Statistical Learning

Chapter 1. Overview of Supervised learning

Pedro Delicado and Alex Sanchez

UPC & UB

Outline

Supervised Learning (the prediction problem)

- Let (X, Y) be a r.v. with support $\mathcal{X} \times \mathcal{Y} \subseteq \mathbb{R}^p \times \mathbb{R}$.
- General supervised learning or prediction problem:
  - Training sample: $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, i.i.d. from (X, Y).
  - The goal is to define a function (possibly depending on the sample) $h_S : \mathcal{X} \mapsto \mathcal{Y}$ such that for a new independent observation (x_{n+1}, y_{n+1}) , from which we only know x_{n+1}, it happens that:

$$\hat{y}_{n+1} = h_S(x_{n+1}) \text{ is close to } y_{n+1} \text{ (in some sense).}$$

- Function h_S is called generically a *prediction function*. (or classification function or regression function, depending on the case).

Classification and Regression problems

The prediction function h_S is said to describe a *classification* or a *regression* problem depending on the case.

- If $\mathcal{Y} \subseteq \mathbb{R}$ (or $\mathcal{Y}$ an interval) we have a standard *regression problem*.
  - Example: *Relating Salary and demographic variables*
- If $\mathcal{Y} = \{0, 1\}$ (or, also, $\mathcal{Y} = \{-1, 1\}$) we have a problem of *binary classification* or discrimination.
  - Example: *Predicting if a COVID patient will require (or not) ICU*
- If $\mathcal{Y} = \{1, \dots, K\}$ (or $\mathcal{Y} = \left\{y \in \{0,1\}^K : \sum_{k=1}^K y_k = 1\right\}\Big)$ we face a of K classes classification problem.
  - Example: *Classifying a tumor into one of many types*

Supervised learning

- Probabilistic model for supervised learning
  - Response variable Y.
  - Explanatory variables (features) $X = \left(X_1, \ldots, X_p\right)$.
  - Data $\left(x_i = \left(x_{i1}, \ldots, x_{ip}\right), y_i\right), i = 1, \ldots, n$ i.i.d. from the random variable

$$\left(X = \left(X_1, \ldots, X_p\right), Y\right) \sim \Pr(X, Y)$$

- $\Pr(X, Y)$ denotes the joint distribution of X and Y.
  - When this joint distribution is continuous, $\Pr(X, Y)$ is the joint probability density function.

- Main interest is *predicting* Y from X.
- Given the probabilistic model it can be re-stated as *learning the conditional distribution* $\Pr(Y \mid X)$.
- In practice we focus on *learning a conditional location parameter*:

$$\mu(x) = \underset{\mu}{\operatorname{argmin}} \mathbb{E}(L(Y, \mu) \mid X = x),$$

where $L(y, \hat{y})$, loss function, measures the error of predicting y with $\hat{y}$.

- For quadratic loss, $L(y, \hat{y}) = (y - \hat{y})^2$, $\mu(x)$ is the regression function:

$$\mu(x) = \mathbb{E}(Y \mid X = x)$$

Unsupervised learning

- It aims at learning relationships and structure from the observed data.
- Probabilistic model:
  - Variables of interest: $X = \left(X_1, \ldots, X_p\right)$.
  - Data $x_i = \left(x_{i1}, \ldots, x_{ip}\right), i = 1, \ldots, n$ i.i.d. from the random variable
$$X = \left(X_1, \ldots, X_p\right) \sim \Pr(X).$$
- $\Pr(X)$ denotes the probability distribution of X.
  - If X is continuous, $\Pr(X)$ is the probability density function of X.
- Main interest: To infer properties of $\Pr(X)$.

Specific problems in unsupervised learning:

- Estimating directly the density function $\Pr(x)$:
  - Density estimation (histogram, kernel densities, Gaussian MM)
- Detecting homogeneous subpopulations $C_1, \ldots, C_k$ s.t.: $\Pr(x) = \sum_{j=1}^{k} \alpha_j \Pr\left(x \mid C_j\right)$, $\alpha_j \geq 0$, $\sum_j \alpha_j = 1$.
  - Clustering (hierarchical clustering, k-means, ...)
- Finding low-dimensional hyper-planes or hyper-surfaces (manifolds) in $\mathbb{R}^p$ around which the probability $\Pr(x)$ is concentrated.
  - Dimensionality reduction (PCA, MDS, Manifold learning ...)
- Proposing generative probabilistic models for X, depending on low-dimensional unobservable random variables F.
  - Extraction of latent variables (Factor Analysis, ...)

Statistical Decision Theory

- The *prediction problem* can be written as a *decision problem* which can be casted in the setting of *Statistical Decision Theory*.
- Let (X, Y) be a r.v. with support $\mathcal{X} \times \mathcal{Y} \subseteq \mathbb{R}^p \times \mathbb{R}$.
- Prediction problem: To look for a prediction function $h : \mathcal{X} \mapsto \mathcal{Y}$ such that $h(X)$ is close to Y in some sense.
- The (lack of) closeness between $h(X)$ and Y is usually measured by a loss function $L(Y, h(X))$.
  - For instance, the squared error loss is $L(Y, h(X)) = (Y - h(X))^2$.
  - $L(Y, h(X))$ is a r.v., with expected value $\mathrm{EL}(h) = \mathbb{E}(L(Y, h(X)))$, called expected loss, that only depends on h.
- Decision problem: To find the prediction function $h : \mathcal{X} \mapsto \mathcal{Y}$ that minimizes the expected loss.

Bayes rule

- Denote by $\Pr_{(X,Y)}(x, y)$ the joint probability distribution of (X, Y).
- Observe that, for any $h : \mathcal{X} \mapsto \mathcal{Y}$ a lower bound for $\mathrm{EL}(h)$ can be set as follows:

$$
\begin{aligned}
\mathrm{EL}(h) &= \mathbb{E}(L(Y, h(X))) \\
&= \int_{\mathcal{X}\times\mathcal{Y}} L(y, h(x))\, d\Pr_{(X,Y)}(x, y) \\
&= \int_{\mathcal{X}} \left(\int_{\mathcal{Y}} L(y, h(x))\, d\Pr_{Y|X=x}(y) \right) d\Pr_X(x) \\
&= \int_{\mathcal{X}} \mathbb{E}(L(Y, h(x)) \mid X = x)\, d\Pr_X(x) \\
&\geq \int_{\mathcal{X}} \min_{y\in\mathcal{Y}} \mathbb{E}(L(Y, y) \mid X = x)\, d\Pr_X(x) \\
&= \mathrm{EL}\left(h_B\right).
\end{aligned}
$$

- From the previous bound: $\mathrm{EL}(h) \geq \mathrm{EL}\left(h_B\right)$, it follows that, given a loss function $L(y, h(x))$ no prediction function can be better than the Bayes rule, or equivalently, that
- The optimal prediction function is the Bayes rule or Bayes classifier defined as:

$$h_B(x) = \arg\min_{y \in \mathcal{Y}} \mathbb{E}(L(Y, y) \mid X = x).$$

The regression problem

- Let (X, Y) be a $(p+1)$-dimensional random variable, with $Y \in \mathbb{R}$.
- The regression problem: To predict Y from known values of X.
- The most common (and convenient) approach is to adopt as loss function is the *squared error loss:* $L(Y, h(X)) = (Y - h(X))^2$.
- Expected loss known as *Prediction Mean Squared Error, (PMSE)*:

$$\mathrm{PMSE}(h) = \mathbb{E}\left((Y - h(X))^2\right).$$

- The Bayes rule in this case is

$$h_B(x) = \arg\min_{y \in \mathcal{Y}} \mathbb{E}\left((Y - y)^2 \mid X = x\right).$$

- Observe that, for any $y \in \mathcal{Y}$ one can decompose the conditional expectation of the squared deviation between Y and yh given $X = x$, $\mathbb{E}\left((Y-y)^2 \mid X = x\right)$ in such a way that:

$$
\begin{aligned}
&\mathbb{E}\left((Y-y)^2 \mid X=x\right) = \\
&= \mathbb{E}\left(((Y-\mathbb{E}(Y \mid X=x)) + (\mathbb{E}(Y \mid X=x)-y))^2 \mid X=x\right) \\
&= \mathbb{E}\left((Y-\mathbb{E}(Y \mid X=x))^2 \mid X=x\right) + \underbrace{(\mathbb{E}(Y \mid X=x)-y))^2}_{\geq 0} \\
&+ 2(\mathbb{E}(Y \mid X=x)-y)\mathbb{E}(Y-\mathbb{E}(Y \mid X=x) \mid X=x) \\
&\geq \mathbb{E}\left((Y-\underbrace{\mathbb{E}(Y \mid X=x)}_{h_B(x)})^2 \mid X=x\right)
\end{aligned}
$$

Optimal predictor in regression

- From the previous development it yields that, for regression problems, the Bayes rule is the conditional expectation of Y given $X = x$,

$$h_B(x) = \mathbb{E}(Y \mid X = x),$$

- It is also known as regression function of Y over x and is usually denoted by

$$m(x) = \mathbb{E}(Y \mid X = x).$$

Parametric regression

- Parametric regression models assume that $m(x)$ is known except for a finite number of unknown parameters,

$$m(x) \equiv m(x;\theta), \theta \in \Theta \subseteq \mathbb{R}^q,$$

- For instance, the multiple linear regression model postulates that $m(x) = \beta_0 + x^\top \beta_1$, with unknown parameters $\beta_0 \in \mathbb{R}, \beta_1 \in \mathbb{R}^p$.
- A training sample, $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, i.i.d. from (X, Y), is used to estimate the parameter θ.
- In this case $h_S(x) = m(x;\hat{\theta})$, where $\hat{\theta} = \hat{\theta}(S)$ is the estimation of θ from sample S.

Least squares estimation

- A usual way to estimate θ in parametric models is by least squares:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} \left(y_i - m\left(x_i; \theta \right) \right)^2$$

- The regression function $m(x)$ is linear in x.
- It can be shown that, independently of the distributions, $\hat{\theta}$ is the Best Linear Unbiased Estimator (BLUE) of θ.
- Assuming joint normality for X and Y the LS-estimator is equivalent to the maximum likelihood estimator of θ
- In this case, the model is $Y = m(X) + \varepsilon$, where ε is an additive noise normally distributed with zero mean and independent from X, also normally distributed.

Least squares estimation and prediction errors

- The LS estimator $\hat{\theta}$ minimizes the prediction error, RSS, in the training sample.
- That is, the Residual Sum of Squares,

$$\mathrm{RSS}(\theta) = \sum_{i=1}^{n} \left(y_i - m\left(x_i; \theta\right)\right)^2,$$

takes its minimum value when $\theta = \hat{\theta}$

$$\overline{\mathrm{err}} = \mathrm{RSS}(\hat{\theta}) = \sum_{i=1}^{n} \left(y_i - m\left(x_i; \hat{\theta}\right)\right)^2$$

Different types of prediction error

- $\mathrm{RSS}(\theta)$ is the *prediction error* a theoretical quantity, based on the training sample, that needs to be estimated.
- $\overline{\mathrm{err}}$, known as the *training error* or the *apparent error*, is an approximation to $\mathrm{RSS}(\theta)$.
- We are interested in the error associated when predicting a new observation, that is the Prediction Mean Squared Error (PMSE)

$$\mathrm{PMSE}(\theta) = \mathbb{E}\left(\left(Y_{n+1} - m\left(x_i; \theta\right)\right)^2\right),$$

- $\overline{\mathrm{err}}$ is an optimistic estimation of the in an observation of (X_{n+1}, Y_{n+1}) independent from the training sample,

$\overline{\text{err}}$ and PMSE are not the same

- In some cases such as in linear regression $\overline{\text{err}}$ is a good approximation to $\min_{\theta \in \mathbb{R}^q} \text{PMSE}(\theta)$
- But, when the parametric family $m(x;\theta), \theta \in \Theta \subseteq \mathbb{R}^q$, is too flexible:

$$\overline{\text{err}} < \text{PMSE}(\hat{\theta}) \neq \min_{\theta \in \mathbb{R}^q} \text{PMSE}(\theta)$$

- This is the case in non-parametric regression and in many machine learning algorithms. (Example: k-nearest neighbors regression, where the tuning parameter is k).
- We will talk later in the course about cross-validation and tuning parameters.

k nearest-neighbors regression

- K-NN is a flexible approach to regression or classification that, instead of relying on a *global model* based on all observations models each observation locally based on its *nearest neighbors*.
- The k nearest-neighbor estimator of $m(t) = E(Y \mid X = t)$ is:

$$\hat{m}(t) = \frac{1}{|N_k(t)|} \sum_{i \in N_k(t)} y_i,$$

  where $N_k(t)$ is the neighborhood of t defined by the k closest points x_i in the training sample.

- Closeness is defined according to a previously chosen distance measure $d(t, x)$, for instance, the Euclidean distance.

K-NN is flexible or way too flexible

- K-NN regression is is a great real-world example of how model flexibility impacts training error vs. prediction error:
  - When k=1 the model memorizes training data, leading to zero training error.
  - However, for a new test observation, predictions are **highly unstable** (high variance): $\mathrm{PMSE}(\hat{\theta}) \gg \overline{\mathrm{err}}$.
  - As k increases, the model becomes less flexible, reducing variance but increasing bias.
  - The optimal k balances both, minimizing PMSE
- **Conclusion:** Overly flexible models, like small k in k-NN, cause **training error to be misleading**.

Practice session

- The R notebook `knn_regr.Rmd` illustrates the advantages and drawbacks of K-NN regression using R.
- Run along it and experiment with different settings.

The classification problem

- Let (X, Y) be a r.v. with support $\mathcal{X} \times \mathcal{Y} \subseteq \mathbb{R}^p \times \{1, \ldots, K\}$.
- We want to predict Y from observed values of X.
- The loss function in this case can be represented by a $K \times K$ matrix L, that will be zero on the diagonal and nonnegative elsewhere.
  - The element (j, k) of L is $L(j, k)$: the price paid for classifying in class k an observation belonging to class j.

The zero-one loss function

- A common loss function for classification is the zero-one loss function is used, where *all misclassifications are charged a single unit*.
- With the 0-1 loss function the Bayes rule is

$$
\begin{aligned}
h_B(x) &= \arg\min_{y \in \mathcal{Y}} \mathbb{E}\left(L_{0-1}(Y, y) \mid X = x\right) \\
&= \arg\min_{k \in \{1,\ldots,K\}} \sum_{j=1}^{K} L_{0-1}(j, k) \Pr(Y = j \mid X = x) \\
&= \arg\min_{k \in \{1,\ldots,K\}} (1 - \Pr(Y = k \mid X = x)) \\
&= \arg\max_{k \in \{1,\ldots,K\}} \Pr(Y = k \mid X = x).
\end{aligned}
$$

- In this context the Bayes rule is known as the Bayes classifier, and says that we classify to the most probable class, conditional to the observed value x of X.

The problem of binary classification

- In the binary classification problem: $\mathcal{Y} = \{0, 1\}$. Then $(Y \mid X = x) \sim \text{Bernoulli}(p = p(x) = \Pr(Y = 1 \mid X = x) = \mathbb{E}(Y \mid X = x))$.
- The Bayes classifier is

$$h_B(x) = \begin{cases} 1 & \text{if} \quad p(x) \geq 1/2 \\ 0 & \text{if} \quad p(x) < 1/2 \end{cases}$$

- As $p(x)$ is unknown, we use a training sample to estimate it.
- Let $(x_1, y_1), \ldots, (x_n, y_n)$ be n independent realizations of (X, Y).
- Given an estimation $\widehat{p}(x)$ of the regression function $p(x)$, the estimated version of the Bayes classifier is

$$h_S(x_{n+1}) = \begin{cases} 1 & \text{if} \quad \widehat{p}(x_{n+1}) \geq 1/2 \\ 0 & \text{if} \quad \widehat{p}(x_{n+1}) < 1/2 \end{cases}$$

Parametric estimation in binary classification

- In parametric modeling it is assumed that $p(x) = \Pr(Y = 1 \mid X = x)$ is known except for a finite number of unknown parameters,

$$p(x) \equiv p(x; \theta), \theta \in \Theta \subseteq \mathbb{R}^q.$$

- The likelihood and log-likelihood are, respectively:

$$L(\theta) = \prod_{i=1}^{n} \Pr\left(Y_i = y_i \mid X_i = x_i\right) = \prod_{i=1}^{n} p\left(x_i; \theta\right)^{y_i} \left(1 - p\left(x_i; \theta\right)\right)^{1-y}$$

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} \left(y_i \log p\left(x_i; \theta\right) + \left(1 - y_i\right) \log\left(1 - p\left(x_i; \theta\right)\right)\right)$$

- Let $\hat{\theta} = \arg\max_{\theta \in \Theta} \ell(\theta)$ be the maximum likelihood estimator of θ.
- Then $\hat{p}(x) = p(x; \hat{\theta})$ is used to define the classification rule.

Other optimization criteria for binary classification

- Maximum likelihood is not the only possibility for estimating θ in $p(x;\theta)$.
- Alternatives:
  - Minimization of the misclassification error:

$$\hat{\theta}_{\text{Miss}} = \arg\min_{\theta\in\Theta} \sum_{i=1}^{n} \left(y_i - \mathbb{I}\left\{p\left(x_i;\theta\right) \geq 0.5\right\}\right)^2.$$

  - Least squares estimation:
    $\hat{\theta}_{LS} = \arg\min_{\theta\in\Theta} \sum_{i=1}^{n} \left(y_i - p\left(x_i;\theta\right)\right)^2$.
  - Least absolute deviation:
    $\hat{\theta}_{LAD} = \arg\min_{\theta\in\Theta} \sum_{i=1}^{n} \left|y_i - p\left(x_i;\theta\right)\right|$.
  - Penalized version of these criteria, when the statistical model $p(x;\theta), \theta \in \mathbb{R}^q$, is too flexible.

k-Nearest Neighbors (k-NN) for classification

k-Nearest Neighbors (k-NN) is a simple and effective classification method.

It relies on the idea that similar instances should belong to the same class.

- Given a training set ($\mathcal{T}$) with labeled instances (x_i, y_i),
- To classify a new instance x, we:
  1. Find the k closest points x_i to x.
  2. Take the *majority vote$ of their corresponding labels y_i.

The decision boundary of k-NN is **nonlinear** and **flexible**, adapting to local patterns in the data.

k-NN Classification Model

The prediction for a new point (x) is given by:

$$\widehat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

where:

- $N_k(x)$ is the set of the k nearest neighbors of x.
- y_i are the corresponding labels (0 or 1 in binary classification).
- Closeness is typically measured using **Euclidean distance**.

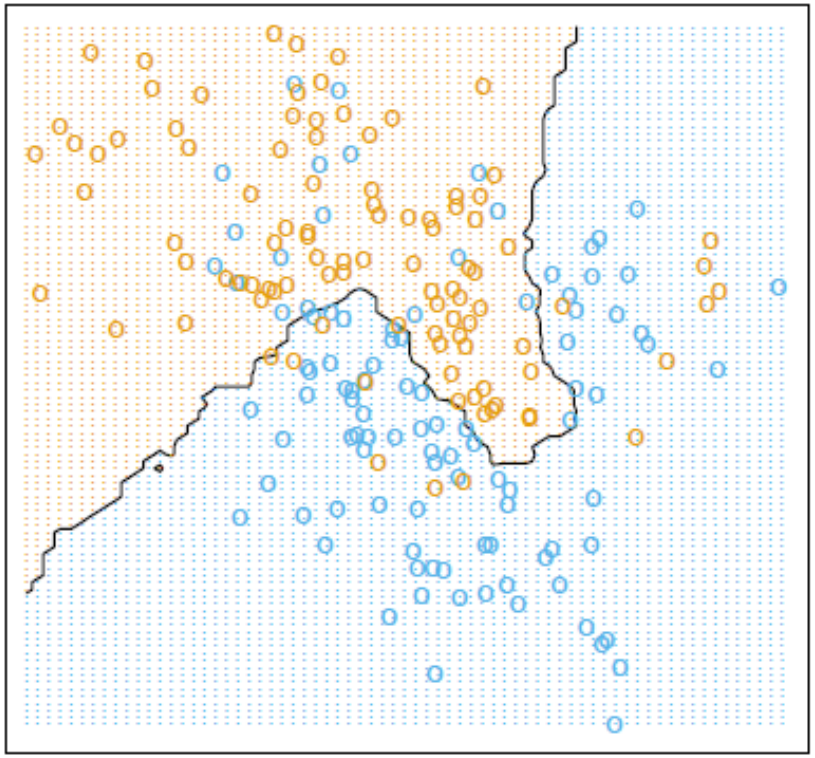
For classification:

- If $\hat{Y}(x) > 0.5$, classify as **Class 1**.
- Otherwise, classify as **Class 0**.

Decision Boundaries - Linear Regression vs. k-NN

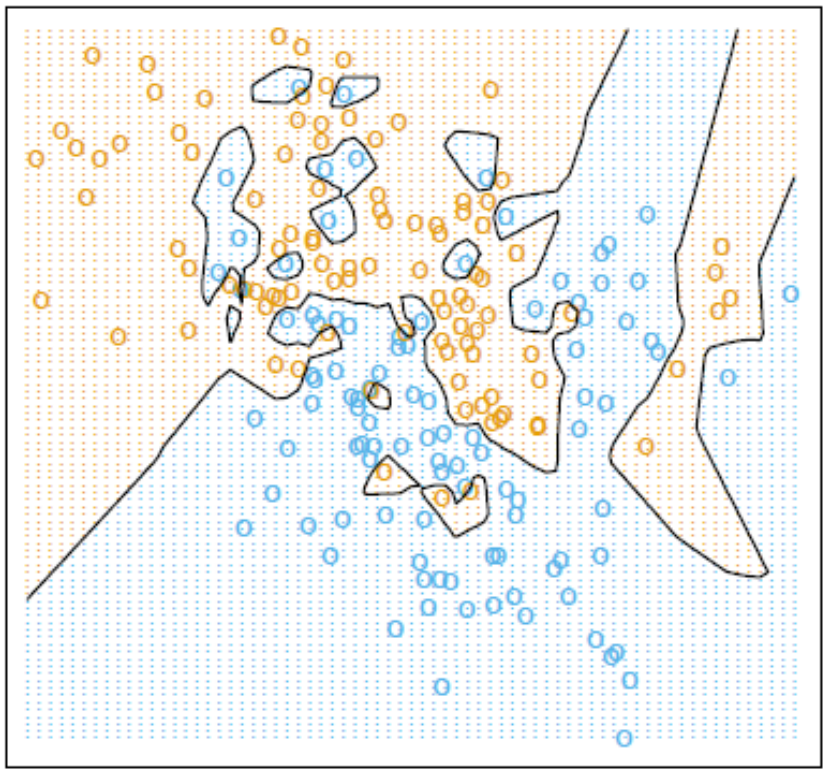
- Linear regression fits a straight line decision boundary: ($x^T\hat{\beta} = 0.5$).
- Misclassifications occur because it assumes a **linear separation**.
- It does not capture **local structures** in the data.

k-NN Decision Boundaries



- Uses a **majority vote** among the 15 closest neighbors.
- The decision boundary is much more **flexible** than linear regression.
- Adapts to **local clusters** of data.

k-NN with k=1



- 1-NN assigns the label of the **single closest** training point.
- Each point belongs to the nearest neighbor's class: **Voronoi tessellation**.
- Decision boundary is **highly irregular** and sensitive to noise.

Choosing k in k-NN

- The parameter k in K-NN reflects its flexibility
  - k=1 leads to **overfitting**, that is, perfect accuracy on training but poor generalization.
  - Larger k smooths the decision boundary but might **lose fine details**.
- **Trade-offs:**
  - Small (k) : **low bias, high variance**.
  - Large (k) : **high bias, low variance**.
  - **Optimal k** is chosen via cross-validation (later in the course) that aims at balancing the former error measures.

Summary

- k-NN is **flexible** and works well for complex decision boundaries.
- It is **non-parametric** and **data-driven**.
- The choice of (k) is critical for **generalization**.
- Compared to linear regression, k-NN adapts better to **nonlinear class distributions**.

k-nn classification, in R

Follow the Rmd files

`SimMixtNorm.Rmd` and `knn_class.Rmd`

Evaluating a binary classification rule

- The explanation has been removed
- Instead you can follow the slides of a talk on Biomarkers where classification performance for bianty classifiers is discussed
    - Link to the slides

Main references