Machine Learning Summary

Nicolò Brandizzi

February 7, 2019

1 Intro

1.1 Intro

ML is producing knowledge from data, learning a function $f : X \rightarrow Y$ that takes some input $x \in X$ and outputs $y \in Y$.
Learning f means finding some approximation $f' \approx f$ so that the returned value $y' \simeq y$ is the closes to y as possible.

Un/Supervised When we have an output y for each sample y for the dataset $D = \{(x_i, y_i)\}$ we have **supervised** learning. When we do not have y_i then we have **unsupervised**.

Reinforcement Learning Having a triple of elements $D : (s_i, a_i, r_i)$ (state, action, reward), RL is the practice of learning a policy π in order to maximize the total discounted reward:

$$\pi : argmax(r_0 + \gamma r_1 + \gamma^2 r_2 + ... + \gamma^n r_n)$$

Where γ is some discount to give more/less weight to current reward on behalf of the future ones.

Hypothesis Given a target function $\phi(x)$ that we want to learn and a set of approximations $H : \{h_1, h_2, ..., h_n\}$ of which $h*$ is the best according to some measure, we have that $h * (x_i)$ is the best approximation of y_i.
$h*$ is **consistent** with the dataset D iff:

$$h * (x_i) = \phi(x_i),\ \forall xinD$$

1.2 Evaluation/Estimators

True Error Given an hypothesis h that approximates a target function ϕ the **true error** is the probability that $h(x)\phi(x)$:

$$error_t(h) \equiv P(h(x) \neq \phi(x))$$

Cannot be computed.

Sample Error Given an hypothesis h that approximates a target function ϕ the **sample error** is the proportion of samples not correctly classified:

$$error_s(h) \equiv \frac{1}{n} \sum_{x \in D} \delta(h(x), \phi(x))$$

Where $\delta(h(x), \phi(x)) = 1$ iff $h(x) \neq \phi(x)$, 0 otherwise.
We have that $accuracy(h) = 1 - error_s(h)$

Estimation Bias We have that the bias between true error and sample error is:

$$bias \equiv E[error_s(h)] - error_t(h)$$

Comparing hypothesis Having two hypothesis h_1, h_2 the comparison can be:

$$d = error_t(h_1) - error_t(h_2)$$

$$\hat{d} = error_s(h_1) - error_s(h_2)$$

$$E[\hat{d}] = d$$

Overfitting An hypo h overfits the data if, given another hypo h' we have that:

$$error_s(h) > error_s(h') \quad and \quad error_t(h) < error_t(h')$$

K-fold Cross Validation Partition the dataset D into k subsets $D : \{S_1, S_2, ..., S_k\}$. Use one subset S_i as test set and all the other make up the training set $T = S_1 \cup S_2 \cup ... \cup S_{i-1} \cup S_{i+1} \cup ... \cup S_k$.

Others

- **Error rate**= $\frac{FN+FP}{TP+TN+FP+FN}$
- **Accuracy**= 1- error rate
- **Recall**= $\frac{TP}{TP+FP}$
- **Precision**=$\frac{TP}{TP+FP}$
- **F1-score**=

Precision + Recall**Confusion Matrix** : $miss-classificationratebetweenclasses$C$_j, C_i$.

1.3 Decision Trees

Given an instance space X coming from a set of attributes a decision tree has: each internal node tests an attribute, each branch denotes a value of an attribute, each leaf assigns a classification value. A rule is generated for each path to a leaf node.

Entropy Having the proportion of positive samples as p_+ and the negative ones as $p_- = 1 - p_+$ the entropy measure the impurity of the set of samples S:

$$Entropy(S) = -p_+^2 - p_-^2$$

Information gain Information gain measures how well a given attribute separates the training examples according to their target classification. Information gain is measured as reduction in entropy.
$Gain(S, A)$ is the expected reduction in entropy of S caused by knowing the value of attribute A:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(S)} \frac{|S_v|}{S} \cdot Entropy(S_v)$$

ID3 Is an algorithm used to generate a decision tree from a dataset. Hypothesis space search by ID3 is complete (target concept is there), outputs a single hypothesis (cannot determine how many DTs are consistent), no back tracking (local minimal), statistically-based search choices (robust to noisy data), uses al the training examples at each step (not incremental).

Issues

- decise depth
- handling continous attributes
- choosing appropriate attribute selection measures
- missing attribute values
- handling attributes with different costs.

Overfitting and pruning To avoid tree overfitting an approach is to not split a node when the data is not statistically significant. Instead grow a full tree and then post-prune (replace nodes not important with leafs).
To determine correct tree size, use a separate set of examples (training test and validation set), apply statistical test to estimate accuracy of a tree on data distribution

1.4 Probability and Bayes

Posterior Conditional (or posterior) probability belief after the arrival of some evidence. I know the outcome of a random variable, how this affect probability of other random variables?

$$P(a|b) = \frac{P(a \wedge b)}{P(b)}$$

Where denominator can be replaced with a normalization constant α.

2 Bayes Learning

Basic formulas:

- **Product Rule**: $P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$
- **Sum Rule** : $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
- **Theorem of Probability**: Given some mutaully exclusive events $A_1, ..., A_n$ where $\sum_{i=1}^{n} P(A_j) = 1$ we have:

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Bayes Theorem Given:

- $P(h)$ the prior probability of the hypothesis h.
- $P(D)$ the prior probability of the training data D.

We have:

$$p(h|D) = \frac{P(D|h)P(h)}{p(D)}$$

2.1 Maximum a Posteriori probability

When classifying new data we want to assign it the most probable hypothesis. To do so we can use **Maximum a posteriori** hypothesis h_{MAP}:

$$h_{MAP} = \arg\max_{h \in H} P(h|D) = \arg\max_{h \in H} \frac{P(D|h)P(h)}{p(D)} = \arg\max_{h \in H} P(D|h)P(h)$$

where H is the set of all the hypothesis, and from the third equation we have removed the normalizing constant $P(D)$.
Moreover if the prior distribution is uniform, i.e. $P(h_i) = P(h_j),\ \forall i, j \in H$ we can use the **Maximum Likelihood** hypothesis h_{ML} and have:

$$h_{ML} = \arg\max_{h \in H} P(D|h)$$

We can estimate the maximum h_{MAP} by computing $P(h_i|D)$ for every $h_i \in H$ and then get the maximum, **but** h_{MAP} return the most probable *hypothesis*, not the most probable classification, so, given a new instance x, $h_{MAP}(x)$ might not the return the correct classification nor the most probable.

Example Let's consider three possible hypothesis $h_1, h_2 h_3$ driven from data D, where the probability of each hypothesis is:

$$P(h_1|D) = 0.4 \quad P(h_2|D) = 0.3 \quad P(h_3|D) = 0.3$$

as we can see the $h_{MAP} = h_1$, that is the hypothesis is the most probable in the dataset.
Let us now consider a new instance x, and the possible classifications by the three hypothesis:

$$h_1(x) = \oplus \quad h_2(x) = \ominus \quad h_3(x) = \ominus$$

It is now obvious that $h_{MAP}(x) = h_1(x) = \oplus$ is not the most probable classification.

2.2 Bayes Optimal Classifier

To fix the above mentioned problem consider the following:

- $C = \{c_1, c_2, ..., c_n\}$ is a set of possible classes.
- $X = \{x_1, x_2, ..., x_m\}$ is a set of instances of the dataset D
- $f : X \to C$ is the target function that maps an instance to a class.
- $P(c_j|x', h_i)$ is the probability of a new instance x' to be classified as the class c_j by a hypothesis h_i.
- $P(c_j|x', D)$ the probability that x' belongs to the class c_j conditioned to the entire dataset D (every hypothesis).

We have that:

$$P(c_j|x', D) = \sum_{h_i \in H} P(c_j|x', h_i)P(h_i|D)$$

Moreover this kind of function f is called **Bayes Optimal Classifier** [OB], so the most probable class c_{OB} for a new instance x' would be:

$$c_{OB} = \arg\max_{c_i \in C} \sum_{h_j \in H} P(c_i|x', h_j)P(h_j|D)$$

Example Given:

hypothesis prior	positive class	negative class
P(h$_1$|D) = 0.4	P($\oplus$|x, h_1) = 0	P($\ominus$|x, h_1) = 1
P(h$_2$|D) = 0.3	P($\oplus$|x, h_2) = 1	P($\ominus$|x, h_2) = 0
P(h$_3$|D) = 0.3	P($\oplus$|x, h_3) = 1	P($\ominus$|x, h_3) = 0

We have that:

$$\sum_{h_i \in H} P(\oplus|x', h_i)P(h_i|D) = 0.4$$

$$\sum_{h_i \in H} P(\ominus|x', h_i)P(h_i|D) = 0.6$$

Thus:

$$c_{OB} = \arg\max_{c_i \in C} \sum_{h_j \in H} P(c_i|x', h_j)P(h_j|D) = \ominus$$

3 Other Distributions

3.1 Bernoulli

Describes the probability distribution of a binary random variable $X \in \{0, 1\}$:

$$P(X = 1) = \theta \quad P(X = 0) = 1 - theta$$

The probability of X being a certain value x, given θ is:

$$P(X = x, \theta) = \theta^x(1 - \theta)^{1-x}$$

While the probability of obtaining k times the same result with n trials is:

$$P(X = k, n, \theta) = \binom{n}{k} \theta^k (1-\theta)^{n-k}$$

Multi-variate Bernoulli distribution Having a set of mutually independent binary random variable $X_1, X_2, ..., X_n$ following a Bernoulli distribution, the Multi-variate Bernoulli distribution [MvBd] is the product of all n distributions:

$$\prod_{i=1}^{n} P(X_i = x_i, \theta_i)$$

4 Naive Bayes Classifier

When we have to deal with a high hypothesis space, the Bayes Optimal Classifier is not practical anymore. A way to avoid computing every hypothesis is using conditional independence.
Let's write a new instance x' as a set of attributes $A = \{a_1, a_2, ..., a_n\}$, so we have that:

$$c_{OB} = P(c_j|x', D) = P(c_j|A, D) = P(c_j|a_1, a_2, ..., a_n, D)$$

Using the Bayes theorem we have:

$$c_{OB} = \arg\max P(c_j|A, D) = \arg\max \frac{P(A|c_j, D)P(c_j|D)}{P(A|D)} = \arg\max P(A|c_j, D)P(c_j|D)$$

In which $P(A|c_j, D)$ is too difficult to compute.
For this reason the Naive classifier assumes that each attribute is independent from the other, effectively transforming $P(a_1, a_2, ..., a_n|c_j, D)$ to $\prod_i P(a_i|c_j, D)$, so that the f_{NB} becomes:

$$c_{NB} = \arg\max_{c_j \in C} P(c_j|D) \prod_i P(a_i|c_j, D)$$

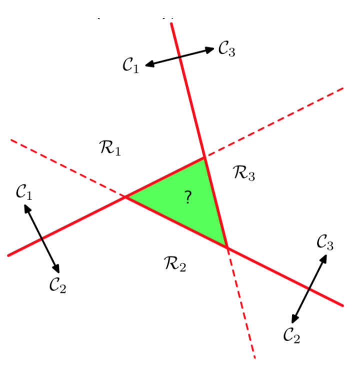
It can happen during estimation that some attributes a_i does not have a prior for a class c_j so that $P(a_i|c_j, D) = 0 \Rightarrow P(c_j|D) \prod_i P(a_i|c_j, D) = 0$. In this case we can set a *virtual prior* to some arbitrary number to avoid having zero.

5 Linear calssification

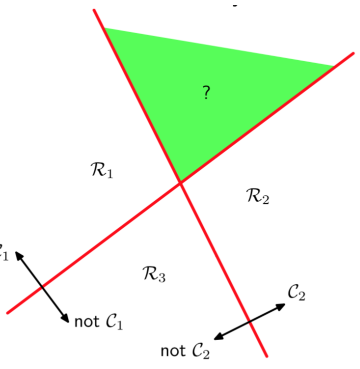
Some instance are linearly separable if it exists some hyper-plane that splits the space in two regions such that different classes are separated. Such hyper-plane is generated by a function f which maps points in $\mathcal{R}^n$ to $C = \{c_1, c_2, ..., c_m\}$

$$f : y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$$

The problem with using multiple classes is shown in Figure **??**.



(a) One Versus One



(b) One Versus Rest

Figure 1: One Versus Rest

A solution is using a **K-Class Discriminant**:

$$f : y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T\boldsymbol{x} + w_{k0}$$

Where the decision boundary between two classes C_j, C_k, is given by:

$$(\boldsymbol{w_j} - \boldsymbol{w_k})^T\boldsymbol{x} + (w_{j0} - wk0)$$

which can be written in a more compact notation having:

- $\tilde{\boldsymbol{w_k}} = \binom{w_{k0}}{\boldsymbol{w_k}}$
- $\tilde{\boldsymbol{x}} = \binom{1}{\boldsymbol{x}}$
- $\tilde{\boldsymbol{W}} = (\boldsymbol{w_1}, \boldsymbol{w_2}, ...\boldsymbol{w_k})$

We get:

$$\boldsymbol{y}(\boldsymbol{x}) = \tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{x}}$$

As shown in figure 2.

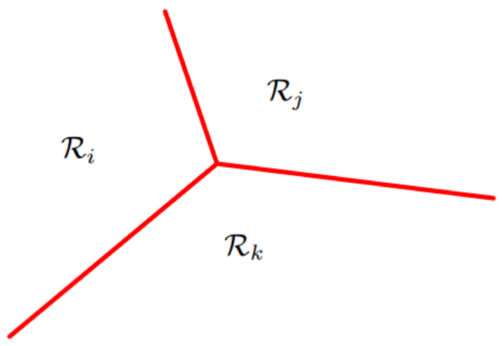


Figure 2: K-class discriminant

Our goal is to determine a good $\tilde{\boldsymbol{W}}$, there are various techniques we can use.

5.1 Least Squares

Given a dataset D made of instances associated with a one hot vector encoder:

$$\forall x_i inD : iff x_i \in c_k \rightarrow t_i = (0, 0, ..., 1, ...0) : t_i[k] = 1$$

The Least Square aims to minimize the sum-of-square error function

5.2 Fischer linear discriminant

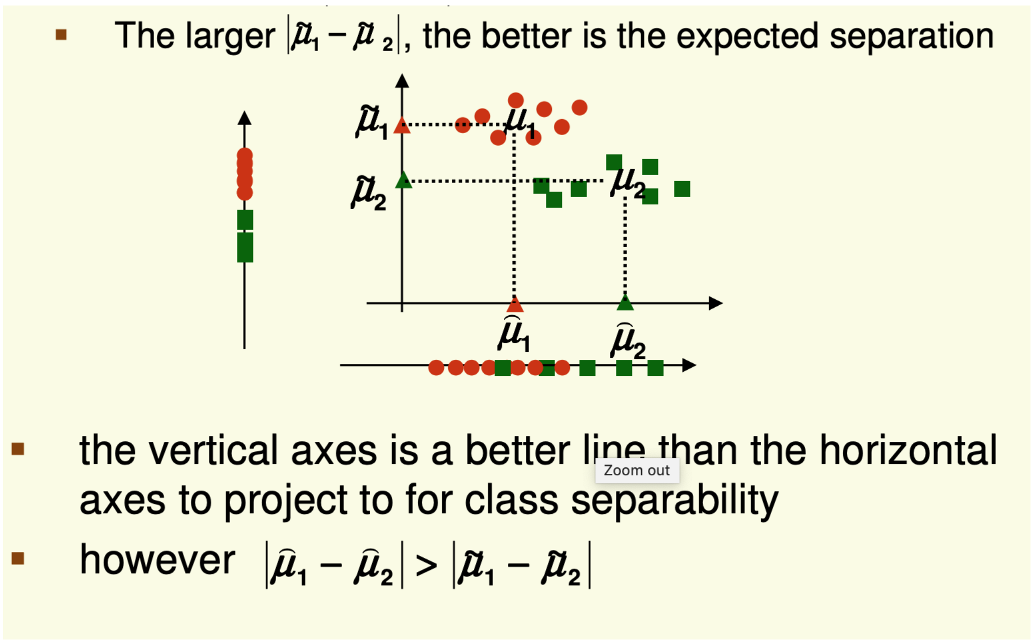
The main idea is to find projection to a line s.t. samples from different classes are well separated.
Suppose we have two classes and n samples $x_1, ..., x_n$ where n_1 samples are from class 1 and n_2 are from 2.
Considere a line which direction is given by a vector v, so that $v^t x_i$ is the projection of a point x_i (2D) onto the line (1D).
We have that the mean of the points of classes 1 is:

$$\mu_1 = \frac{1}{n_1} \sum_{x_i \in C_1}^{n_1} x_i$$

Same thing for μ_2. While the mean of the points **projected** onto the line is simply:

$$\tilde{\mu_1} = v^t \mu_1$$

So we can use $dist = |\tilde{\mu_1} - \tilde{\mu_2}|$ as a measure of separation, since the bigger the better, but look at Figure 4:



- The larger $|\tilde{\mu}_1 - \tilde{\mu}_2|$, the better is the expected separation
- the vertical axes is a better line than the horizontal axes to project to for class separability
- however $|\hat{\mu}_1 - \hat{\mu}_2| > |\tilde{\mu}_1 - \tilde{\mu}_2|$

Figure 3: Mean proble

The problem is that it does not consider the variance between classes, so we need to normalize *dist* by something proportional to the variance.
Let's define the *scatter* of some samples $z_1, ..., z_n$ as :

$$s = \sum_{i=1}^{n} (z_i - \mu_z)^2$$

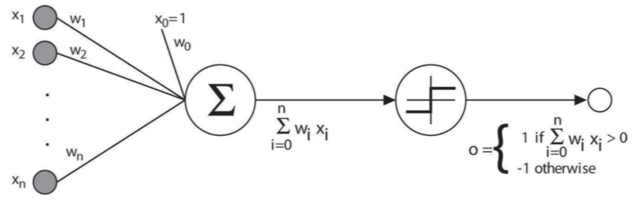
Which is:

$$s_1^2 = \sum_{x_i \in C_1} (x_i - \tilde{\mu_1})^2$$

and the same for s_2^2 So we can now normalize the *dist* by the scatter, getting the fisher linear discriminant:

$$J(v) = \frac{(\tilde{\mu_1} - \tilde{\mu_2})^2}{s_1^2 - s_2^2}$$

6 Perceptron


x_1
w_1
x_2
w_2
x_n
w_n
$x_0=1$
w_0
Σ
$\sum_{i=0}^{n} w_i x_i$
$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

$$o(x_1, \dots, x_d) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_d x_d > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Sometimes we'll use simpler vector notation (adding $x_0 = 1$):

$$o(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} > 0 \\ -1 & \text{otherwise.} \end{cases} = sign(\mathbf{w}^T\mathbf{x})$$

Figure 4: Perceptron

The function to minimize is the square error, given the output o and the true label t:

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}(t_n - o_n)^2 = \frac{1}{2}\sum_{n=1}^{N}(t_n - \boldsymbol{w}^T x_n)^2$$

Since we need to minimize this error we want to move to the direction of the gradient, thus computing the derivative of:

$$\frac{(\boldsymbol{w})}{\partial w_i} = \sum_{n=1}^{N}(t_n - \boldsymbol{w}^T x_n)(-x_{i,n})$$

Where $-x_{i,n}$ is the value of i-th feature.
so we can update the weight w_i by $w_{i+1} = w_i + \Delta w$, where:

$$\Delta w = -\eta\frac{\partial E(\boldsymbol{w})}{\partial w_i} = \sum_{n=1}^{N}(t_n - sign(\boldsymbol{w}^T x_n))) - x_{i,n}$$

Batch mode: Consider all dataset D

$$\Delta w_i = \eta \sum_{(\mathbf{x},t)\in D} (t - o(\mathbf{x}))\, x_i$$

Mini-Batch mode: Choose a small subset $S \subset D$

$$\Delta w_i = \eta \sum_{(\mathbf{x},t)\in S} (t - o(\mathbf{x}))\, x_i$$

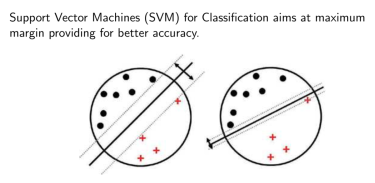
Incremental mode: Choose one sample $(\mathbf{x}, t) \in D$

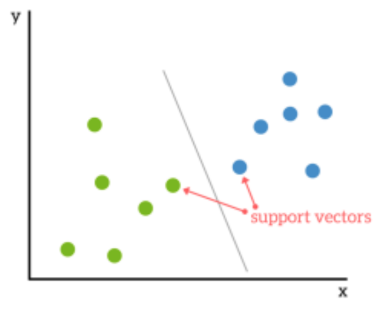
$$\Delta w_i = \eta\, (t - o(\mathbf{x}))\, x_i$$

Figure 5: Perceptron Algotithm

7 Support Vector Machine [SVM]

SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes


Support Vector Machines (SVM) for Classification aims at maximum margin providing for better accuracy.



Support vectors are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. The distance between the hyperplane and the nearest data point from either set is known as the margin. The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, where the margin is estimated as:

$$\arg\max_{\boldsymbol{w}, w_0} \frac{1}{||\boldsymbol{w}||} \min_{n=1..N} [t_n(\hat{\boldsymbol{w}}^{\boldsymbol{T}} \boldsymbol{x_n} + \hat{\boldsymbol{w}}_{\boldsymbol{0}})]$$

Where the

- $\hat{h} = \hat{\boldsymbol{w}}^{\boldsymbol{T}} \boldsymbol{x_n} + \hat{\boldsymbol{w}}_{\boldsymbol{0}}$ is the hyperplane
- $\frac{1}{||\boldsymbol{w}||} \min_{n=1..N} [t_n(\hat{\boldsymbol{w}}^{\boldsymbol{T}} \boldsymbol{x_n} + \hat{\boldsymbol{w}}_{\boldsymbol{0}})]$ is the smallest distance between the hyperplane and x.

8 Probabilistic Generative Models

Consider the case of two classes C_1, C_2, we have that:

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

with:

$$\alpha = ln\frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}$$

We can write:

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

Which is the sigmoid function.

9 Exercises

9.1 Bayes

Provide design and implementation choices for solving the following problem through *Naive Bayes Classifier*:

Classification of scientific papers in categories according to their main subject. The categories to be considered are: ML (Machine Learning), KR (Knowledge Representation), PL (Planning). Data available for each scientific paper are: title, authors, abstract and publication site (name of the journal and/or of the conference).

In Multinomial Naïve Bayes each $p(c)$ is a multinomial distribution, so it is possible to solve the classification problem of documents using Multinomial Naïve Bayes because multinomial distribution works well for data which can easily be turned into counts, such as word counts in a text. Given:

- $P(c_j)$: probability of class c_j
- $P(w_i|c_j)$: probability of the word w_i given the class c_j.

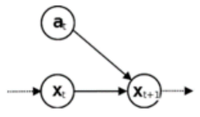
We must estimate $P(c_j)$ and $P(w_i|c_j)$ using multinomial distribution, then use them to classify a new document. Remove every word in the documents not appearing in the vocabulary V build in the previous phase and then use:

$$V_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_{i=1}^{lenght(d)} P(w_i|c_j, D)$$

9.2 Markov

Describe the Markov property of Markovian models representing dynamic systems. Describe the difference between a Markov Decision Process (MDP) and a Hidden Markov Model (HMM). Draw and explain the graphical models of MDP and HMM.

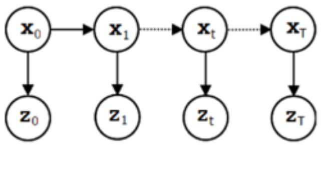
Markov properties once the current state is known, the evolution of the dynamic system does not depend on the history of states, actions and observations. The current state contains all the information needed to predict the future. Future states are conditionally independent of past states and past observations given the current state. The knowledge about the current state makes past, present and future observations statistically independent. Markov process has Markov properties.



MDP An MDP is for decision making where the states are fully observable, hence there is no need of observations. In presence of non-deterministic or stochastic actions, they can be fully observed after its execution.
Having

- X finite set of states:
- A: finite set of actions
- Deterministic $\delta : X \times A \to X$: a function which maps an action-state tuple to the next state.
- Non-deterministic $\delta : X \times A \to 2^X$.
- Stochastic $\delta : P(x'|x,a)$: the probability of having he state x' given previous state x and action a.
- Deterministic $r : X \times A \to R$: a function which maps a tuple state-action to a reward.
- Non-deterministic/Stochastic $r : X \times A \times X \to R$

An MDP is made of $MSP = \{X, A, \delta, r\}$

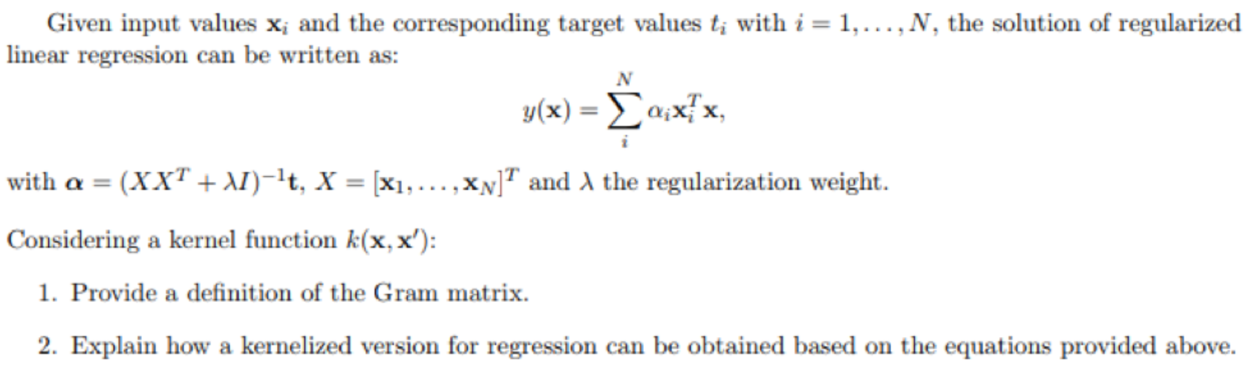


HMM In a HMM sate states x_t are discrete and not observable, while the observations z_t can be either discrete or continuous. Finally there is no control over the system.
Given:

- X: set of not observable states
- Z: set of observations
- $P(x_t|x_{t-1})$: the transition model
- $P(z_t|x_t)$: an observation models which maps the probability of having an observation z_t to a state x_t.
- π_0: an initial distribution
- $\boldsymbol{A} = \{A_{ij}\} = P(x_t = j|x_{t-1} = i)$ a transition matrix
- $b_k(z_t) = P(z_t|x_t = k)$ an observation model.
- $\pi_0 = P(x_0)$: an initial probability.

We have that a HMM is made of $HMM = \{X, Z, \pi_0\}$.

9.3 Kernelized linear model (regression)



Given input values $\mathbf{x}_i$ and the corresponding target values t_i with $i = 1, \dots, N$, the solution of regularized linear regression can be written as:

$$y(\mathbf{x}) = \sum_{i}^{N} \alpha_i \mathbf{x}_i^T \mathbf{x},$$

with $\boldsymbol{\alpha} = (XX^T + \lambda I)^{-1}\mathbf{t}$, $X = [\mathbf{x}_1, \dots, \mathbf{x}_N]^T$ and λ the regularization weight.

Considering a kernel function $k(\mathbf{x}, \mathbf{x}')$:

1. Provide a definition of the Gram matrix.
2. Explain how a kernelized version for regression can be obtained based on the equations provided above.

Gram Matrix Given some kernel $k(x_i, x_j)$, the Gram Matrix $K = \boldsymbol{XX^T}$ is a $N \times N$ symmetric matrix of the form:

$$K_{nm} = k(x_n, x_m)$$

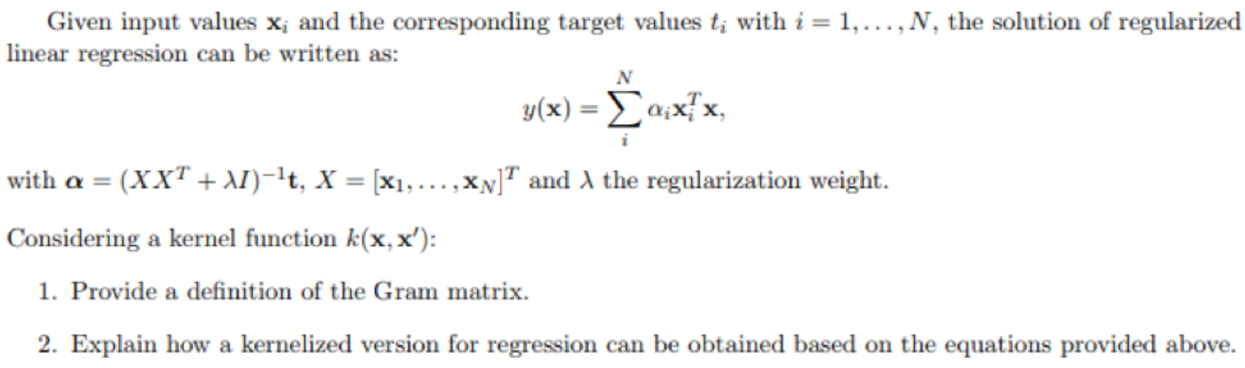
Kernalized regression Considering a linear kernel $k(x, x') = x^T x'$ we can rewrite the model as:

$$y(x) = \sum_{i=1}^{N} \alpha_i k(x_i, x)$$

having:

$$\alpha = (K + \lambda I)^{-1} t$$

9.4 Linear Regression



Given input values $\mathbf{x}_i$ and the corresponding target values t_i with $i = 1, \dots, N$, the solution of regularized linear regression can be written as:

$$y(\mathbf{x}) = \sum_{i}^{N} \alpha_i \mathbf{x}_i^T \mathbf{x},$$

with $\boldsymbol{\alpha} = (XX^T + \lambda I)^{-1}\mathbf{t}$, $X = [\mathbf{x}_1, \dots, \mathbf{x}_N]^T$ and λ the regularization weight.

Considering a kernel function $k(\mathbf{x}, \mathbf{x}')$:

1. Provide a definition of the Gram matrix.
2. Explain how a kernelized version for regression can be obtained based on the equations provided above.

Linear regression The goal of regression is to predict the value of one or more continuous target variables $Y = \mathcal{R}$ given the value of a D-dimensional vector of input variables $X \subset \mathcal{R}^D$. The simplest linear model for regression is one that involves a linear combination of the input variables:

$$y(x, w) = w_0 + w_1 x_1 + ... + w_D x_D = \boldsymbol{w^T x}$$

Where $x_0 = 1$.

Batch vs sequential We have that a target value is given by:

$$t = y(x, w) + \epsilon$$

Where ϵ is some Gaussian noise. If we assume the samples to be i.i.d. then we can use the maximum likelihood and have:

$$max[P(\{t_1, ..., t_n\} | x_1, ..., x_n, w, \beta)]$$

where β^{-1} is the variance of the noise. This approach is used with batch learning in which the entire training set is processed altogether.

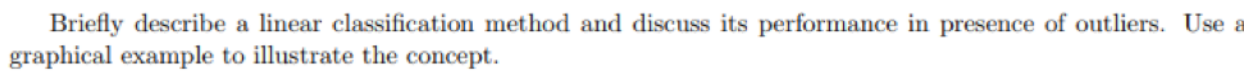
For sequential learning *stochastic gradient descent* can be used to update the model parameters in the following way:

$$w_{n+1} = w_n + \eta \nabla E_n = w_n + [t_n - w_n^T \phi(x_n)]\phi(x_n)$$

Where:

- η: is the learning rate
- $\phi(x_n)$: is a non linear transformation of the input vector

9.5 Linear classification


Briefly describe a linear classification method and discuss its performance in presence of outliers. Use a graphical example to illustrate the concept.

The goal in classification is to take an input vector x and to assign it to one of K discrete classes C_k where $k = 1, ..., K$. In the most common scenario, the classes are taken to be disjoint, so that each input is assigned to one and only one class. Data sets whose classes can be separated exactly by linear decision surfaces are said to be linearly separable. There are various ways of using target values to represent class labels.
For probabilistic models, the most convenient, in the case of two-class problems, is the binary representation in which there is a single target variable $t \in \{0, 1\}$ such that $t = 1$ represents class C_1 and $t = 0$ represents class C_2. The value of t can be viewed as representing the probability of belonging to the class C_1.
For $K > 2$ it is convenient to use a 1-of-K coding scheme, which is a vector $T = \{t_1, t_2, ..., t_n\}$ of length K such that if the class is C_j, for some sample $x_i \in X,\ |X| = n$, then the value of t_i is zero everywhere except for t_i^j which is one.
Given a dataset in the form $D = \{(x_n, t_n)_{n=1}^N\}$ we want to find a linear discriminant $y(x) = W^T x$, hence we minimize the sum of squares error function:

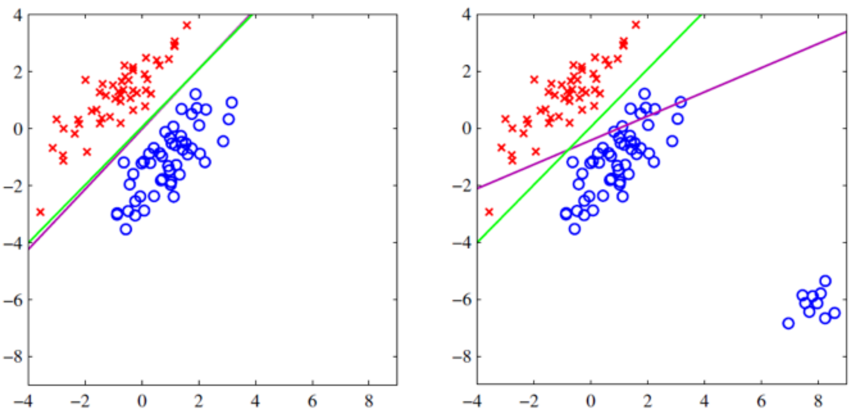
$$E(W) = \frac{1}{2} Tr\{(XW - T)^T (XW - T)\}$$

obtaining:

$$W = (X^T X)^{-1} X^T T$$

$$y(x) = W^T x$$

Unfortunately, least squares solutions lack robustness to outliers and are highly sensitive to them, unlike logistic regression.



The left plot shows data from two classes (red crosses and blue circles) with the decision boundary found by least squares (magenta) and by the logistic regression model (green). The right plot shows the corresponding results obtained when extra data points (outliers) are added at the bottom left of the diagram.

9.6 K-NN

1. Provide the main steps of classification based on K-nearest neighbors (K-NN).
2. Draw an example in 2D demonstrating the application of the 3-NN algorithm for the classification of 3 points given a dataset consisting of points from 4 different classes.

Notes: You can choose how the points of the 4 classes are distributed. Use a different symbol for each class (e.g. use (*,x,+,-) for the classes and (o) for the points to be classified).

Main steps Classification with K-NN, having a target function $f : X \to C$ and a dataset $D = \{(x_i, y_i)_{i=1}^N\}$ is :

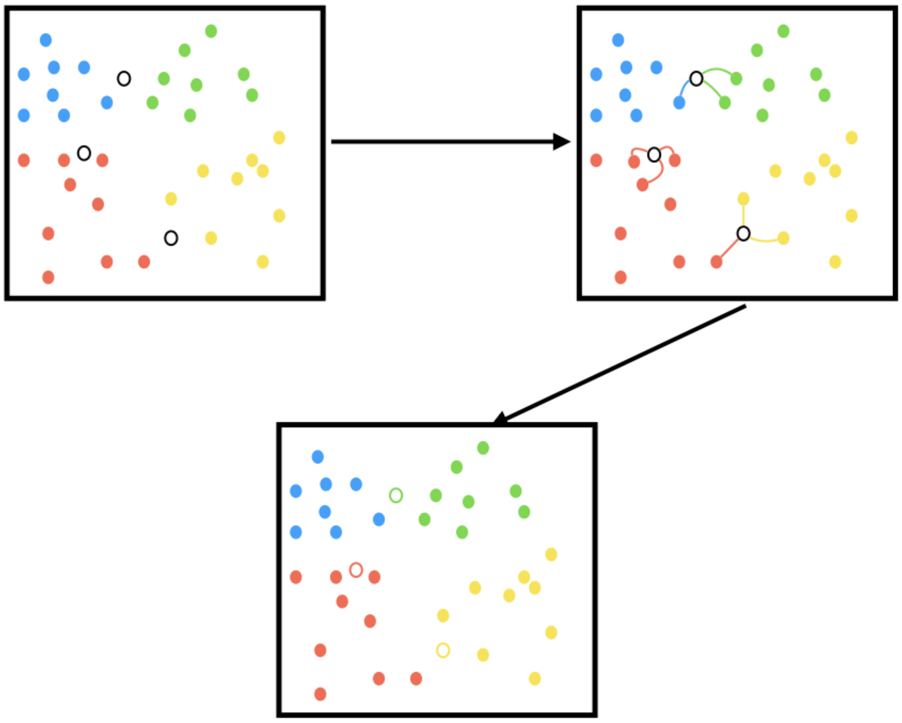
- Find K nearest neighbors of the new instance x'
- assign x' to the most common label among the majority of the neighbors.

We can estimate the likelihood of x' belongign to the class c as:

$$p(c|x', D, K) = \frac{1}{K} \sum_{i \in N_k(x', D)} \mathcal{I}(y_i = c)$$

with:

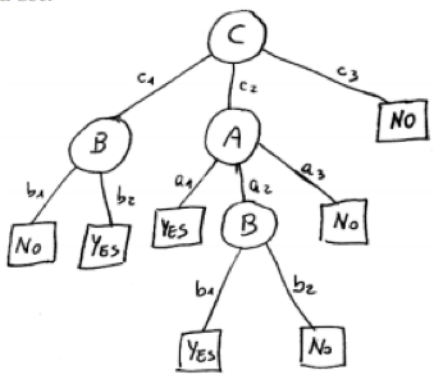
- $N_k(x', D)$ is the set of K nearest points to x'
- $\mathcal{I}(y_i = c)$ is one if the label y_i is equal to c, zero otherwise



Exercise

9.7 Tree classification

Given a classification problem for the function $f : A \times B \times C \rightarrow \{+, -\}$, with $A = \{a_1, a_2, a_3\}, B = \{b_1, b_2\}, C = \{c_1, c_2, c_3\}$ and the following decision tree T that is the result of a learning algorithm on a given data set:



1. Provide a rule based representation of the tree T.
2. Determine if the tree T is consistent with the following set of samples $S \equiv \{s_1 = \langle a_1, b_1, c_1, No\rangle, s_2 = \langle a_2, b_1, c_2, Yes\rangle, s_3 = \langle a_1, b_2, c_3, Yes\rangle, s_4 = \langle a_2, b_2, c_2, Yes\rangle\}$. Show all the passages needed to get to the answer.

Rules

1. $c_1 \wedge b_1 \Rightarrow No$
2. $c_1 \wedge b_2 \Rightarrow Yes$
3. $c_2 \wedge a_1 \Rightarrow Yes$
4. $c_2 \wedge a_2 \wedge b_1 \Rightarrow Yes$
5. $c_2 \wedge a_2 \wedge b_2 \Rightarrow No$
6. $c_2 \wedge a_3 \Rightarrow No$
7. $c_3 \Rightarrow No$

Consistency

- s_1 is consistent because of 1
- s_2 is consistent because of 4
- s_3 is not consistent because of 7
- s_4 is not consistent because of 5

9.8 Tree overfitting

1. Provide a formal definition of overfitting.
2. Discuss the problem of overfitting in learning with Decision Trees and illustrate possible solutions to it.

Overfitting Overfitting is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably". An overfitted model is a statistical model that contains more parameters than can be justified by the data.[2] The essence of overfitting is to have unknowingly extracted some of the residual variation (i.e. the noise) as if that variation represented underlying model structure.

Overfitting in Tree Consider an hypothesis h with error on the training set $error_{train}(h)$, while the error on the enitre distribution D is $error_D(h)$. The hypo h overfits the training data if there is some other hypo h' and the following holds:

$$error_{train}(h) < error_{train}(h') \quad and \quad error_D(h) > error_D(h')$$

stop growing when data split is not statistically significant, acquire more training data, remove irrelevant attributes, grow full tree, then post-prune. In order to select the best tree, we have to measure performance over training data, measure performance over separate validation data set, add complexity penalty to performance measure.

9.9 Boosting

1. **Provide the main features about boosting.**
2. **Write the error function whose minimization leads to a formulation equivalent to the AdaBoost algorithm.**

Boosting In supervised learning, boosting converts weak learners to strong ones. A weak learner is defined to be a classifier that is only slightly correlated with the true classification (it can label examples better than random guessing). In contrast, a strong learner is a classifier that is arbitrarily well-correlated with the true classification.

AdaBoost No prior knowledge about base learner is required, no parameters to tune (except for M), can be combined with any method to find base learners and theoretical guarantees given enough data and base learners with moderate accuracy.
The error function to minimize is:

$$J_m = \sum_{n=1}^{N} w_n^{(m)} \mathcal{I}(y_m(x_n) \neq t_n$$

Given

- $\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} \mathcal{I}(y_m(x_n) \neq t_n}{\sum_{n=1}^{N} w_n^{(m)}}$
- $\alpha = ln(\frac{1-\epsilon_m}{\epsilon_m})$

The output of the linear classifier is:

$$Y_m(x) = sign(\sum_{m=1}^{M} \alpha_m y_m(x))$$

1. Provide the main features about boosting.
2. Write the error function whose minimization leads to a formulation equivalent to the AdaBoost algorithm.

Boosting In supervised learning, boosting converts weak learners to strong ones. A weak learner is defined to be a classifier that is only slightly correlated with the true classification (it can label examples better than random guessing). In contrast, a strong learner is a classifier that is arbitrarily well-correlated with the true classification.

AdaBoost No prior knowledge about base learner is required, no parameters to tune (except for M), can be combined with any method to find base learners and theoretical guarantees given enough data and base learners with moderate accuracy.
The error function to minimize is:

$$J_m = \sum_{n=1}^{N} w_n^{(m)} \mathcal{I}(y_m(x_n) \neq t_n)$$

Given

- $\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} \mathcal{I}(y_m(x_n) \neq t_n}{\sum_{n=1}^{N} w_n^{(m)}}$
- $\alpha = ln(\frac{1-\epsilon_m}{\epsilon_m})$

The output of the linear classifier is:

$$Y_m(x) = sign(\sum_{m=1}^{M} \alpha_m y_m(x))$$

9.10 Unsupervised vs supervised

Machine learning problems can be categorized in supervised and unsupervised. Explain the difference between them providing a precise formal definition (not only explanatory text) in terms of input and output of the two categories of problems.

Supervised learning is typically done in the context of:

- **Classification**: when we want to map input to output labels.
- **Regression**: when we want to map input to a continuous output.

In both regression and classification, the goal is to find specific relationships or structure in the input data that allow us to effectively produce correct output data.

Unsupervised learning uses techniques such as clustering, representation learning and density estimation. In all of these cases, we wish to learn the inherent structure of our input data without using explicitly-provided labels. Since no labels are provided, when analyzing the output there is no specific way to compare model performance in most unsupervised learning methods.

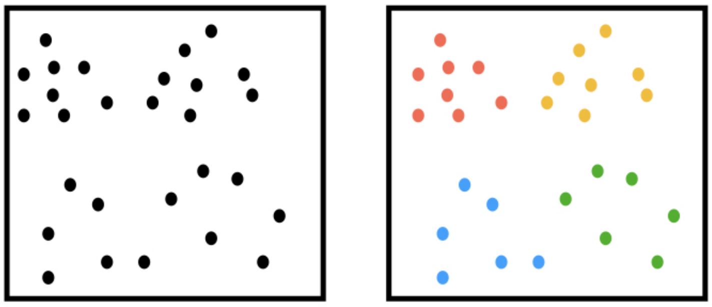
On the other hand, supervised learning intends to infer a conditional probability distribution $p_X(x|y)$ conditioned on the label y of input data; unsupervised learning intends to infer an a priori probability distribution $P_X(x)$. Compared to supervised learning where training data is labeled with the appropriate classifications, models using unsupervised learning must learn relationships between elements in a data set and classify the raw data without "help."

9.11 Unsupervised

1. Define with a precise formal definition the unsupervised learning problem.
2. Provide a full example of unsupervised learning problem (i.e., a specific invented data set), possibly in a graphical form.
3. Describe a solution to the defined problem based on K-Means, providing examples of execution of some steps of the algorithm and a reasonable solution.

Definition Unsupervised learning is a branch of machine learning that learns from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning identifies commonalities in the data and reacts based on the presence or absence of such commonalities in each new piece of data. Alternatives include supervised learning and reinforcement learning.

Example Using K-Means with number of clusters equal to 4, we get



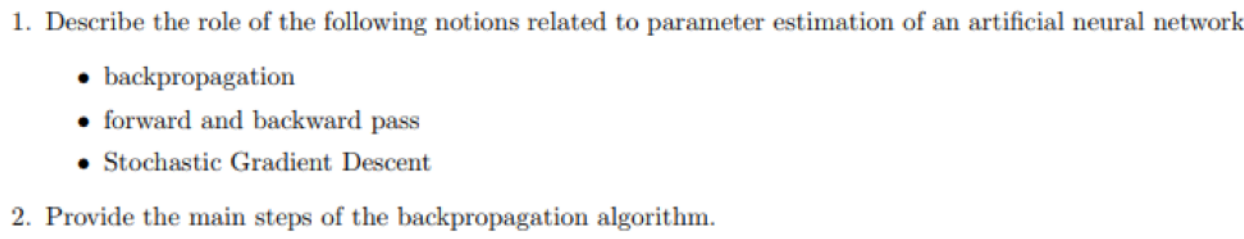
K-Means Nope

9.12 ANN

Consider a regression problem for the target function $f : \Re^8 \rightarrow \Re^4$. Design a solution based on Artificial Neural Network for this problem: draw a layout of a suitable ANN for this problem and discuss the choices.

1. Determine the size of the ANN model (i.e., the number of unknown parameters).
2. Is Backpropagation algorithm affected by local minima? If so, how can we avoid or attenuate it?
3. Is Backpropagation algorithm affected by overfitting? If so, how can we avoid or attenuate it?

9.13 ANN2



1. Describe the role of the following notions related to parameter estimation of an artificial neural network

   - backpropagation
   - forward and backward pass
   - Stochastic Gradient Descent

2. Provide the main steps of the backpropagation algorithm.

Back-Propagation Backpropagation algorithm is used to propagate gradient computation from the cost through the whole network. The goal is to compute the gradient of the cost function w.r.t. the parameters $\nabla_\theta J(\theta)$ in a simple and efficient way.

Backward and Forward pass The forward pass computes values from inputs to output and the backward pass then performs backpropagation which starts at the end and recursively applies the chain rule to compute the gradients all the way to the inputs of the circuit. The gradients can be thought of as flowing backwards through the circuit.

Stochastic Gradient Descend Stochastic Gradient Descent updates the weight parameters after evaluation of the cost function after each sample. That is, rather than summing up the cost function results for all the sample then taking the mean, SGD updates the weights after every training sample is analysed.

Back-Propagation algorithm