of k on decision function III



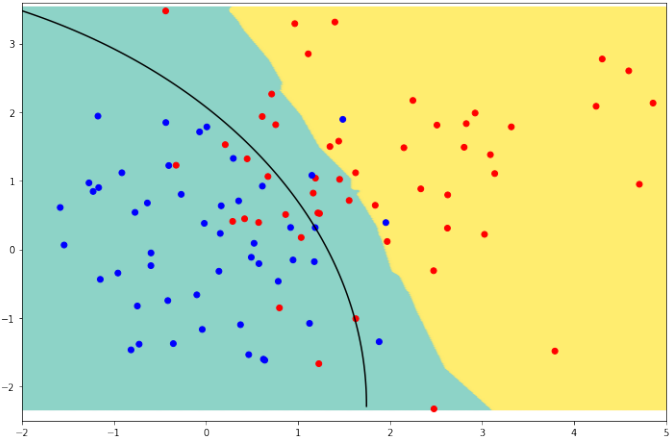
$k = 5$

Impact of k on decision function IV



$k = 7$

Impact of k on decision function V



$k = 95$

k-NN properties

- interpretability: yes and no
- consistency: no but possible if $k = \log(n)$
- minimax: no
- parameter-free: no (k, but other hidden parameter!)
- speed: yes and no, possible in $\mathcal{O}(n \log n)$. Whats the costs of fitting? and predicting?
- online: yes

Plan

1 Linear regression

2 Logistic regression

3 k-Nearest neighbors

4 Trees

5 Two ensemble methods

6 Regularization

CART: Classification and Regression Trees

- non parametric method
- principle: recursively partition the input space into rectangular cells, constant prediction inside each cell
- tree: the partitioning is represented by a tree, each node corresponds to a variable and a split threshold
- adapted to both regression and classification (same principle as k-NN)
- can work directly with categorical variables
- illustration on board

How to grow your tree

Phase 1 : Build a complete tree top to bottom

Recursive partitioning, fromm the root (= whole space)

1. pick a variable and a split which "separates best the data"
   - a threshold (numerical variable)
   - a subset of values (categorical data)
2. iterate over each of the two newly created leaves as long as:
   - each leaf contains enough observations
   - each leaf is not homogeneous enough (from the y point of view)

Phase 2 : Prune the tree bottom to top

For each parent of two children

1. check if the two children are "heterogeneous enough", otherwise merge them
2. go on with the parents

How to measure a cell's heterogeneity

Heterogeneity of a cell: function of labels of points inside that celle

- 0 if y constant in the cell
- maximal when
  - all modalities of y are present in same proportions for classification
  - y spread out for y continuous

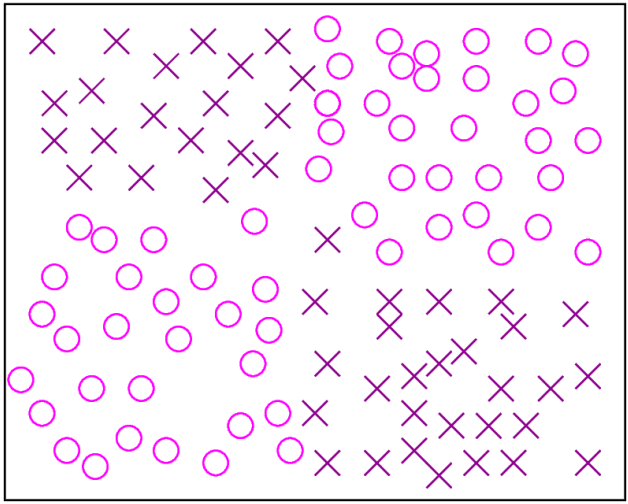
Examples:

- Categorical y:
  - Gini criterion: $N^{-2} \sum_{k \neq \ell} N_k N_l$

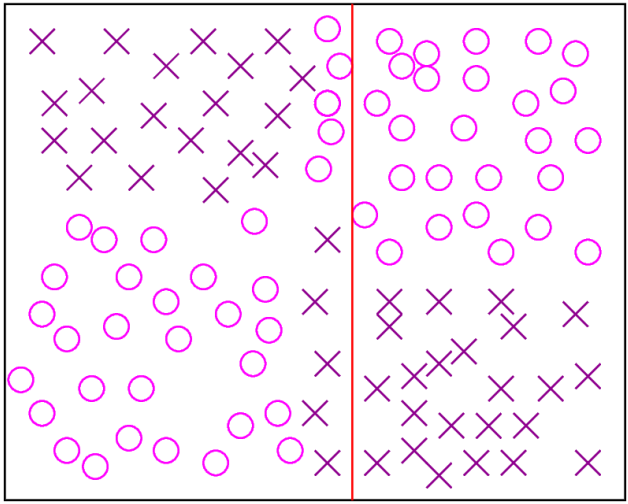
    where $N =$ nb of observations in cell, $N_k =$ nb of observations for which $y = k$
  - entropy : $\sum_k N_k \log \frac{N}{N_k}$
- Continuous y : empirical variance $\sum_i \left(Y_i - \bar{Y}\right)^2$

Splits are found in a greedy way (possible to use random-greedy too)

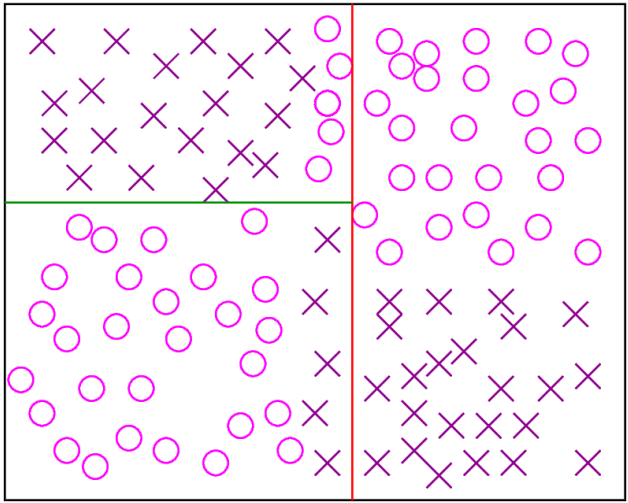
Why do we need the pruning phase? I



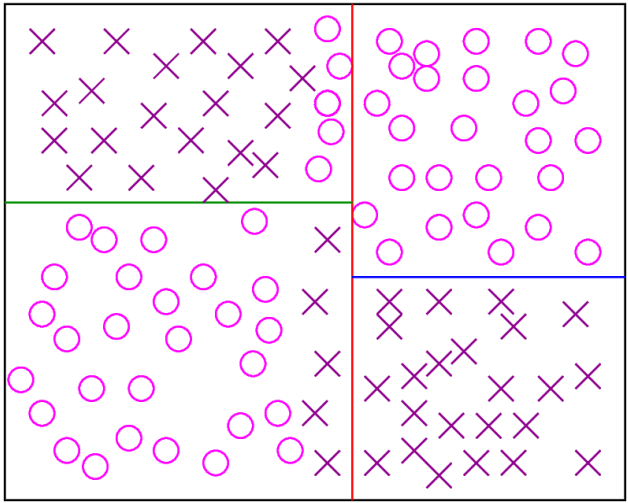
Why do we need the pruning phase? II



Why do we need the pruning phase? III



Why do we need the pruning phase? IV



Why do we need the pruning phase? V

$\Longrightarrow$ making splits even when no immediate gain:

- can lead to great progress in homogeneity in future steps
- but can be useless too...
- $\hookrightarrow$ requires a subsequent pruning phase

Other models: bagging and boosting

Random Forest (bagging)

- group of trees
- each tree is built on a random subset of samples
- at each split, a random subset of features is selected
- prediction = aggregation of each tree's prediction (mean, most frequent)
- can be done in parallel

Adaboost (boosting)

- group of simple models (e.g. shallow trees = *tree stumps*)
- fit a simple inefficient rule
- find next one to fix prediction over points with large error
- aggregate predictions with clever weights
- sequential by design

$\hookrightarrow$ Frequently amongst most successful methods in practice!

Plan

1 Linear regression

2 Logistic regression

3 k-Nearest neighbors

4 Trees

5 Two ensemble methods

6 Regularization

From binary to multiclass

Some estimators are **directly adapted**:

- logistic regression
- k-nearest neighbors
- classification trees

Otherwise any binary classifier can be **canonically transformed** into a multiclass one:

- One Versus Rest: train one binary classifier per class k, predict class $= k$ vs class $\neq k$
- One Versus One: train one binary classifier per pair (k, k')
- then aggregate decision by majority voting
- strengths and weaknesses of both approaches?

Overfitting

- learner's enemy when $n \leq$ #parameters
- a perfect fit on the data is usually not desirable (noise)
- solution here: regularization

$$\min_{\theta} \mathcal{L}(\theta) + \lambda R(\theta)$$

- $\mathcal{L}$ datafit or data fidelity term, R regularizer or penalty
- $\lambda > 0$ controls the trade-off
- what choice for R?

First regularisation for linear models: Ridge

(Generalized) linear model: $F_\theta(x) = f(\theta^\top x)$

Squared Euclidean norm

- penalizes squared ℓ_2 norm:

$$\min_\theta \mathcal{L}(\theta) + \frac{\lambda}{2}\|\theta\|_2^2$$

- smooth, amenable to 1st and 2nd order methods (gradient descent, Newton method)
- always has a unique solution if $\mathcal{L}$ is **convex**
- improves convergence speed of algorithms

Plan

1 Linear regression

2 Logistic regression

3 k-Nearest neighbors

4 Trees

5 Two ensemble methods

6 Regularization

Second regularisation: Lasso

(Generalized) linear model: $F_\theta(x) = f(\theta^\top x)$

Norme L1: $\|\theta\|_1 = \sum_1^p |\theta_i|$

- penalizes ℓ_1 norm:

$$\min_\theta \mathcal{L}(\theta) + \lambda \|\theta\|_1$$

- non-smooth
- but efficient algorithms (proximal methods)
- solution is sparse! $\|\theta^*\|_0 \ll p$

Why the ℓ_1 norm?

Sparsity

- We would like:

$$\min_{\theta} \frac{1}{2}\|X\theta - y\|^2 \quad \text{subject to} \quad \|\theta\|_0 \leq k$$

- issues with this formulation?
- relax it to

$$\min_{\theta} \frac{1}{2}\|X\theta - y\|^2 \quad \text{subject to} \quad \|\theta\|_1 \leq k$$

- under some assumptions, same solutions
- generalisations: group Lasso, Elastic Net, sparse Logistic regression, ...
- other non convexes penalties: MCP, SCAD, ℓ_q $(0 < q < 1)$: better properties