Fundamentels of Machine Learning

Ullrich Köthe

WS 2017, Heidelberg

Skript zur Vorlesung an der Universität Heidelberg

Inhaltsverzeichnis

Rational of Machine Learning

- Interest in attributes/quantities Y ("response"), but they are not easily measurable.
- Choose attributes/quantities X ("features"), that are easy to measure.
- find a mapping $Y = f(X)$ to determine Y indirectly
- many problems don't have an explicit analytical $f(X)$
- $\Rightarrow$ use "generic mapping": $Y = f(X, \theta)$, θ : adjustable parameters (ideally a universal approximate) and adjust θ by learning from a training set:

$$TS = \{(X_i, Y_i)\}_{i=1}^{N}$$

- usually the relation between X and Y is not deterministic
- posterior probability:

$$p(Y|X;\theta)$$

$$f(X) = \begin{cases} Y_1 & \text{with probability } p(Y = Y_1|X;\theta) \\ Y_2 & \text{with probability } p(Y = Y_2|X;\theta) \end{cases}$$

Kinds of Variables

- numeric: $X \in \mathbb{R}^D, Y \in \mathbb{R}^M$ (usually M=1)
- discrete: $X \in \{A, B, C, ...\}$
- ordinal: categories are ordered $A < B < C$
- categorical: $X \in \{"red", "green", "blue"\}, Y \in \{"pea", "pear", "peach"\}$

if response is discrete $\Rightarrow$ classification
if response is numeric $\Rightarrow$ regression

Notation

- instances in training set subscript $i \in \{1, ..., N\}$
- X_i : features of training instance (row vector)
- Y_i : response
- features we measure: subscript $j \in \{1, ..., M\}$
- X_j : j-th feature of all instances (column vector)
- X_{ij} : j-th feature of instance i (a scalar)

Kinds of training data

- supervised learning: Y_i is known for all training instances
  - possible if we can obtain Y_i in a research setting
  - measuring, asking expert
  - measuring Y_i is destructive (crash test)
  - Y_i is only known in hind sight
  - we know the mapping $Y_i = f(Y_i)$ for $TS\{(X_i, Y_i)\}$
  - training strategies:
    - watch training: TS is given beforehand
    - online training
    - active training
- unsupervised learning: X_i are known, Y_i not ("data mining")
  - group data by similarity
  - determine useful categories
  - find interesting features
  - estimate probability distribution $p(X)$
  - novely detection - find unusual X

1 Classification

1.1 Rules

- instances are i.i.d. (independent identically distributed)
- Y is discrete with C categories $Y \in \{1, 2, ..., C\}$
  $C = 2 : Y \in \{0, 1\}, \{-1, 1\}$
- X is numeric or discrete
- if the outcome is certain: estimate posterior probability $p(Y|X; \theta)$
- but in many situations a hard decision is needed:

Example: hiring X, credentials: $p(Y = \text{will design good bike}|X)$ needs a hard decision function:

$$f(x) = \begin{cases} \text{hire if X is convincing} \\ \text{not hire else} \end{cases}$$

1.2 How badly does a bad decision function perform?

Suppose we know the prior probability of each category (prior - before, without measuring X)

$$p(Y = K) = \pi_k \quad k = 1, ..., C$$

$$\Rightarrow \text{most sensible decision function:} \quad f(k) = \arg\max_k \pi_k = \hat{k}$$

$$\text{success rate} = \pi_{\hat{k}}$$

$$\text{error rate} = 1 - \pi_{\hat{k}} = 1 - \arg\max_k \pi_k$$

1.3 How good can a decision function perform in an uncertain environment?

Sources of uncertainty:

- intrinsic uncertainty:
  - fundamental(Q.M.)
  - noise (can measure X and Y only to certain accuracy)
- insufficient knowledge:
  - X may not have enough information to determine Y exactly
  - missing data
- modelling uncertainty:
  - $Y = f(X, \theta)$ may not be powered enough to express the true relationship $Y = f^*(X)$
  - θ may not be set to the optimal values

Assume that there is no modelling error and no noise in Y, no missing data

$$\Rightarrow \text{our posterior:} \quad \hat{p}(Y|X) = p^*(Y|X)$$

so we know the probability distribution.
What decision funtion $\hat{f}$ minimizes the error fo C = 2?

$$p^*(Y=1|X) \quad p^*(Y=-1|X)$$

$$\text{Two decision functions:} \begin{cases} f_1(X) = f(X;\theta_1) = 1 \\ f_{-1}(X) = f(X;\theta_{-1}) = -1 \end{cases}$$

choose f_1:

- a) true positive $Y^*(X) = 1$
- b) false positive $Y^*(X) = -1$

choose f_{-1}:

- c) true negative $Y^*(X) = -1$
- d) false negative $Y^*(X) = 1$

Probabilities: a) = d): $p^*(Y=-1|X) = 1 - p^*(Y=1|X)$
Success is maximized if we always decide fo most probable outcome, given X:

$$\hat{Y} = \hat{f}(X) = \arg\max_k p^*(Y=K|X) \quad \text{Bayes classifier rule}$$

$$\begin{aligned} p(error_{Bayes}) &= \mathbb{E}_x[p(error(x))] \\ &= \int (1 - \max_k(y=k|x))p^*(x)dx \quad x \in \mathbb{R}^D \text{ with } p^*(x) \end{aligned}$$

Definition: Decision regions are connected regions in $\mathbb{R}^D$ where $\hat{f}(x) = const.$

1.4 Discriminative vs. Generative models

$$\text{Bayes Rule:} \quad p(Y|X) = \frac{p(X|Y) * p(Y)}{p(X)}$$

$$\text{posterior} : p(Y|X) \quad \text{likelihood} : p(X|Y) \quad \text{prior} : p(Y) \quad \text{data density/evidence} : p(X)$$

discriminative model: learn LHS of Bayes: p(Y|X)
generative model: learn RHS of Bayes: p(Y), p(X|Y), p(X)

$$p(X) = \sum_{K=1}^{N} p(X|Y=K)p(Y=K)$$

⇒ can be used to create more data by simulation, disadvantage: usually needs more training data for the same success rate

$$C = \text{Bayes decision function} \quad f(x) = \begin{cases} +1 & \text{if} \quad p(y=1|x) \geq p(y=-1|x) \\ -1 & else \end{cases}$$

$$\frac{p(x|y=1)p(y=1)}{p(x)} > \frac{p(x|y=-1)p(y=-1)}{p(x)}$$

$$\iff \quad \frac{p(x|y=1)p(y=1)}{p(x|y=-1)p(y=-1)} \geq 1$$

$$\iff \quad \log(p(x|y=1)p(y=1)) - \log(p(x|y=-1)p(y=-1)) \geq 0$$

$$\iff \quad \log(\frac{p(x|y=1)}{p(x|y=-1)}) + log(\frac{\pi_1}{\pi_{-1}}) \geq 0$$

$$\pi_k = \frac{1}{C} = const \quad \pi_1 = \pi_{-1} = \frac{1}{C}$$

$$\Rightarrow \quad \text{max. likelihood decision rule} \quad f(x) = \begin{cases} 1 & if \frac{p(x|y=1)}{p(x|y=-1)} \geq 1 \\ -1 & else \end{cases}$$

Example: Y ∈ red, blue, X ∈ ball pen, marker

$$\pi_{red} = \pi_{blue} = 0.5$$

$$p(marker|red) = \frac{5}{7} \quad p(marker|blue) = \frac{2}{7} \quad p(ball|red) = \frac{2}{7} \quad p(ball|blue) = \frac{5}{7}$$

$$p(ball) = p(ball|y=red) * p(red) + p(ball|y=blue) * p(blue)$$
$$= \frac{2}{7} * \frac{1}{2} + \frac{5}{7} * \frac{1}{2} = \frac{1}{2}$$

$$p(red|ball) = \frac{p(ball|red)}{p(ball)} = \frac{\frac{2}{7} * \frac{1}{2}}{\frac{1}{2}}$$

$$p(blue|ball) = \frac{5}{7} \quad p(red|marker) = \frac{5}{7} \quad p(blue|marker) = \frac{2}{7}$$

⇒ Bayes decision:

$$f(x=marker) = \arg\max_k p(y=k|marker) \quad = \text{red}$$

$$f(x=ball) = \arg\max_k p(y=k)|ball) \quad = \text{blue}$$

1.5 Nearest Neighbor Classification

Intuition: in an unknown situation, act as you did in the most similar situation in the past

- 'past': training set
- 'act as you did': copy the training label to the new instance

For 'most similar' we need a distance function between features $d(x, x')$

$$\text{decision rule} : f_{NN}(x) = y_i, \quad i = \underset{n \in Trainingset}{\arg\min} \quad d(x_i, x_{i'})$$

effect: split feature space according to the distanc to training examples

$$neighbors(x_i) = \{x | d(x, x_i) \leq d(x, x_{i'})\} \quad \forall i \neq i'$$

'Voroni tessellation' with centers $\{x_i\}_{i=1}^N$
each region is a voroni cell of x_i
decision boundaries: bisectors between centers

1.5.1 Performance Analysis of NN classifier

- derive analytic formulas for error for finite training set, this ist the best, but usually very difficult
- derive analytic error formulas in the limit for infinitly many training data 'asymptotic analysis', this is usually easier, but often unrealistic (when error decreases slowly with N)
- measure error empirically on independent test data('ground truth'): most ralistic, but must be repeated for every model and application, beware of the multiple testing bias if test data is reused

finite sample analysis example:

$$C = 2 \quad y \in \{0, 1\}, \quad p(y = 0) = p(y = 1) = \frac{1}{2}$$

$$p(x|y = 0) = 2 - 2x \quad p(x|y = 1) = 2x$$

$$\int_0^1 P(x|y = 0)dx = \int_0^1 (2 - 2x)dx$$
$$= 2x - x^2 \Big|_0^1 = 1$$

$$\begin{aligned} p(x) &= p(x|y = 0)p(y = 0) + p(x|y = 1)p(y = 1) \\ &= (2 - 2x)\frac{1}{2} + 2x\frac{1}{2} \\ &= 1 - x + x = 1 \end{aligned}$$

$$\begin{aligned} \text{postkrias} \quad p(y = 0|x) &= \frac{p(x|y = 0)p(y = 0)}{p(x)} \\ &= \frac{(2 - 2x)\frac{1}{2}}{1} \\ &= 1 - x \end{aligned}$$

$$p(y=1|x) = \frac{2x\frac{1}{2}}{1} = x$$

define two decision rules with 'threshold' t:

$$A: \quad f_A(x;t) = \begin{cases} 0 & \text{if } x \leq t \\ 1 & \text{if } x > t \end{cases}$$

$$B: \quad f_B(x;t) = \begin{cases} 1 & \text{if } x \leq t \\ 0 & \text{if } x > t \end{cases}$$

$$\mathbb{1}[condition] = \begin{cases} 1 & \text{if condition = true} \\ 0 & \text{if condition = false} \end{cases} \quad \text{'Indicator function'}$$

$$\begin{aligned} p(A;t|error) &= \mathbb{E}_x[p(f_A(x;t) \neq Y^*|t)] \\ &= \mathbb{E}_x[p(y=1|x)\mathbb{1}[x \leq t]] + \mathbb{E}_x[p(y=0|x)\mathbb{1}[x;t]] \\ &= \int_0^1 p(y=1|x)\mathbb{1}[x]p(x)dx + \int_0^1 p(y=0|x)\mathbb{1}[x > t]p(x)dx \\ &= \int_0^1 p(y=1|x)p(x)dx + \int_t^1 p(y=0|x)p(x)dx \\ &= \int_0^t xdx + \int_t^1 (1-x)dx \\ &= \frac{x^2}{2}\Big|_0^t + (x - \frac{x^2}{2})\Big|_t^1 \\ &= \frac{t^2}{2} - 0 + 1 - \frac{1}{2} - t + t^2 \\ &= t^2 - t + \frac{1}{2} = (t - \frac{1}{2})^2 + \frac{1}{4} \\ &= p(error|A,t) \end{aligned}$$

$$p(error|B,t) = \frac{3}{4} - (t - \frac{1}{2})^2 = 1 - p(error|A,t)$$

Bayes classifier(minimizes error): rate A with $t = \frac{1}{2}$

$$p(error|A, t = \frac{1}{2}) = \frac{1}{4}$$

Error of NN classifier, simplest possible TS with N = 2:

- sample z = (x, y)
- repeat: sample z' = (x', y') until $y' \neq y$, 'rejection sampling'

Two possible outputs:

$$\left.\begin{aligned} A: x_0(y_0=0) \leq x_1(y_1=1) : ruleA \\ B: x_0(y_0=0) > x_1(y_1=1) : ruleB \end{aligned}\right\} t = \frac{x_0 + x_1}{2}$$

$$p(error) = \mathbb{E}_{TS}[p(error|TS)] = \mathbb{E}_{TS(A)}[p(error(A,t))] + \mathbb{E}_{TS(B)}[p(error|B,t)]$$

$$
\begin{aligned}
\mathbb{E}_A &= \int_0^1 \int_0^1 \int_0^1 \underbrace{p(error|A,t)}_{=(t-\frac{1}{2})^2+\frac{1}{4}} \quad \underbrace{p(A,t|x_0,x_1)}_{=\mathbb{1}[x_0 \leq x_1]\delta(t-\frac{x_0+x_1}{2})} \quad \underbrace{p(x_0,x_1)}_{=p(x|y=0)p(x|y=1)} \quad dt dx_1 dx_0 \\
&= \int_0^1 p(x|y=0) \int_{x_0}^1 p(x|y=1) p(error|A, t = \frac{x_0+x_1}{2}) dx_1 dx_0 \\
&= \int_0^1 (2-2x) \int_{x_0}^1 2x_1((\frac{x_0+x_1}{2} - \frac{1}{2})^2) + \frac{1}{4}) dx_1 dx_0 \\
&= \frac{83}{360}
\end{aligned}
$$

$$p(error|B) = \frac{43}{360} \Rightarrow p(error) = \frac{7}{20}$$

Cross Validation We need: generalization error (on unseen, new data) p(error)
We have: training error/fit error

$$h_{TS} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[f(x_i) \neq y_i']$$

$$p_{error} = h_{TS} + w_{modeloptimism} \quad w \geq 0$$

if $p_{error} - h_{TS} = w$ is big, the model overfits the training set. Many models tend to overfit quite badly.
⇒ solutions:

- use more training data (expensive)
- use better models
- use regularization

e.q. nearest neighbor classifier $h_{TS} = 0$
how to estimate the error?

- split the training set at random in two subsets for training and test
- train on the training subset
- calculate the error on the test subset

⇒ since the choice of training and testset was arbitrary, reverse their roles and repeat and take the average of the two error (2-fold cross validation)
results are improved (error more reliable) by using more subsets 'K-fold cross validation'

- bring data into a random order (random-shuffle)
- put the first $\frac{N}{K}$ instances into fold 1
- put the second $\frac{N}{k}$ instances into fold 2
- repeat for $l = 1, ..., K$
- use all folds except fold l for training
- use fold l for testing
- compute means and variance of the K errors
- popular $K = 2, 5, 10$ K = N: 'leave-one-out-cross-validations' for theoretical analysis

Asymptotic Analysis

- find analytic formulas for how the method performs with infinite training data
- $N \to \infty$ (training data)
- Definition: A learning algorithm is called consistent if it converges to the optimal Bayes classifier as $N \to \infty$
- prove now: NN classifier is <u>not</u> consistent, but not too far of (a factor of 2): $p_{00}^{NN} \leq 2p*$
- let $p(error|x, x')$ be the expected error for test point x, when x' is its nearest training point
- let p(x|x') be the probabilty that x' is n.n. of test point

$$\begin{aligned} p(error|y) &= \int p(error|x, x') p(x'|x) dx' \quad \text{(marginalize over unknown point x')} \\ &= \mathbb{E}_{x'}[p(error|x, x')] \end{aligned}$$

1) If density p(x) is continous and positive:

$$\lim_{N\to\infty} p(x'|x) = \delta(x - x')$$

Let $p_\varepsilon(x)$ be the probability that an ε-ball around x:

$$B_\varepsilon(x) = \{x' | \|x - x'\| \leq \varepsilon\}$$

contains at least one training point. Then $(1 - p_\varepsilon)^N$ is the probabilty, that none of N training points is in $B_\varepsilon(x)$

$$\text{By assumption} \quad \forall \varepsilon > 0 \quad p_\varepsilon(x) = \int_{B_\varepsilon(x)} p(x') dx' > 0$$

$$\lim_{N\to\infty} (1 - p_\varepsilon(x))^N = 0 \Rightarrow \forall \varepsilon > 0 \quad \text{there is a point in } B_\varepsilon(x)$$

2)

$$\begin{aligned} p(error|x, x') &= 1 - p(correct|x, x') \\ &= 1 - \sum_{k=1}^{c} p(y = k, y' = k|x, x') \\ &= 1 - \sum_{k=1}^{c} p(y = k|x) p(y' = k|x') \quad \text{due to i.i.d.} \end{aligned}$$

3)

$$\begin{aligned} \text{Insert:} \quad p_\infty(error|x) &= \int \underbrace{p(error|x, x')}_{1-\sum_i^c} \underbrace{p(x'|x)}_{\delta(x-x')} dx' \\ &= 1 - \sum_{k=1}^{c} p(y = k|x)^2 \quad \text{Gini impurity at point x} \end{aligned}$$

- if data at point x are pure, i.e. only one class occurs, say $y = k^* \Rightarrow p(y = k^*|x) = 1$ and $p(y = k|x) = 0$ for $k \neq k^* \Rightarrow p_\infty(error|x) = 0$
- worst: data are impure, i.e. all classes gave same probability $p(y = k|x) = \frac{1}{c} \Rightarrow$

$$p_\infty(error|x) = 1 - \sum_k \frac{1}{c^2} = 1 - \frac{1}{c} = \frac{c-1}{c} \geq \frac{1}{2}$$

4) Derive worst case behavior aver all x as a function of Bayes error p^*

$$p_\infty(error) = \mathbb{E}_x[p_\infty(error|x)]$$

Let $p(y = \hat{k}|x)$ be the Bayes decision at x, $\hat{k} = \arg\max_k p(y = k|x)$
⇒ Bayes error at x:

$$p^*(error|x) = 1 - p(y = \hat{k}|x)$$

$$\sum_{k=1}^{c} p(y = k|x)^2 = (1 - p^*(error|x))^2 + \sum_{k=\hat{k}} p(y = k|x)^2$$

worst case analysis: make the error big, i.e. make this sum small

$$\text{Probability:} \quad \sum_k p_k^2 \quad \text{is minimized under constrains} \quad p_k \geq 0 \quad \text{and} \quad \sum_k p_k = const$$

$$if \quad p_k = p'_k \forall k, k' \quad p_k = p^*(error|x)$$

$$\Rightarrow p_k = \frac{p^*(error|x)}{c}$$

worst case error:

$$\sum_{k=1}^{c} p(y = k|x) \geq (1 - p^*(error|x))^2 + \sum_{k=k'} (\frac{p^*(error|x)}{c - \frac{1}{2}})^2$$

$$= 1 - 2p^*(error|x) + p^*(error|x) + \frac{p^*(error|x)^2}{c-1} \quad = 1 - 2p^*(error|x) + \frac{c}{c-1} p^*(error|x)^2$$

5) Inserting gives the relationship between error of NN classifier and Bayes classifier:

$$p_\infty(error|x) = 1 - \sum_{k}^{c} p(y = k|x) \leq 1 - (1 - 2p^*(error|x)) + \frac{c}{c-1} p^*(error|x)^2$$

$$= 2p^*(error|x) - \frac{c}{c-1} p^*(error|x)^2$$

6) Total error = expectation over x

$$\begin{aligned} p_\infty(error) &= \mathbb{E}_x[p_\infty(error|x)] = \int p_\infty(error|x)p(x)dx \\ &\leq \int 2p^*(error|x)p(x)dx - \int \frac{c}{c-1} p^*(error|x)^2 p(x)dx \\ &= 2\mathbb{E}_x[p^*(error|x)] - p^*(error) \\ &\leq 2p^*(error) - \frac{c}{c-1} p^*(error)^2 \end{aligned}$$

simplified by non neg. of variance:

$$\int (p^*(error|x) - p^*(error))^2 p(x)dx \geq 0$$

$$\leftrightarrow \int p^*(error|x)^2 p(x)dx \geq p^*(error)^2$$

$$\text{Result:} \quad p^*(error) \leq p_\infty^N N(error) \leq p^*(error)(2 - \frac{c}{c-1} p^*(error))$$

Special Cases:

- best case: $p^* = 0 \Rightarrow p^\infty \leq 0 \quad 0(2 - \frac{c}{c-1}0) = 0$ NN is perfect
- worst case: $p^* = \frac{c-1}{c}$ (pure guessing) $\Rightarrow$

$$\begin{aligned} p_\infty &< \frac{c-1}{c}(2 - \frac{c}{c-1}\frac{c-1}{c}) \\ &= \frac{c-1}{c} \end{aligned}$$

- normal case: Bayes classifier performs well, but not perfect:

$$\begin{aligned} p^* = \varepsilon \ll 1 \forall c \geq 2 : \frac{c}{c-1} \leq 2 \\ p_\infty \leq \varepsilon(2 - \underbrace{\frac{c}{c-1}}_{\ll 1}\varepsilon) \leq 2\varepsilon = 2p^* \end{aligned}$$

Advantages of NN-method:

- simple and intuitive
- often easy to implement
- performs elecently in practice

1.5.2 Limitations of Nearest Neighbor Classifier

1. NN is not consistent: $p_\infty(error) \leq 2p^*(error)$ (consistent: $p_\infty(error) = p^*(error)$)
solution: K-nearest neighbor algorithm:

- find the k nearest neighbors
- take majority vote
- is consistent if $k(N)$ such that

$$\lim_{N\to\infty} k(N) = \infty, \quad \lim_{N\to\infty} \frac{k(N)}{N} = 0$$

- e.g. k(N) $\log N$

2. nearest neighbor search is expensive: naive algorithm $\mathcal{O}(D * N)$ D: feature dimension, N: instances
solutions:

- reduce D:
  - dimension reduction(later, ch.'unsupervised learning')
  - relevant feature selection
- reduce N, relevant instance selection e.g.:
  - sort TS randomly
  - memorize the next instance only if the memorized set so far classifies incorrect

exactly:

- compute Voronoi tesselation

- drop all instances whose neighbors all have the same class
- clustering: find groups of similar instances ('clusters') which can be replaced by simple representative (later in chapter 'unsupervised learning')

use an efficient search algorithm: D-dimensional search trees('k-d tree, x tree, R^H tree')
use approximate n.n. search: find a near neighbor fast and with high probability of being correct ⇒ several ANN libraries in the internet

3. nearest neighbor selection depends on the distance function $d(x, x')$. How to define a 'good' $d()$?
Depending on units, different neighbor might minimize $d(x, x')$ solution. Therefor use dimensionless features, i.e. standardize data. Divide each feature by its TS standard deviation [actually, also substract means - 'centralization']
Better solution: learn the metric from the TS (or from additional TS with 'is similar'/'is not similar')-labels.
⇒ research area: 'metric learning'
Much of the success of neural networks is their ability to implicilly find a good set of intermediate features and metric.

1.6 Quadratic and Linear Discriminant Analysis

1.6.1 Motivation

In nearest neighbors, we can reduce search time by reducing N ⇒ extreme case: One representatice per label.
Obvious choice is the mean of each class:

$$\forall k \in 1, ..., c: \quad \mu_k = \frac{1}{N_k} \sum_{i:y_i=k} x_i \quad N_k = \text{instances in class k}$$

This works well, if clusters are roughly circular.
To find the desired decision bound, we neet to consider the actual shape of the class ⇒ correction for non-circularity
Simplest generalization: approximate cluster shape by an ellipse instead of circle (higher dimension: ellipsoid)
⇒ Natural choice: multi-dimensional Gaussian distribution

1.6.2 QDA

assumptions:

- each class prior $p(y = k)$ is well approcimatet by the TS proportion $\hat{p}(y = k) = \frac{N_k}{N} = \pi_k$
- the data likelihood for each class p(x|y=k) is well approximated by a Gaussian distribution

⇒ generative model:

$$p(y = k|x) = \frac{p(x|y = k)p(y = k)}{p(x)}$$

$$p(x) = \sum_{k=1}^{c} p(x|y = k)p(y = k)$$

Gaussian distribution

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{x-\mu}{2\sigma^2}}$$

generalization of σ^2: Σ covariance matrix D x D, symmetric, positive definite

$$\Sigma = \frac{1}{N} \sum_i (x_i - \mu)(x_i - \mu)^T$$

$$\text{for example:} \quad \Sigma = R^{-1} \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix} R$$

principle axes of ellipse: directin of largest and smallest width $\leftrightarrow$ eigenvectors of covariance matrix Σ

$$\text{multivariable Gaussian} \quad p(x; \vec{\mu}, \Sigma) = \frac{1}{\sqrt{det(2\pi\Sigma)}} e^{-\frac{1}{2}(x-\vec{\mu})^T \Sigma^{-1} (x-\vec{\mu})}$$

To find $p(x|y = k)$, we must define $\underbrace{\vec{\mu_k}}_{\text{cluster pos.}}$, $\underbrace{\Sigma_k}_{clustershape}$ for each k:

How to fit a multi-dim. Gaussian Let $\{x_i\}_{i=1}^N$ be the instance of just a single class (drop index k for clarity).

$$p(x) = \text{multi-dim. Gaussian}$$

Fit according to maximum likelihood principle assumed that TS is typical for true (unknown) distribution.
$\Rightarrow$ we want the TS to be typical for our model as well, when simulating our model, the TS should occur with high (maximum) probability.

$$\begin{aligned} p(TS) &= p(x_1, ..., x_N) \Rightarrow \text{maximized under our model} \\ &\overset{i.i.d.}{=} p(x_1; \mu, \Sigma) p(x_2; \mu, \Sigma) ... p(x_N, \mu, \Sigma) \end{aligned}$$

$$\log(p(x_1, ..., x_N)) = \sum_{i=1}^N \log(p(x_i; \mu, \Sigma))$$

To maximize, take derivatives with respect to parameter set to 0
$\Rightarrow$ system of equations, solve to find $\hat{\mu}, \hat{\Sigma}$

$$\frac{d \sum_i \log(x_i; \mu, \Sigma)}{d\mu} \overset{!}{=} 0 \quad \frac{d \sum_i \log(x_i; \mu, \Sigma)}{d\Sigma} \overset{!}{=} 0$$

How to fit a Gaussian? maximize likelihood of TS:

$$\begin{aligned} \hat{\Theta} &= \arg\max \log p(x_1, ...x_n | \Theta) \\ &\overset{i.i.d.}{=} \arg\max \sum_{i=1}^N \log p(x_1, ...x_n | \Theta) \end{aligned}$$

$$p(x_i; \Theta) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} e^{-\frac{1}{2}(x_i-\mu)^T \Sigma^{-1} (x_i-\mu)}$$

Define: $K = \Sigma^{-1}$ 'precession matrix'

$$\text{Lin. Algebra:} \quad \det(a * A) = a^D \det A \quad \det(A^{-1}) = \frac{1}{\det(A)} \quad \text{a: Scalar, A: D*D Matrix}$$

$$\Rightarrow \frac{1}{\sqrt{\det(2\pi\Sigma)}} = (2\pi)^{-\frac{D}{2}} \frac{1}{\sqrt{\det(K^{-1})}} = (2\pi)^{-\frac{D}{2}} \det(K)^{\frac{1}{2}}$$

$$\log p(x_i; \mu, K) = \sum_{i=1}^{N} -\frac{D}{2} \log 2\pi + \frac{1}{2} \log\det(K) - \frac{1}{2}(x_i - \mu)^T K (x_i - \mu)$$

$$\text{find } \mu \text{ by} \quad \frac{\partial}{\partial\mu} \log p(x_i, ...x_n) \overset{!}{=} 0$$

$$\iff \sum_{i=1}^{N} -K(x_i - \mu) = 0$$

$$\iff \sum_{i=1}^{N} (x_i - \mu) = 0$$

$$\iff \sum_{i=1}^{N} x_i = \sum_{i=1}^{N} \mu = N\mu$$

$$\implies \mu = \frac{1}{N} \sum_{i=1}^{N} x_i \quad \text{empirical mean of TS}$$

$$\text{Lin. Algebra:} \quad \frac{\partial}{\partial v} v^T A v = 2Av$$

$$\text{find K:} \quad \frac{\partial}{\partial K} \log p(x_1, ..., x_N) \overset{!}{=} 0$$

$$\iff \sum_{i=1}^{N} (\frac{1}{2} K^{-1} - \frac{1}{2} z_i z_i^T) = 0 \quad \text{centered coordinate } z_i = x_i - \mu$$

$$\text{Matrix calculus:} \quad \frac{\partial}{\partial K} \log\det(K^T)^{-1} = (K^T)^{-1} = K^{-1} = \Sigma$$

$$\frac{\partial}{\partial A} v^T A v = vv^T$$

$$N\Sigma = \sum_{i=1}^{N} \Sigma = \sum_{i=1}^{N} \underbrace{(x_i - \mu)(x_i - \mu)^T}_{scattermatrix}$$

$$\Sigma = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})(x_i - \hat{\mu})^T \quad \text{sample/empirical covariance matrix}$$

Training of QDA Repeat this for every class to get $\mu_1, \Sigma_1, ..., \mu_c, \Sigma_c$
QDA prediction: -generative model:

$$\begin{aligned}
\hat{k} &= \arg\max_k p(y=k]x) \\
&= \arg\max_k \frac{p(x|y=k)p(y=k)}{p(x)} \\
&= \arg\max_k \underbrace{p(x|y=k)}_{Gauss(x,\mu_k,\Sigma_k)} \underbrace{p(y=k)}_{\pi_k=\frac{N_k}{N}} \\
&= \arg\min_k -\log p(y=k|x) \\
&= \arg\min_k \frac{D}{2}\log 2\pi + \frac{1}{2}\log\det\Sigma_k + \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \log\pi_k \\
&= \arg\min_k \cancel{\frac{1}{2}}\underbrace{\log\det\Sigma_k - 2\log\pi_k}_{=b_k} + \cancel{\frac{1}{2}}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) \\
&= \arg\min_k \underbrace{(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}_{\text{squared Mambalanbis distance btw. } x \text{ and } \mu_k} + b_k
\end{aligned}$$

Euclidean: $\Sigma = \mathbb{1}$ $\sigma \neq \mathbb{1}$ adjust for elliptic cluster sphere.

Define square root of matrix $A^{\frac{1}{2}} \iff A = (A^{\frac{1}{2}})^T(A^{\frac{1}{2}})$

Find Σ^{-1} by decomposition $z_k = \Sigma_k^{-\frac{1}{2}}(x-\mu_k)$

$$\Rightarrow z_k^T z_k = (x-\mu_k)^T \frac{\Sigma_k^{-\frac{1}{2}T}\Sigma_k^{-\frac{1}{2}}}{\Sigma_k^{-\frac{1}{2}}}(x-\mu_k)$$

$\Rightarrow$ can use standard nordmed of z_k

1.6.3 LDA Linear Discriminant Analysis

simplifications: assume that clusters have the same size elliptic shape

$$\forall k, k' \quad \Sigma_k = \Sigma_{k'} = \Sigma$$

$$QDA: \quad \hat{k} = \arg\min_k (x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + b_k$$

$$\begin{aligned}
LDA: \quad \hat{k} &= \arg\min_k (x-\mu_k)^T\Sigma^{-1}(x-\mu_k) \\
&= \arg\min_k x^T\Sigma^{-1}x - \Sigma\mu_k^T\Sigma^{-1}x + \mu_k^T\Sigma^{-1}\mu_k \\
&= \arg\min_k -\Sigma\mu_k^T\Sigma^{-1}x + \mu_k^T\Sigma^{-1}\mu_k + b_k \\
&= \arg\min_k w_k^T x + b'_k
\end{aligned}$$

New variables: $w_k^T = \Sigma\mu_k^T\Sigma^{-1}$ $b'_k = -\mu_k^T\Sigma^{-1}\mu_k - b_k$
LDA is a particular way to define w_k and b'_k. There are many other possible ways, e.g. logistic regression.

Special case: C=2

$$\hat{k} = \begin{cases} 0 & \text{if} \quad w_0^T x + b_0' > w_1^T x + b_1' \\ 1 & \text{if} \quad w_0^T x + b_0' < w_1^T x + b_1' \end{cases}$$

$$= \begin{cases} 0 & \text{if} \quad (w_0^T - w_1^T)x + b_0' - b_1' > 0 \\ 1 & \text{otherwise} \end{cases}$$

$$= \begin{cases} 0 & \text{if} \quad w^T x + b' > 0 \\ 1 & \text{if} \quad w^T x + b' < 0 \end{cases}$$

Define: $w = w_1 - w_0 \quad b' = b_1' - b_0'$
$a = w^T x$ is a 1D projection of x onto the vector w
⇒ apply the mean classifier to the a value

1.6.4 LDA

Two classes C = 2

$$\hat{y} = \begin{cases} 1 & \text{if } w^T x + b \geq 0 \\ 0 & else \end{cases}$$

Three different derivations of how to choose the best w and b:

1. our derivation: LDA is the same as QDA with all classes having the same cluster shape
⇒ we just fit a Gaussian distribution to within - class covariance

$$\hat{\Sigma} = \frac{1}{N}\Sigma_{i=1}^{N}(x_i - \mu_{k=y_i})(x_i - \mu_{y_i})^T$$

$$w^T = 2(\mu_1 - \mu_0)\hat{\Sigma}^{-1}$$

2. R. Fishers original derivation: we seek the optimal 1-dimensional projection of D-dimensional data

- compute μ_0, μ_1, Σ
- define 1-D projection $z_1 = w^T x_i$ ⇒ projected means $m_k = w^T \mu_k$, 1-D variance $\sigma^2 = w^T \Sigma$
- determine w such that the z_i are seperated as good as possible

$$m_0 - m_1 = w^T(\mu_1 - \mu_0)$$

- Fisher criterion: Choose w that maximizes $\frac{(m_1 - m_0)^2}{\sigma^2}$

3. derivation via least-squares regression
define class labels $Y \in \{-1, 1\}$
LSQ: find w, b that minimzes:

$$\sum_{i=1}^{N}(w^T x_i + b_i - y_i)^2$$

if classes are balanced: $N_{-1} = N_{+1}$
⇒ optimal solution is again the same ⇒ proof: home-work

1.7 Logistic Regression LR

- Not really regression, because it's a classifier, term is partially justified because LR predicts class probabilities. It actually computs the posterior $p(Y|X)$
- generative model (LDA) vs. discriminative model (LR)
- LDA:
  - define RHS of Bayes theorem (likelihood and prior)
  - learn the parameters of RHS (fit a Gaussian for every class)
  - apply Bayes to compute the posterior
- LR
  - define RHS of Bayes
  - apply Bayes to compute posterior (LHS)
  - learn parameters of the LHS
  - or: merge first two steps and define LHS model directly (e.g. NN)

derive LR from LDA: 2 classes C=2, equal priors $p(y=0)=p(y=1)=\frac{1}{2}$, cluster shape (e.g. covariance Σ of both classes equal)

$$\begin{aligned}
p(y=1|x) &= \frac{p(x|y=1)\cancel{p(y=1)}}{p(x|y=0)\cancel{p(y=0)}+p(y|y=1)\cancel{p(y=1)}} \\
&= \frac{p(x|y=1)}{p(x]y=0)+p(x|y=1)} \\
&= \frac{p(x|y=1)}{p(x|y=1)} * \frac{1}{\frac{p(x|y=0)}{p(x|y=1)}+1}
\end{aligned}$$

Gaussian likelihoods:

$$p(x|y=0) = \frac{1}{\sqrt{det(2\pi\Sigma)}} e^{-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)}$$

$$p(x|y=1) = \frac{1}{\sqrt{det(2\pi\Sigma)}} e^{-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)}$$

$$\mu = p(y=0)\mu_0 + p(y=1)\mu_1 = \frac{\mu_0+\mu_1}{2} = 0$$

assume that dara are centered $\Rightarrow \mu = 0$

$$\begin{aligned}
p(y=1|x) &= \frac{1}{1+\frac{p(x|y=0)}{p(x|y=1)}} \\
&= \frac{1}{1+\exp(-0.5[(x_{\mu 1})^T\Sigma^{-1}(x+\mu_1)-(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)])} \\
&= \frac{1}{1+exp(-0.5[4\mu_1^T\Sigma^{-1}x])} \\
&= \frac{1}{1+exp(-2\mu_1^T\Sigma^{-1}x)} \\
&= \frac{1}{1+exp(-w^Tx)}
\end{aligned}$$

$$w^T = 2\mu_1^T \Sigma^{-1}$$

$$p(y=0|x) = 1 - p(y=1|x)$$

$$\underline{Decisionrule}: \quad \hat{y} \begin{cases} 1 & if \quad p(y=1|x) \geq p(y=0|x) = 1 - p(y=1|x) \\ 0 & otherwise \end{cases}$$

$$\text{logistic sigmoid function :} \quad \sigma(t) = \frac{1}{1 + exp(-t)}$$

Properties:

$$\sigma(-t) = 1 - \sigma(t)$$

$$\sigma(-t) = \frac{1}{1+exp(t)} = 1 - \frac{1}{1+exp(-t)} = \frac{exp(-t)}{1+exp(-t)} = \frac{1}{exp(t)+1}$$

Derivative:

$$\frac{d}{dt}\sigma(t) = \frac{d}{dt}(1+exp(-t))^{-1} = (1+exp(-t))^{-2}exp(-t) = \sigma(t)\sigma(-t) = \sigma(t)(1-\sigma(t))$$

1.7.1 Learning LR

- maximum likelihood principle: choose the model such that the training data are a typical realization means:

$$\begin{aligned} \hat{w} &= \arg\max_w p((x_1, y_1), ..., (x_n, y_n)|w) \\ &= \arg\min_w - \log p((x_i, y_i)|w) \\ &\overset{i.i.d.}{=} \arg\min_w -\Sigma_{i=1}^N \log p(y = y_i|x_i, w) \\ &= \arg\min_w -[\Sigma_{i,y_i=1} \log p(y=1|x_i, w) + \Sigma_{i,y_i=0} \log(1 - p(y=1|x_i, w))] \\ &= \arg\min_w = -\Sigma_{i=1}^N [y_i \log p(y=1|x_i, w) + (1-y_i)\log(1 - p(y=1|x_i, w))] \end{aligned}$$

Find w

$$\begin{aligned} \frac{\partial}{\partial w} - \sum_{i=1}^N ... &= -\sum_{i=1}^N [y_i \frac{1}{\sigma(w^T x_i)} \sigma(w^t x_i)\sigma(-w^T x_i)x_i + (1-y_i)\frac{1}{1-\sigma(w^T x_i)}(-1)\sigma(w^T x_i)\sigma(-w^T x_i)x_i] \\ &= -\sum_{i=1}^N (y_i \underbrace{\sigma(-w^T x_i)}_{1-\sigma(w^T x_i)} x_i - (1-y_i)\sigma(w^T x_i)x_i) \\ &= -\sum_{i=1}^N y_i x_i - y_i\sigma(w^T x_i)x_i - \sigma(w^T x_i)x_i + y_i\sigma(w^T x_i)x_i \\ &= \sum_{i=1} (y_i - \sigma(w^T x_i))x_i \overset{!}{=} 0 \end{aligned}$$

no analytical solution ⇒ need to solve numerically
Numerical algorithms:

- classical: few training data $N \leq 1000$ ⇒ use Newton-Raphson algorithm ⇒ Iterative Reweighted Least Squares (RLS ⇒ later)
  - advantage: needs few iterations

  - drawback: each iteration is expensive when N gets bigger $\mathcal{O}(N^3)$ or $\mathcal{O}(N^2)$ with tricks

- modern: lots of training data: stochastic gradient descent
  - choose initial guess for $w^{(0)}$ e.g. $w^{(0)} = 0 \Rightarrow \sigma(w^T x) = \frac{1}{2} \quad \forall x \Rightarrow p(y = 1|x) = p(y = 0|x)$
  - for t = 1, ..., T (or until convergence)
    - bring TS into random order (e.g. random shuffle) of indices
    - for i = 1,..., N: $w' = w - \tau(y_i - \sigma(w^T x_i)x_i), \quad \tau$: learning rate
    - reduce learning rate $\tau \leftarrow \frac{t}{t-1}\tau$

1.8 Histogramms and Density Trees

1.8.1 Introduction

- we had: generative models vs. discriminative models (learn LHS or RHS of Bayes)
- new distinction:
  - parametric models: choose probabilities from a family with analytic formula (Gaussian) and we learn its parameters (Gaussian: μ, Σ)
  - non-parametric models: don't restrict the probability - 'universal model' (neural net) and learn many more parameters (network weights)

	generative	discriminative
parametric	LDA and QDA	LR
non-parametric	histogramm, density tree	nearest neighbors

- histogramm: count the frequency of random events:

$$\frac{\sum_{i=1} \mathbb{1}[x_i = z]}{N}$$

z: events considered and create table for all events

- x is discrete, throwing dice: $z \in \{1, ..., 6\}$

$$\text{hist}(z) = \frac{[x = z]}{N}$$

- x is continous $x \in R \Rightarrow$ discretize into <u>bins</u>

$$b_l = \{x| \underbrace{x_{min} + l\Delta x}_{x_l} \leq x \leq \underbrace{x_{min} + (l+1)\Delta x}_{x_{l+1}}\}$$

- find bin index of x:

$$l = \left\lfloor \frac{x - x_{min}}{\Delta x} \right\rfloor \quad \Delta x: \text{ bin width} \quad \lfloor\rfloor: \text{ floor function}$$

$$\mathbb{1}[a] = \begin{cases} 1 & \text{if a is true} \\ 0 & \text{else} \end{cases}$$

p_l= prob for X in bin l

- approx likelihood:
$$p(x|y) \approx \sum_l p(l)\mathbb{1}\lfloor x \in b_l \rfloor$$

- meaning: piecewise constant approximation ⇒ can make error arbitrarily small by choosing more bins, but not more than the TS allows

- optimal approx, given TS and Δx

$$\hat{p} = \arg\min_p Error = \int \underbrace{(p^*(x) - p(x))^2 dx}_{\cancel{p^*(x)^2} - 2p^*(x)p(x) + p(x)^2} \qquad p^*: \text{ truth} \quad \hat{p}: \text{ best approx.}$$

$$\begin{aligned}
-2^*(x)p(x)dx &= -2\int p^*(x)\sum_l p(l)\mathbb{1}[x \in b_l]dx \\
&= -2\sum_l p(l)\int p^*(x)\mathbb{1}[x \in b_l]dx \\
&= -2\sum_l p(l)\int_{x_l}^{x_{l+1}} p^*(x)dx \\
&= -2\sum_l p(l)\mathbb{E}_{p^*(x)}[\mathbb{1}[x \in b_l]]
\end{aligned}$$

$$\mathbb{E}_{p^*(x)}[\mathbb{1}(x \in b_l)] \approx \frac{N_l}{N} \quad {}_l = (x \text{ is in } l)$$

$$\mathbb{1}[x \in b_l]^2 = \mathbb{1}[x \in b_l] \quad l \neq l': \quad \mathbb{1}[x \in b_l]\mathbb{1}[x \in b_{l'}] = 0$$

$$\begin{aligned}
\int p(x)^2 dx &= \int (\sum_l p_l \nVdash [x \in b_l])^2 \\
&= \int \sum_l p_l^2 \mathbb{1}[x \in b_l]dx \\
&= \sum_l p_l^2 \int \mathbb{1}[x \in b_l]dx \\
&= \sum_l p_l^2 \int_{x_l}^{x_{l+1}} 1dx \\
&= \sum_l p_l^2 \Delta x
\end{aligned}$$

$$Error =^* (x)^2 dx - 2\sum_l p_l \frac{N_l}{N} + \sum_l p_l^2 \Delta x$$

$$\hat{p_l} = \arg\min \text{Error} = \arg\max_{p_l} \sum_l p_l^2 \Delta x - 2\sum_l p_l \frac{N_l}{N}$$

$$\frac{\partial}{\partial p_l} ... = 2p_l\Delta x - 2\frac{N_l}{N} \overset{!}{=} 0$$

$$p_l = \frac{N_l}{N\Delta x} \quad \text{probability density estimate}$$

insert into error:

$$\begin{aligned}
Error &= \int p^*(x)^2 dx - 2\sum_l \frac{N_l}{N\Delta x}\frac{N_l}{N} + \sum_l (\frac{N_l}{N\Delta x})^2 \Delta x \\
&= \int p^*(x)^2 dx - \sum_l (\frac{N_l}{N})^2 \frac{1}{\Delta x} \\
&= \int p^*(x)^2 dx - \sum_l \hat{p_l}^2 \Delta x
\end{aligned}$$

- how to choose Δx: Difficult, rules of thumb:
  - Scotts's rule:

    $$\Delta x = \frac{3.5\sigma}{\sqrt[3]{N}} \quad \text{exactly optimal if } p^* \text{ is Gaussian)}$$

  - Freedman-Diaconis rule:

    $$\Delta x = \frac{2\text{IQR}(x)}{\sqrt[3]{N}}$$

    IQR: inter-quartile range: place data in sorted order: $x_{[1]}, x_{[2]}, ..., x_{[N]}$

    $$IQR = x_{[\frac{3}{4}N]} - x_{[\frac{1}{4}N]}$$

  - Shimazaki/Shinomoto rule:

    $$\hat{\Delta x} = \arg\max_{\Delta x} \frac{2m - v}{(\Delta x)^2} \quad m\text{: mean bin count, } v\text{: variance}$$

  - cross-validation: split into training sets of size N and test sets of size M
    for each candidate δx compute error

    $$\sum_l \frac{1}{\Delta x}(\frac{N_l}{N} - \frac{M_l}{M})^2$$

    and choose Δx that minimizes error
    in general: if φ is a hyperparameter (here: Δx): use Cross Validation and grid search

- typical bin counts (e.g. Freedman-Diaconis): scale x such that IQR $(x) = \frac{1}{2}$

  $$\Rightarrow \Delta x = \frac{1}{\sqrt[3]{N}}$$

  if data are uniformely distributed:

  $$x_{max} - x_{min} = 2\text{IQR}x = 1$$

  $$\text{bin count} \quad \frac{x_{max} - x_{min}}{\Delta x} = \frac{1}{\Delta x} = \sqrt[3]{N}$$

  $$N = 1000 \Rightarrow 10 \text{ bins} \quad N = 10^6 \Rightarrow 100 \text{ bins}$$

- generalize to the multi-dimensional case $x \in \mathbb{R}^D$
  naive solution: split each dimension according to Freedman/Diaconis
  ⇒ 10 bins per dimension ⇒ 10^D bins in total
  ⇒ no TS can ever fill 10^D bins ⇒ most are empty ⇒ no estimate
  ⇒ doesn't work

- use a 1-dimensional histogram for each dimension (only 10*D bins)
- only use 1-D histogramms, one per feature per class
- probabalistic interpretation: if we know the class y, all fearture dimension are statistically independent

$$p(x|y) = p(\{x_j\}|y) = p(x_{j=1}|y)p(x_{j=2}|y)p(x_{j=D}|y)$$

- density trees: place bins adaptively: many bins in subregions with many data bins, few bins in subregions with few data points

1.8.2 Naive Bayes

Using one histogramm per dimension $\iff$ assumption that values of different features are independent, given the class $\iff$ if we know the class label y_i, then knowing the feature x_{i1} doesn't tell us anything about other feature values $x_{ij} \quad j \neq 1$
This assumption is often violated, e.g. in images. Consider an image region with class label 'sky'. Let one pixel in the sky have color 'blue'. Than it's likely that the neighbor pixels are probably also 'blue'. The same applies to other colors (e.g. grey for clouds or red for sunset). If assumption is true, joint probability

$$p(x_i = [x_{i1}, ..., x_{iD}]|y_i) = \Pi_{j=1}^{D} p_j(x_{ij}|y_i)$$

$$\text{Bayes:} \quad p(y_i|x_i) = \frac{p(x_i|y_i)p(y_i)}{p(x_i)} \overset{\text{naive}}{=} \frac{\Pi_{j=1}^{D} p_j(x_{ij}|y_i)p(y_i)}{p(x_i)}$$

$$\text{simplify} \quad p(y_j) = p(y_{ji}) = \frac{1}{C} \quad \text{uniform priors}$$

$$\begin{aligned}\text{decision rule:} \quad \hat{y_i} &= \arg\max_k p(y_i = k|x_i) \\ &= \arg\max_k \Pi_{j=1}^{D} p_j(x_{ij}|y_i = k) \\ &= \arg\max_k \sum_{j=1}^{D} \log p_j(x_{ij}|y_i = k)\end{aligned}$$

training: for $k = 1, ...C$ for $j = 1, ..., D$: fit 1-D histogram $p_j(x_{ij}|y_i = k)$ using the j-th feature of all instances of class k

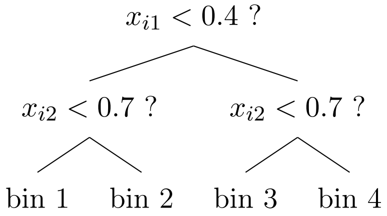
Variant: Instead of 1-D histogramms, fit 1-D Gaussian distributions $\Rightarrow$ naive BAyes becomes equivalent to QDA with constraint that covariance matrices Σ_k are all diagonal $\iff$ axes of ellipses are parallel to coordinate axes.

1.8.3 Density trees

- idea: place bins adaptively, small bins where there are many datapoints, big bins where there are few
- how to find the bin for a data point efficently:

$$1 - D \quad l = \left\lfloor \frac{x - x_{min}}{\Delta x} \right\rfloor$$

- higher dimensions: represent binning by a binary tree



- bin index for L bins can be found with $\mathcal{O}(\log L)$ decisions (if tree balanced) - 'recursive subdivision'
- training
  - build the tree
  - define response ($\hat{p_l}$) of the leaves, e.g. as in 1-D histograms

$$\hat{p_l} = \frac{N_l}{NV_l} \quad N_l \text{ instances in bin l, } V_l \text{ volume of bin l}$$

or: fit Gaussian to each bin (adjust normalization for finite bin size)

Build tree by 'recursive best-first expansion'
Init:

- put all data into a single bin (root of tree), size: bounding rectangle of data points
- fix bins $L = \tau \sqrt[3]{N}$
- for $t = 1, ..., L - 1$
  - compute 'score' of all current leave modes
  - split best leaf into two children (original leaf is now an interior node)

score of a leaf: maximal improvement of our objective funtion (e.g. error) if we would split this leaf)

- Criminisi et al.: try a number of random splits and remember the best (allow oblique splits)
- exhaustive search for best split (preferable when we split axis orthogonal)

exhaustive search: give leaf with boundaries $x_j \in [m_j, M_j]$, $N_l =$ instances in this leaf
for each feature $j \in 1, ..., D$:

- define candidate split thresholds

$$s_j = \{s_{ja}, ..., s_{j(N_l+1)}\} : \quad \text{sort data according to feature j}$$

$$x_{[0]j} = m_j = x_{[1]j} \leq x_{[2]j} \leq ... \leq x_{[N_l]j} \leq M_j = X_{[N_l+1]j}$$

  place candidate threshold in the middle of each pair

- compute the score of every candidate split return dimension j and threshold s_{ja} and score g_{ja} of best candidate split (among $D(N_l + 1)$)

scores:

- minimize squared error of histogram:

$$\text{error} = \int p^*(x)^2 dx - \sum_{l}^{L_t} \hat{p_l}^2 V_l \quad \text{before split}$$

$$\text{error'} = \int p^*(x)^2 dx - \sum_{l=1}^{L_t+1} \hat{p_l'}^2 V_l$$

  suppose split leaf l into λ and ρ

$$\text{gain } g = \text{error} - \text{error'} = -\hat{p_l}^2 V_l + \hat{p_\lambda}^2 V_\lambda + \hat{p_\rho}^2 V_\rho$$

- split nodes where data distribution is far from uniform $\iff \frac{N_l}{N} \sim V_l$

$$\text{non-uniformity : } \left| \frac{N_\lambda}{N_l} - \frac{V_\lambda}{V_l} \right|$$

- split to minimize etropy (ex. fit Gaussian)

$$H = \frac{1}{2} \log(\det(2\pi e \Sigma))$$

$$g = H_l - \frac{N_\lambda}{N} H_\lambda - \frac{N_\rho}{N} H_\rho$$

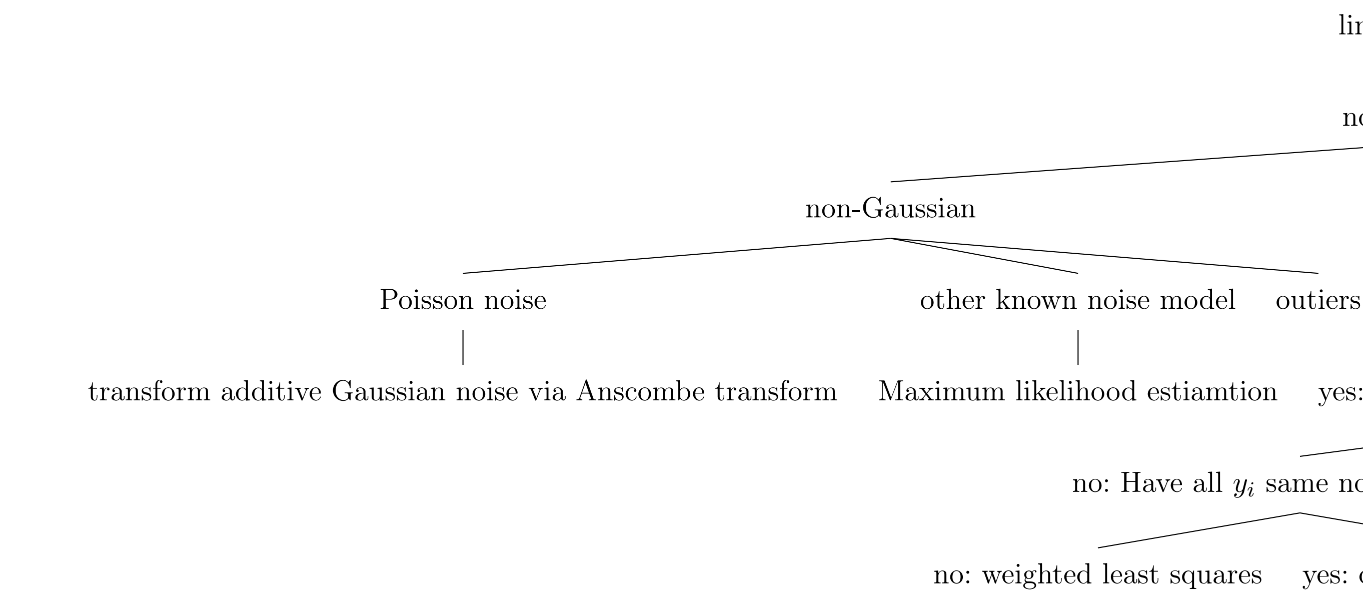
density trees tend to overfit:

- traditional: pruning $\widehat{=}$ cut-off subtrees with high overfitting and replace by single leaf
- modern: density forest $\widehat{=}$ train many trees and take their average probability ('ensemble method' ⇒ later)
  how to make the trees different:
  1. train every tree on a random subset of the training data (bootstrap sampling)
  2. consider a random subset of the features inthe split: search algorithm

2 Regression

2.1 Introduction

- Learn model $y = f(x) : X \in \mathbb{R}^D \Rightarrow y \in \mathbb{R}^{D'} (D' = 1)$
- training set $\{(\underbrace{x_i}_{\text{D-dim features}}, \underbrace{y_i}_{\text{real-valued true response}})\}_{i=1}^N$ assume i.i.d. - matrix notation


lin
nc
non-Gaussian
Poisson noise
other known noise model
outiers
transform additive Gaussian noise via Anscombe transform
Maximum likelihood estiamtion
yes:
no: Have all y_i same nc
no: weighted least squares
yes: c

2.2 Ordinary Least Squares (OLS)

- linear model with additive Gaussian noise, X is noise-free, all Y have same noise variance

$$Y = \underbrace{X}_{\text{row vector D-dim features}} \underbrace{\beta}_{columnvectorofD-dimweights/activations} + \underbrace{\varepsilon}_{Gaussiandistributedscalar}$$

$$\varepsilon \sim N(0, \sigma^2) \iff y \sim N(X\beta, \sigma^2)$$

- find best $\hat{\beta}$ via Maximum Likelihood principle $\hat{=}$ make training set typical under the model
  compute the residuals
  $$r_i = y_i - x_i\beta$$
  If the model is correct ($\hat{\beta} = \beta^*$): $r_i \sim N(0, \sigma^2)$
  If model is correct and we know the truth $\beta = \beta^* \to r_i = \varepsilon_i$, we only have $\beta = \hat{\beta} \to r_i \sim \varepsilon_i$

- ML principle

$$\begin{aligned}\hat{\beta} &= \arg\max_{\beta} p(\{y_1, ..., y_N\}|\{x_1, ..., x_N\}; \beta) \\ &= \arg\max_{\beta} \prod_{i=1}^{N} \underbrace{p(y_i|x_i;\beta)}_{y * e^{\frac{-(y_i - x_i\beta)^2}{2\sigma^2}}} \\ &= \arg\min_{\beta} - \sum_{i=1}^{N} \log p(y_i|x_i;\beta) \\ &= \arg\min_{\beta} \sum_{i=1}^{N} \frac{(y_i - x_i\beta)^2}{2\sigma^2} - N \log v \\ &= \arg\min_{\beta} \sum_{i=1}^{N} (y_i - x_i\beta)^2 \quad \text{least-squares objective}\end{aligned}$$

- example: fit a line in 2D $X \in \mathbb{R}$

$$Y = [X; 1][a; b]^T + \varepsilon$$

$$Y = aX + b + \varepsilon$$

$$\hat{a}, \hat{b} = \arg\min_{a,b} \underbrace{\sum_{i=1}^{N} (Y_i - aX_i - b)^2}_{\text{Loss}}$$

$$\frac{\partial \text{Loss}}{\partial b} = \sum \sum_{i=1}^{N} (Y_i - aX_i - b)(-1) \overset{!}{=} 0$$

$$\sum_i Y_i - a \sum_i X_- \sum_i b = 0$$

$$\frac{1}{N} \sum_i Y_i = a \frac{1}{N} \sum_i X_i + b$$

$$\bar{Y} = a\bar{X} + b$$

Regressionline always goes through the origin ⇒ always center data(i.e. $\bar{X} = 0, \bar{Y} = 0$)

  * regression goes through origin ⇒ no need for intercept b
  * numbers get smaller ⇒ regression is numerically more stable

⇒ assume $X \Rightarrow X - \bar{X}, Y \Rightarrow Y - \bar{Y}$

rewrite objective in matrix form

$$\hat{\beta} = \arg\min_{\beta} (Y - X\beta)^T (Y - X\beta)$$

$$\frac{\partial \text{Loss}}{\partial \beta} = 2X^T (Y - X\beta) \overset{!}{=} 0$$

$$\underbrace{X^T X}_{\text{scatter matrix}} \quad \beta = X^T Y \quad \text{linear system of equations 'normal equations'}$$

⇒ solve for β
possibilities to solve:

1. formal solution:

$$\underbrace{(X^TX)^{-1}(X^TX)}_{\mathbb{1}}\beta = (X^TX)^{-1}X^TY$$

   $(X^TX)^{-1}$ exists if X has full rank description

$$\hat{\beta} = \underbrace{(X^TX)^{-1}X^T}_{X^+}Y$$

   $X^+ = (X^TX)^{-1}X^T$: Moore-Penrose pseudo-inverse, inverse to rectangular matrices

2. Cholesky factorization: X^TX is positive definite symmetric
   for every such matrix, there is a decomposition:

$$R^TR = X^TX \quad R : DxD \text{ upper triangular}$$