Lecture Notes I – Examples of Predictors

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

September 25, 2023

Prediction problems by the type of output

The "learning" paradigm and vocabulary

Some concepts in Classification

The Nearest-Neighbor and kernel predictors

Linear predictors

- Least squares regression
- Linear Discriminant Analysis (LDA)
- QDA (Quadratic Discriminant Analysis)
- Logistic Regression
- The PERCEPTRON algorithm

Classification and regression tree(s) (CART)

The Naive Bayes classifier

Reading HTF Ch.: 2.3.1 Linear regression, 2.3.2 Nearest neighbor, 4.1–4 Linear classification, 6.1–3. Kernel regression, 6.6.2 kernel classifiers, 6.6.3 Naive Bayes, 9.2 CART, 11.3 Neural networks, Murphy Ch.: 1.4.2 nearest neighbors, 1.4.4 linear regression, 1.4.5 logistic regression, 3.5 and 10.2.1 Naive Bayes,4.2.1–3 linear and quadratic discriminant, 14.7.3– kernel regression, locally weighted regression, 16.2.1–4 CART, (16.5 neural nets), Bach Ch.:

Prediction problems by the type of output

In supervised learning, the problem is *predicting* the value of an **output** (or **response** – typically in regression, or **label** – typically in classification) variable Y from the values of some observed variables called **inputs** (or **predictors, features, attributes**) $(X_1, X_2, \ldots X_d) = X$. Typically we will consider that the input $X \in \mathbb{R}^d$.

Prediction problems by the type of output

In supervised learning, the problem is *predicting* the value of an **output** (or **response** – typically in regression, or **label** – typically in classification) variable Y from the values of some observed variables called **inputs** (or **predictors, features, attributes**) $(X_1, X_2, \ldots X_d) = X$. Typically we will consider that the input $X \in \mathbb{R}^d$. Prediction problems are classified by the type of response $Y \in \mathcal{Y}$:

- *regression*: $Y \in \mathbb{R}$
- *binary classification*: $Y \in \{-1, +1\}$
- *multiway classification*: $Y \in \{y_1, \ldots y_m\}$ a finite set
- *ranking*: $Y \in \mathbb{S}_p$ the set of permutations of p objects
- *multilabel classification* $Y \subseteq \{y_1, \ldots y_m\}$ a finite set (i.e. each X can have several labels)
- *structured prediction* $Y \in \Omega_V$ the state space of a graphical model over a set of [discrete] variables V

Example (**Regression.)**

- Y is the proportion of high-school students who go to college from a given school in given year. X are school attributes like class size, amount of funding, curriculum (note that they aren't all naturally real valued), median income per family, and other inputs like the state of the economy, etc. Note also that $Y \in [0, 1]$ here.
- $Y \geq 0$ is the income of a person, and X_j are attributes like education, age, years out of school, skills, past income, type of employment.

  Economic forecasts are another example of regression. Note that in this problem as well as in the previous, the Y in the previous period, if observed, could be used as a predictor variable for the next Y. This is typical of structured prediction problems.
- Weather prediction is typically a regression problem, as winds, rainfall, temperatures are continuous-valued variables.
- Predicting the box office totals of a movie. What should the inputs be?
- Predicting perovskite degradation. Perovskites are a type of crystal considered promising for the fabrication of solar cells. In standard use, such a material must have a life time Y of 30 years. How can one predict which material will last that long without waiting for 30 years?

  Y is time to degradation, X_j are material composition, experimental conditions, and measurements of initial values of physical parameters.

Example (**(Anomaly) detection.)**

This is a binary classification problem. $Y \in \{\mathrm{normal}, \mathrm{abnormal}\}$. For instance, Y could be "HIV positive" vs "HIV negative" (which could be abbreviated as "+", "-") and the inputs X are concentration of various reagents and lymph cells in the blood.
Anomaly detection is a problem also in artificial systems, as any device may be functioning normally or not. There are also more general detection problems, where the object detected is of scientific interest rather than an "alarm": detecting Gamma-ray bursts in astronomy, detecting meteorites in Antarctica (a robot collects rocks lying on the ice and determines if the rock is terrestrial or meteorite). More recently, *Artificial Intelligence* tasks like detecting faces/cars/people in images or video streams have become possible.

Example (**Multiway classification.)**

Handwritten digit classification: $Y \in \{0, 1, \ldots 9\}$ and X=black/white 64× 64 image of the digit. For example, ZIP codes are being now read automatically off envelopes.
OCR (Optical character recognition). The task is to recognized printed characters automatically. X is again a B/W digital image, $Y \in \{a - z, A - Z, 0 - 9, "." , "," , \ldots\}$, or another character set (e.g. Chinese).

Example (**Diagnosis)**

Diagnosis is multiway classification $+$ anomaly detection. $Y = 0$ means “normal/healthy”, while $Y \in \{1, 2, \ldots\}$ enumerates failure modes/diseases.

Example (**Structured prediction.**)

Speech recognition. X is a segment of digitally recorded speech, Y is the word corresponding to it. Note that it is not trivial to *segment speech*, i.e to separate the speech segment that corresponds to a given word. These segments have different lengths too (and the length varies even when the same word is spoken).

The classification problem is to associate to each segment X of speech the corresponding word. But one notices that the words are not indepedent of other neighboring words. In fact, people speak in sentences, so it is natural to recognize each word in dependence from the others. Thus, one imposes a graphical model structure on the words corresponding to an utterance $X^1, X^2, \ldots X^m$. For instance, the labels $Y^{1:m}$ could form a *chain* $Y^1 - Y^2 - \ldots Y^m$. Other more complex graphical models structures can be used too.

The “learning” paradigm and vocabulary

- **predictor** = a [deterministic] function that associates to an input x a corresponding $\hat{y} = f(x)$.
- A predictor is a kind of model (not yet a statistical model, though).
- **model class** $\mathcal{F}$ = the set of possible predictors for a problem

The “learning” paradigm and vocabulary

- **predictor** = a [deterministic] function that associates to an input x a corresponding $\hat{y} = f(x)$.
- A predictor is a kind of model (not yet a statistical model, though).
- **model class** $\mathcal{F}$ = the set of possible predictors for a problem

- **Training**
  - choose the “best” predictor in $\mathcal{F}$ (for a particular task)
  - based on a **sample** or (**training set**) of **labeled data**

$$\mathcal{D} = \{(x^1, y^1), \, (x^2, y^2), \, \ldots \, (x^n, y^n)\}$$

  - n is the **sample size**.
  - (x^i, y^i) are **examples**
  - In binary classification labels are conventionally in $\{\pm\}$ (or $\{\pm 1\}$). We use the terms **negative**, respectively **positive** examples

The “learning” paradigm and vocabulary

- **predictor** = a [deterministic] function that associates to an input x a corresponding $\hat{y} = f(x)$.
- A predictor is a kind of model (not yet a statistical model, though).
- **model class** $\mathcal{F}$ = the set of possible predictors for a problem

- **Training**
  - choose the “best” predictor in $\mathcal{F}$ (for a particular task)
  - based on a **sample** or (**training set**) of **labeled data**

$$\mathcal{D} = \{(x^1, y^1), (x^2, y^2), \dots (x^n, y^n)\}$$

  - n is the **sample size**.
  - (x^i, y^i) are **examples**
  - In binary classification labels are conventionally in $\{\pm\}$ (or $\{\pm 1\}$). We use the terms **negative**, respectively **positive** examples
- **Prediction** (also called **testing**)
  - Given predictor f, and new input x, calculate

$$\hat{y} \ = \ f(x)$$

Prediction – the workflow

Training phase

- Get labeled data $\mathcal{D} = \{(x^1, y^1), (x^2, y^2), \ldots (x^n, y^n)\}$
- Choose model class $\mathcal{F}$
- Learn/estimate/fit the model $f \in \mathcal{F}$ from data $\mathcal{D}$
  - Here the goal is to find f that predicts $y^{1,\ldots N}$ well
  - How to do it is the **learning algorithm** and depends on $\mathcal{F}$

[**Validation phase** How good really is this f?]

"Testing" phase=Prediction

- now you have a predictor f, use it
- whenever new, unlabeled x comes in, output $\hat{y} = f(x)$

Prediction – the workflow

Training phase

- Get labeled data $\mathcal{D} = \{(x^1, y^1), (x^2, y^2), \dots (x^n, y^n)\}$
- Choose model class $\mathcal{F}$
- Learn/estimate/fit the model $f \in \mathcal{F}$ from data $\mathcal{D}$
  - Here the goal is to find f that predicts $y^{1,\dots N}$ well
  - How to do it is the **learning algorithm** and depends on $\mathcal{F}$

[**Validation phase** How good really is this f?]

Learning Theory

- How to guarantee statistically that f predicts new $y(x)$ well

"Testing" phase=Prediction

- now you have a predictor f, use it
- whenever new, unlabeled x comes in, output $\hat{y} = f(x)$

Classifiers with real-valued output

Binary classification

- Since $y \in \{\pm 1\}$, naturally $f : \mathsf{X} \to \{\pm 1\}$
- But sometimes we prefer a classifier $f : \mathsf{X} \to \mathbb{R}$ (from a predictor class $\mathcal{F}$ of real-valued functions)
- In this case, the prediction $\hat{y}$ is usually

$$\hat{y} \;=\; \operatorname{sgn}(f(x)) \tag{1}$$

  This is sometimes known as the sign trick.

Examples of real-valued classifiers

- Logistic Regression
- Naive Bayes
  in both of the above, $f(x) = P[Y = 1|X = x] \in [0, 1]$. Hence

$$\hat{y} \;=\; \operatorname{sgn}\left(f(x) - \frac{1}{2}\right) \tag{2}$$

- Support Vector Machines
- Kernel classifiers
- Neural Networks

Sign trick

The *sign* function $\operatorname{sgn}(y) = y/|y|$ if $y \neq 0$ and 0 iff $y = 0$ turns a real valued variable Y into a discrete-valued one.

Why real valued f?

- for statistical models $f(x) = P[Y = 1 \mid X = x]$ Example: Logistic regression
- for non-statistical models, $|f(x)|$ measures **confidence** in prediction $\hat{y}$, with $|f(x)| \approx 0$ meaning low confidence. Example: SVM
- if f is differentiable[1], the gradient ∇f is used in **learning algorithms** Examples: Logistic Regression, neural networks, some forms of linear regression such as Lasso

The margin (assuming $y \in \{\pm 1\}$)

- The **margin** of a classifier f at point $x \in \mathrm{X}$ is defined as

$$z \;=\; yf(x). \tag{3}$$

- Note that $\mathrm{sgn}(z) = y\hat{y}$.

- If $z > 0$, $\hat{y} = y$ and $f(x)$ is correct
- If $z \gg 0$, then $f(x)$ is correct, and classifier has high confidence
- If $z < 0$, then $f(x)$ is incorrect, and $|z|$ measures "how wrong" is f on this x
- Note also that $z \approx 0$ means that the classification $\hat{y}$ is not robust, whether correct or not

[1] and ∇f not 0 almost everywhere

Real valued multi-way classifiers

- We train m classifier $f_{1:m} : \mathsf{X} \rightarrow \mathbb{R}$. Then (typically)

$$\hat{y} \;=\; \operatorname*{argmax}_{c=1:m} f_{1:m}(x). \tag{4}$$

- $\hat{y} = y$ means the classifier is correct
- the training can be done
    - independently for each f_c, $c = 1 : m$ (e.g. generative classifiers – in Lecture II)
    - or at the same time (e.g. neural networks, SVM)

- The **margin** is defined as

$$z(x) \;=\; f_y - \max_{c \neq y} f_c(x) \tag{5}$$

In other words

- if $\hat{y} = y$ (correct), then $z = f_{\rm true} - f_{\rm nextbest} \;>\; 0$
- if $\hat{y} \neq y$ (mistake), then $z = f_{\rm true} - f_{\hat{y}} \;<\; 0$ (since $f_{\hat{y}}(x)$ is the max of $f_c(x)$)

Decision regions, decision boundary of a classifier

Let $f(x)$ be a classifier (not necessarily binary)

- $\hat{y}(x)$ takes a finite set of values
- The **decision region** associated with class $y =$ the region in X space where f takes value y, i.e. $D_y = \{x \in \mathbb{R}^d,\, f(x) = y\} = f^{-1}(y)$.
- The boundaries separating the decision regions are called **decision boundaries**.

Decision regions, decision boundary of a classifier

Let $f(x)$ be a classifier (not necessarily binary)

- $\hat{y}(x)$ takes a finite set of values
- The **decision region** associated with class y = the region in X space where f takes value y, i.e. $D_y = \{x \in \mathbb{R}^d,\, f(x) = y\} = f^{-1}(y)$.
- The boundaries separating the decision regions are called **decision boundaries**.

- For a binary classifier, we have two decision regions, D_+ and D_-. By convention $f(x) = 0$ on the decision boundary.
- For binary classifier with real valued $f(x)$ (i.e $\hat{y} = \mathrm{sgn} f(x)$) we define $D_+ = \{x \in \mathbb{R}^d,\, f(x) > 0\}$, $D_- = \{x \in \mathbb{R}^d,\, f(x) < 0\}$ and the decision boundary $\{x \in \mathbb{R}^d,\, f(x) = 0\}$

The Nearest-Neighbor predictor

- **1-Nearest Neighbor** The label of a point x is assigned as follows:
  1. find the example x^i that is nearest to x in $\mathcal{D}$ (in Euclidean distance)
  2. assign x the label y^i, i.e.

$$\hat{y}(x) = y^i$$

The Nearest-Neighbor predictor

- **1-Nearest Neighbor** The label of a point x is assigned as follows:
  1. find the example x^i that is nearest to x in $\mathcal{D}$ (in Euclidean distance)
  2. assign x the label y^i, i.e.

$$\hat{y}(x) \;=\; y^i$$

- **K-Nearest Neighbor** (with $K = 3, 5$ or larger)
  1. find the K nearest neighbors of x in $\mathcal{D}$: $x^{i_1, \ldots i_K}$
  2. - for classification $f(x) =$ the most frequent label among the K neighbors (well suited for multiclass)
     - for regression $f(x) = \frac{1}{K} \sum_{i \text{ neighbor of } x} y^i$ = mean of neighbors' labels

The Nearest-Neighbor predictor

- **1-Nearest Neighbor** The label of a point x is assigned as follows:
  1. find the example x^i that is nearest to x in $\mathcal{D}$ (in Euclidean distance)
  2. assign x the label y^i, i.e.

$$\hat{y}(x) = y^i$$

- **K-Nearest Neighbor** (with $K = 3, 5$ or larger)
  1. find the K nearest neighbors of x in $\mathcal{D}$: $x^{i_1, \ldots i_K}$
  2.
     - for classification $f(x) =$ the most frequent label among the K neighbors (well suited for multiclass)
     - for regression $f(x) = \frac{1}{K} \sum_{i \text{ neighbor of } x} y^i$ = mean of neighbors' labels

- No parameters to estimate!
- No training!
- But all data must be stored (also called memory-based learning)

Kernel regression and classification

- Like the K-nearest neighbor but with "smoothed" neighborhoods
- The predictor

$$f(x) = \sum_{i=1}^{n} \beta_i b(x, x^i) y^i \tag{6}$$

where β_i are coefficients

Kernel regression and classification

- Like the K-nearest neighbor but with “smoothed” neighborhoods
- The predictor

$$f(x) \;=\; \sum_{i=1}^{n} \beta_i b(x, x^i) y^i \tag{6}$$

  where β_i are coefficients
- Intuition: center a “bell-shaped” *kernel* function b on each data point, and obtain the prediction $f(x)$ as a weighted sum of the values y^i, where the weights are $\beta_i b(x, x^i)$
- Requirements for a kernel function $b(x, x')$
  1. non-negativity
  2. symmetry in the arguments x, x'
  3. optional: radial symmetry, bounded support, smoothness
- A typical kernel function is the **Gaussian kernel** (or **Radial Basis Function (RBF)**)

$$b(z) \;\propto\; e^{-z^2/2} \tag{7}$$

$$b_h(x, x') \;\propto\; e^{-\frac{||x-x'||^2}{2h^2}} \quad \text{with} \quad h = \textbf{the kernel width} \tag{8}$$

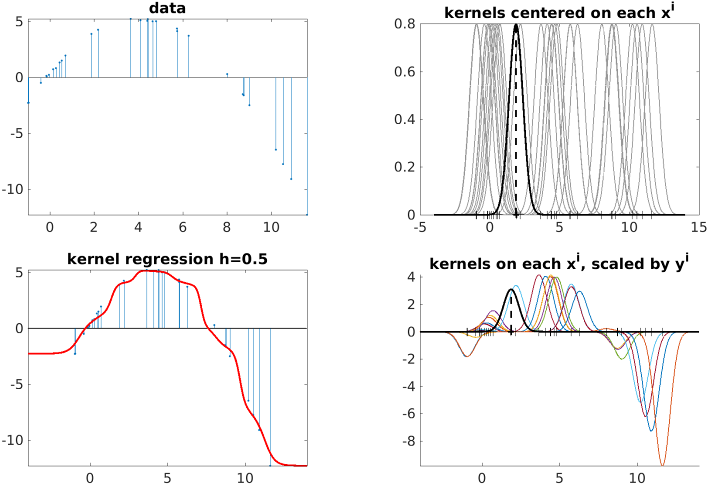
Regression example

A special case in wide use is the Nadaraya-Watson regressor

$$f(x) \;=\; \frac{\sum_{i=1}^{n} b\left(\frac{||x-x^i||}{h}\right) y^i}{\sum_{i=1}^{n} b\left(\frac{||x-x^i||}{h}\right)}. \tag{9}$$

In this regressor, $f(x)$ is always a convex combination of the y^i's, and the weigths are proportional to $b_h(x, x^i)$.
The Nadaraya-Watson regressor is biased if the density of P_X varies around x.

An example: noisy data from a parabola



data

kernels centered on each x^i

kernel regression h=0.5

kernels on each x^i, scaled by y^i

Local Linear Regression

To correct for the bias (to first order) one can estimate a regression line around x.

1. Given **query point** x
2. Compute kernel $b_h(x, x^i) = w_i$ for all $i = 1, \ldots N$
3. Solve weighted regression $\min_{\beta,\beta_0} \sum_{i=1}^{d} w_i \left(y^i - \beta^T x^i - \beta_0\right)^2$ to obtain β, β_0 (β, β_0 depend on x through w_i)
4. Calculate $f(x) = \beta^T x + \beta_0$

Exercise Show that Nadaraya-Watson solves a local linear regression with fixed $\beta = 0$

Kernel binary classifiers

- Obtained from Nadaraya-Watson by setting y^i to ± 1.
- Note that the classifier can be written as the difference of two non-negative functions

$$f(x) \propto \sum_{i:y^i=1} b\left(\frac{||x-x^i||}{h}\right) - \sum_{i:y^i=-1} b\left(\frac{||x-x^i||}{h}\right). \tag{10}$$

(Radial) Basis Function predictors

A regressor similar to the kernel predictor is

$$f(x) \;=\; \sum_{i=1}^{M} b(x, \xi^i)\beta_i \tag{11}$$

The difference is that the "bumps" are not placed on data points, but at a set of M points ξ^i to be determined.

The f in (11) is an example of **function basis** (or **basis functions** approach to prediction. In this approach, we choose a set $\mathcal{B} = \{b(\,;\xi),\ \xi \in \Xi\}$ called a **basis** or a **dictionary**. $\mathcal{B}$ is parametrized by $\xi \in \Xi$; Ξ can be finite or infinite. The elements of $\mathcal{B}$ are called **basis functions**. The predictor class $\mathcal{F}$ is the set of linear combinations of elements of $\mathcal{B}$.

Example

- Fourier basis, wavelet bases
- spline families
- two layer linear output neural networks

Linear predictors

One can classify prediction methods by the type of predictor (i.e. by the type of model class $\mathcal{F}$). The linear predictor is defined as

$$f(x) \;=\; \beta^T x \tag{12}$$

where $Y \in \mathbb{R}$, $X \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$ is a **vector of parameters**.
Model class is $\mathcal{F} = \{\beta \in \mathbb{R}^d\}$ the set of all linear functions over $\mathbb{R}^d$.

Transforming categorical inputs into real values

- if X_j takes two values (e.g "yes", "no"), map it to $\{\pm 1\}$ or $\{0, 1\} \subset \mathbb{R}$.
- discrete multivariate inputs
  - Let X_j take values in $\Omega_j = \{0, \ldots r-1\}$.
  - One defines the $r-1$ binary variables $\tilde{X}_{jk} = 1_{[]} X_j = k,\ k = 1 : r-1$. The variable X_j is replaced with $\tilde{X}_{jk},\ k = 1 : r-1$
  - the parameter β_j with $r-1$ parameters $\beta_{j1} \ldots \beta j, r-1$ representing the coefficients of $\tilde{X}_{j1}, \ldots \tilde{X}_{j,r-1}$.

This substitution is widely used to parametrize any function of a discrete variable as a linear function

Example: The demographic variable Race takes values in $\{$ African, Asian, Caucasian, $\ldots\}$; the corresponding parameters in the model will be $\hat{\beta}_{\mathrm{Asian}}, \hat{\beta}_{\mathrm{Caucasian}}, \ldots$.

The intercept as a slope

- Sometimes we like f to have an intercept $f(x) = \beta^T x + \beta_0$, with $x, \beta \in \mathbb{R}^d$. Such a function is affine, not linear, and not homogeneous. Here is a trick to get the best of both worlds.
- Add a dummy input $x_0 \equiv 1$ to x. Then its coefficient β_0 is the intercept.

$$\tilde{x} \leftarrow \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_d \end{bmatrix} \in \mathbb{R}^{d+1} \quad \tilde{\beta} \leftarrow \begin{bmatrix} \beta_0 \\ \beta_1 \\ \dots \\ \beta_d \end{bmatrix} \in \mathbb{R}^{d+1} \quad f(x) \;=\; \tilde{\beta}^T \tilde{x} \tag{13}$$

- in classification, β_0 is called **threshold** or **bias term**

The linear predictor as classifier

The linear predictor can be used for [binary] classification with the sign trick above.

$$f(x) = \mathrm{sgn} \beta^T x \tag{14}$$

Later in the course we will see a natural way to use real-valued predictors for multi-way classification.
What is the meaning of the β parameter for (14)?
Below we give three possible “interpretations” for β, which correspond three different ways to construct a linear classifier for a problem.

- Problem: how to fit a linear predictor to data? (aka **learn** it)
- Now: examples of what one can do
- Later lectures: larger view of the estimation problem for predictors

(Linear) least squares regression

- define **data matrix** or (transpose) **design matrix**

$$\mathrm{X} \;=\; \begin{bmatrix} (x^1)^T \\ (x^2)^T \\ \cdots \\ (x^i)^T \\ \cdots \\ (x^n)^T \end{bmatrix} \in \mathbb{R}^{N\times n} \quad \text{and} \quad Y \;=\; \begin{bmatrix} y^1 \\ y^2 \\ \cdots \\ y^n \end{bmatrix}, \quad E \;=\; \begin{bmatrix} \varepsilon^1 \\ \varepsilon^2 \\ \cdots \\ \varepsilon^d \end{bmatrix} \in \mathbb{R}^d$$

- Then we can write

$$Y \;=\; X\beta + E$$

- The solution $\hat\beta$ is chosen to minimize the sum of the squared errors $\sum_{i=1}^d (\varepsilon^i)^2 = \sum_{i=1}^d (y^i - \beta^T x^i)^2 = ||E||^2$

$$\hat\beta \;=\; \operatorname*{argmin}_{\beta\in\mathbb{R}^d} \sum_{i=1}^d (y^i - \beta^T x_i)^2$$

- This optimization problem is called a **least squares** problem. Its solution is

$$\hat\beta \;=\; (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T Y \tag{15}$$

- Underlying statistical model $y \;=\; \beta^T x + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2)$ (and i.i.d. sampling of $(x^{1:N}, y^{1:N})$ of course).
  Then $\hat\beta$ from (15) is the Maximum Likelihood (ML) estimator of the parameter β.

Linear Discriminant Analysis (LDA)

Fitting a linear predictor for classification, first approach. (We are in the binary classification case)

- Assume each class is generated by a Normal distribution

$$P_{X|Y}(x|+) = \mathcal{N}(x;\mu_+,\Sigma_+), \quad P_{X|Y}(x|-) = \mathcal{N}(x;\mu_-,\Sigma_-) \quad \text{and } P_Y(1) = p$$

- Given x, what is the probability it came from class $+$?

$$P_{Y|X}(+|x) \;=\; \frac{P_Y(1)P_{X|Y}(x|+)}{P_Y(1)P_{X|Y}(x|+) + P_Y(-)P_{X|Y}(x|+-)} \quad \text{and} \quad P_{Y|X}(-|x) \;=\; 1 - P_{Y|X}(+|x) \quad (16)$$

  This formula is true whether the distributions $P_{X|Y}$ are normal or not.
- We assign x to the class with maximum posterior probability.

$$\hat{y}(x) \;=\; \operatorname*{argmax}_{y\in\{\pm 1} P_{Y|X}(y|x) \qquad (17)$$

  This too, holds true whether the distributions $P_{X|Y}$ are normal or not.

LDA – continued

Now we specialize to the case of normal class distribution. We assume in addition that $\Sigma_+ = \Sigma_- = K^{-1}$.

- Decision rule: $\hat{y} = 1$ iff $P_{Y|X}(+|x) > P_{Y|X}(-|x)$
- or equivalently iff

$$
\begin{aligned}
0 \;\le\; f(x) \;&=\; \ln \frac{P_{Y|X}(+|x)}{P_{Y|X}(-|x)} && (18)\\
&= \ln \frac{p}{1-p} - \frac{1}{2}\left[x^T K x - 2\mu_+^T K x + \mu_+^T K \mu_+\right] \\
&\quad -\frac{1}{2}\left[x^T K x - 2\mu_-^T K x + \mu_-^T K \mu_-\right] && (19)\\
&= \left[K(\mu_+ - \mu_-)\right]^T x + \ln \frac{p}{1-p} + \frac{\mu_-^T K \mu_- - \mu_+^T K \mu_+}{2} && (20)\\
&= \beta^T x + \beta_0 && (21)
\end{aligned}
$$

- The above is a linear expression in x, hence this classifier is called **(Fisher's) Linear Discriminant**
- Note that if we change the variables to $x \leftarrow \sqrt{K}x$, $\mu_\pm \leftarrow \sqrt{K}\mu_\pm$, and if we shift the origin to $\frac{\mu_+ + \mu_-}{2}$ (21) becomes

$$2\mu_+^T x + \ln \frac{p}{1-p} \qquad (22)$$

This has a geometric interpretation

LDA Algorithm

LDA Algorithm

Train

1. Estimate μ_+ from data $\{(x^i, y^i),\ y^i = +1\}$
2. Estimate μ_- from data $\{(x^i, y^i),\ y^i = -1\}$
3. Estimate Σ jointly for both classes, calculate $K = \Sigma^{-1}$. Exercise Derive the formula for this estimate, in the Max Likelihood setting
4. Estimate $p = |\{(x^i, y^i),\ y^i = +1\}|/n$.

Predict Now apply (21) to classify new data x

And QDA (Quadratic Discriminant Analysis)

- If we do not assume $\Sigma_+ = \Sigma_-$ then (18) is a quadratic function of x Exercise Plot the curve $f(x) = 0$ in (18) for various data sets in two dimensions. What kind of curves do you observe? Can the decision region be bounded?

$$
\begin{aligned}
f(x) &= \\
&= \ln\frac{p}{1-p} - \tfrac{1}{2}\ln|\Sigma_+| + \tfrac{1}{2}\ln|\Sigma_-| && (23)\\
&- \frac{1}{2}\left[x^T\Sigma_+^{-1}x - 2\mu_+^T\Sigma_+^{-1}x + \mu_+^T\Sigma_+^{-1}\mu_+\right] && (24)\\
&+ \frac{1}{2}\left[x^T\Sigma_-^{-1}x - 2\mu_-^T\Sigma_-^{-1}x + \mu_-^T\Sigma_-^{-1}\mu_-\right] && (25)\\
&= \left[\ln\frac{p}{1-p} - \tfrac{1}{2}\ln|\Sigma_+| + \tfrac{1}{2}\ln|\Sigma_-| - \tfrac{1}{2}\mu_+^T\Sigma_+^{-1}\mu_+ + \tfrac{1}{2}\mu_-^T\Sigma_-^{-1}\mu_-\right] && (26)\\
&+ \underbrace{\left[\mu_+^T\Sigma_+^{-1} - \mu_-^T\Sigma_-^{-1}\right]x}_{\text{linear}} \underbrace{- \frac{1}{2}x^T\left[\Sigma_+^{-1} - \Sigma_-^{-1}\right]x}_{\text{quadratic}} && (27)
\end{aligned}
$$

Logistic Regression

Fitting a linear predictor for classification, another approach.
Let $f(x) = \beta^T x$ model the **log odds** of class 1

$$f(X) = \frac{P(Y=1|X)}{P(Y=-1|X)} \tag{28}$$

Then

- $\hat{y} = 1$ iff $P(Y=1|X) > P(Y=-1|X)$
    - just like in the previous case! so what's the difference?

Logistic Regression

Fitting a linear predictor for classification, another approach.
Let $f(x) = \beta^T x$ model the **log odds** of class 1

$$f(X) \;=\; \frac{P(Y=1|X)}{P(Y=-1|X)} \tag{28}$$

Then

- $\hat{y} = 1$ iff $P(Y=1|X) > P(Y=-1|X)$
  - just like in the previous case! so what's the difference?
  - Answer: We don't assume each class is Gaussian, so we are in a more general situation than LDA
- What is $p(x) = P(Y=1|X=x)$ under our linear model?

$$\ln\frac{p}{1-p} \;=\; f, \quad \frac{p}{1-p} \;=\; e^f, \quad p \;=\; \frac{e^f}{1+e^f} \quad 1-p \;=\; \frac{1}{1+e^f} \tag{29}$$

An alternative "symmetric" expression for $p, 1-p$ is

$$p \;=\; \frac{e^{f/2}}{e^{f/2}+e^{-f/2}}, \quad 1-p \;=\; \frac{e^{-f/2}}{e^{f/2}+e^{-f/2}}. \tag{30}$$

Estimating the parameters by Max Likelihood

- Denote $y_* = (1-y)/2 \in \{0,1\}$
- The likelihood of a data point is $P_{Y|X}(y|x) = \frac{e^{y_* f(x)}}{1+e^{f(x)}}$
- The log-likelihood is $l(\beta; x) = y_* f(x) - \ln(1+e^{f(x)})$
- $\frac{\partial l}{\partial f} = y_* - \frac{e^f}{1+e^f} = y_* - \frac{1}{1+e^{-f}}$
  This is a scalar, and $\operatorname{sgn}\frac{\partial l}{\partial f} = y$
- We have also $\frac{\partial f(x)}{\partial \beta} = x$
- Now, the gradient of l w.r.t the parameter vector β is

$$\frac{\partial l}{\partial \beta} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial \beta} = \left(y_* - \frac{1}{1+e^{-f(x)}}\right)x \tag{31}$$

Interpretation: The infinitezimal change of β to increase log-likelihood for a single data point is along the direction of x, with the sign of y

Estimating the parameters by Max Likelihood

- Denote $y_* = (1-y)/2 \in \{0,1\}$
- The likelihood of a data point is $P_{Y|X}(y|x) = \frac{e^{y_* f(x)}}{1+e^{f(x)}}$
- The log-likelihood is $l(\beta;x) = y_* f(x) - \ln(1+e^{f(x)})$
- $\frac{\partial l}{\partial f} = y_* - \frac{e^f}{1+e^f} = y_* - \frac{1}{1+e^{-f}}$
  This is a scalar, and $\operatorname{sgn}\frac{\partial l}{\partial f} = y$
- We have also $\frac{\partial f(x)}{\partial \beta} = x$
- Now, the gradient of l w.r.t the parameter vector β is

$$\frac{\partial l}{\partial \beta} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial \beta} = \left(y_* - \frac{1}{1+e^{-f(x)}}\right)x \tag{31}$$

  Interpretation: The infinitezimal change of β to increase log-likelihood for a single data point is along the direction of x, with the sign of y
- Log-likelihood of the data set $\mathcal{D}$

$$l(\beta;\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{d} l(\beta;(x^i,y^i)) \tag{32}$$

- The optimal β maximizes $l(\beta;\mathcal{D})$ and therefore

$$\frac{\partial l(\beta;\mathcal{D})}{\partial \beta} = \frac{1}{N}\sum_{i=1}^{d}\left(y_*^i - \frac{1}{1+e^{-f(x^i)}}\right)x^i = 0 \tag{33}$$

- Unfortunately, (33) does not have a closed form solution!
  We maximize the (log)likelihood by iterative methods (e.g. gradient ascent) to obtain the β of the classifier.

The gradient – an alternative formula

- We use the original y values instead of y_*
- Note that

$$P_{Y|X}(y|x) = \frac{1}{1+e^{-yf(x)}} = \phi(yf(x)) \tag{34}$$

- with $\phi' = \phi(1-\phi)$
- Then, $\frac{\partial \ln P_{Y|X}(y|x)}{\partial f} = \frac{\partial \ln \phi(yf)}{\partial f} = \frac{y\phi(yf)(1-\phi(yf)}{\phi(yf)} = y(1-\phi(yf)$
- The gradient of the log-likelihood of the data is now

$$\frac{\partial l(\beta;\mathcal{D})}{\partial \beta} = \frac{1}{N}\sum_{i=1}^{d}\left(1 - \underbrace{\phi(e^{yf(x^i)})}_{P_{Y|X}(y_i|x^i,\beta)}\right) y_i x^i \tag{35}$$

The Perceptron algorithm

Fitting a linear predictor for classification, third approach.
Define $f(x) = \beta^T x$ and find β that classifies all the data correctly (when possible).
Perceptron **Algorithm**

```
Input labeled training set $\mathcal{D}$
Initialize $\beta = 0$, for all $i$, $x^i \rightarrow \frac{x^i}{||x^i||}$ (normalize the inputs)
Repeat until no more mistakes
    for $i = 1 : N$
        1. if $\mathrm{sgn}(\beta^T x^i) \neq y^i$ (a mistake)
                $\beta \leftarrow \beta + y^i x^i$
```

(Other variants exist)

The perceptron algorithm and linearly separable data

- $\mathcal{D}$ is **linearly separable** iff there is a β_* so that $\mathrm{sgn}\beta_*^T x^i = y^i$ for all $i = 1, \ldots N$.
  If one such β_* exists, then there are an infinity of them

Theorem

Let $\mathcal{D}$ be a linearly separable data set, and define

$$\gamma \;=\; \min_i \frac{|\beta_*^T x^i|}{||\beta_*||||x^i||}. \tag{36}$$

Then, the number of mistakes made by the Perceptron *algorithm is at most $1/\gamma^2$.*

- Note that if we scale the examples to have norm 1, then γ is the minimum distance to the hyperplane $\beta_*^T x = 0$ in the data set.
  Exercise Show that if $\mathcal{D}$ is linearly separable, the scaling $x^i \rightarrow \frac{x^i}{||x^i||}$ leaves it linearly separable.
- If $\mathcal{D}$ is not linearly separable, the algorithm oscillates indefinitely.

Classification and regression trees (CART)

- A **classification tree** or (**decision tree**) is built recursively by splitting the data with hyperplanes parallel to the coordinate axes.
  - At each split, try to separate $+$ examples from $-$ examples as well as possible.
  - Eventually, all the regions will be "pure", i.e. will contain examples from one class only.
- Classification trees can be used in multiway classification as well (there we try to create a pure region on at least one side of the split)
- A **regression tree** uses the same principle for regression
  here we try to obtain regions where the outputs are nearly the same

Hierarchical partitions

- a **hierarchical partition** $\mathcal{T}$ of $\mathbb{R}^d$ is a set of regions $\{R_q\}$, so that $\mathbb{R}^d = \bigcup_q R_q$ and between any two $R_q, R_{q'}$ we have either

$$R_q \cap R_{q'} = \emptyset, \text{ or } R_q \subset R_{q'} \text{ or } R_{q'} \subset R_q. \tag{37}$$

- In a CART, the partitions are usually chosen to be axis-aligned, i.e. $R_q = \{x \mid \pm x_{j_1}" > "\tau_1, \pm x_{j_2}" > "\tau_2, \ldots \pm x_{j_l}" > "\tau_l\}$ where $" > "$ stands for one of $>$ or $\geq$, so that the union of all regions covers $\mathbb{R}^d$.
- The number of inequalities l defining the region is called the *level* of the region.
- R_q is a **leaf** of $\mathcal{T}$ iff there is no other $R_{q'}$ included in it.

Example (**A hierarchical partition with 3 levels over $\mathbb{R}^2$**)

Level 1: $R_1 = \{x \mid x_2 > 3\}$,
$R_2 = \{x \mid x_2 \leq 3\}$
Level 2: $R_3 = \{x \mid x_2 > 3,\ x_1 \geq -2\}$,
$R_4 = \{x \mid x_2 > 3,\ x_1 < -2\}$,
$R_5 = \{x \mid x_2 \leq 3, x_1 > 0\}$,
$R_6 = \{x \mid x_2 \leq 3, x_1 \leq 0\}$
Level 3: $R_7 = \{x \mid x_2 > 3,\ x_1 \geq -2, x_1 < 4\}$,
$R_8 = \{x \mid x_2 > 3,\ x_1 \geq 4\}$,
$R_9 = \{x \mid x_2 < 3, x_1 \geq 1\}$
$R_{10} = \{x \mid x_2 \leq 3, x_1 \leq 0, x_2 > -1\}$,
$R_{11} = \{x \mid x_2 \leq -1, x_1 \leq 0\}$,
$R_{12} = \{x \mid x_2 < 3, x_1 > 0, x_1 < 1\}$

The leaves are $R_4, R_7, \ldots R_{12}$. Not all leaves are at the same level; for example R_4 is at level 2.

Purity

- Natural ways to set y_q based on the data, once the partition $\mathcal{T}$ has been fixed:
  - denote $Y_q = \{y^i \,|\, x^i \in R_q,\ i = 1:N\}$ the set of labels at a leaf R_q
  - Regression y_q = average of Y_q
  - Classification y_q = majority label of Y_q
- a leaf R_q is **pure** if all labels are the same, i.e. if $|Y_q| = 1$
- criteria for the **(im)purity** of a leaf R_q
  - Regression impurity = sample variance of Y_q
  - Classification let p_q be the frequency of y_q in Y_q

$$\text{impurity} \;=\; \begin{cases} \text{Misclassification error} & 1 - p_q \\ \text{Gini} & p_q(1-p_q) \\ \text{Entropy} & p_q \ln p_q + (1-p_q)\ln(1-p_q) \end{cases} \tag{38}$$

These measures generalize naturally to the multiclass setting.

“Learning” a CART

A standard algorithm for building a decision tree works recursively in top-down fashion.

Input Training set $\mathcal{D}$ of size n

Initialize with a tree with only one region, that contains all the data

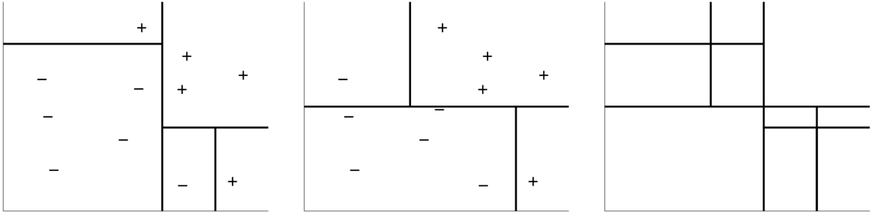
Repeat until all leaves are pure (or until desired purity is attained in all leaves)

1.
2. Find the “optimal” split over all leaves R_q and all possible splits of R_q.
   “Optimal” is defined in terms on purity (e.g split the least pure leaf, find the split that makes one of the new leaves pure)
3. Perform the “optimal” split and add the two new leaves to the tree

This is a greedy algorithm. Sometimes, trees obtained this way are pruned back to smaller sizes.

A decision tree over $\mathcal{D}$ is not unique

Same dataset $\mathcal{D}$, two different trees. Both classify the sample $\mathcal{D}$ perfectly.



$\mathcal{T}_1$ $\mathcal{T}_2$ differences of $\mathcal{T}_1$, $\mathcal{T}_2$

But they produce different decision regions.

The Naive Bayes classifier

- Idea: Assume the distribution in each class factors

$$P_{X|Y=y} \;=\; \prod_{j=1}^{d} P_{X_j|Y=y} \tag{39}$$

  In other words, the n attributes are independent given the class y.
- Then, $P_{Y|X}$ is given by Bayes' rule as follows

$$P_{Y|X}(+1|x) \;=\; \frac{P_Y(1)P_{X|Y}(x|+1)}{P_Y(1)P_{X|Y}(x|+1)+P_Y(-1)P_{X|Y}(x|-1)} \tag{40}$$

- The advantage of this classifier is in its simplicity
  1. for each class y, and each attribute j estimate $P_{X_j|Y=y}$
  2. for each class y, estimate $P_Y(y)$.
     (works for multiclass too!) Exercise Write the NB formula below for multiclass.
- For binary classification (as in LDA), we obtain

$$f(x) \;=\; \ln\frac{P_{Y|X}(+1|x)}{P_{Y|X}(-1|x)} \;=\; \ln\frac{p}{1-p}+\sum_{j=1}^{d}\ln\frac{P_{X_j|Y}(x_j|+1)}{P_{X_j|Y}(x_j|-1)} \tag{41}$$

  This is an additive model (see later) and in some cases it can be a linear model.