Supervised Learning

R Basheer Ahammad

1 Introduction

Given a set of data points $\{x^{(1)}, \ldots, x^{(m)}\}$ associated with a set of outcomes $\{y^{(1)}, \ldots, y^{(m)}\}$, we want to build a model that learns how to predict y from x.

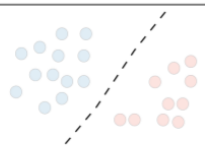
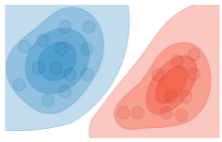
Type Of Prediction

The different types of predictive models are summed up in the table below:

	Regression	Classifier
Outcome	Continuous	Class
Examples	Linear regression	Logistic regression, SVM, Naive Bayes

Type Of Model

The different models are summed up in the table below:

	Discriminative model	Generative model
Goal	Directly estimate $P(y\|x)$	Estimate $P(x\|y)$ to deduce $P(y\|x)$
What's learned	Decision boundary	Probability distributions of the data
Illustration		
Examples	Regressions, SVMs	GDA, Naive Bayes

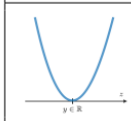
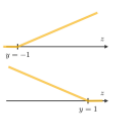
2 General Concepts

Hypothesis

The hypothesis is noted h_θ and is the model that we choose. For a given input data $x^{(i)}$, the model prediction output is $h_\theta(x^{(i)})$.

Loss Function

A loss function is a function $L : (z, y) \in \mathbb{R} \times Y \to L(z, y) \in \mathbb{R}$ that takes as inputs the predicted value z corresponding to the real data value y and outputs how different they are. The common loss functions are summarized in the table below:

Least squared	Logistic	Hinge	Cross-entropy
$\frac{1}{2}(y-z)^2$	$\log(1+\exp(-yz))$	$\max(0, 1-yz)$	$-\left[y\log(z) + (1-y)\log(1-z)\right]$
			
Linear regression	Logistic regression	SVM	Neural Network

Cost Function

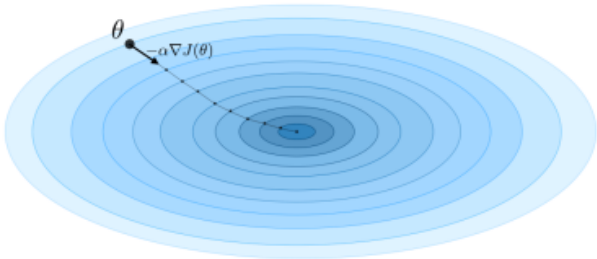
The cost function J is commonly used to assess the performance of a model, and is defined with the loss function L as follows:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} L(h_\theta(x^{(i)}), y^{(i)})$$

Gradient Descent

By noting $\alpha \in \mathbb{R}$ the learning rate, the update rule for gradient descent is expressed with the learning rate and the cost function J as follows:

$$\theta \leftarrow \theta - \alpha \nabla J(\theta)$$



Likelihood

The likelihood of a model $L(\theta)$ given parameters θ is used to find the optimal parameters θ through maximizing the likelihood. In practice, we use the log-likelihood $\ell(\theta) = \log(L(\theta))$ which is easier to optimize. We have:

$$\theta_{\text{opt}} = \arg\max_\theta L(\theta)$$

3 Support Vector Machines

The goal of support vector machines is to find the line that maximizes the minimum distance to the line

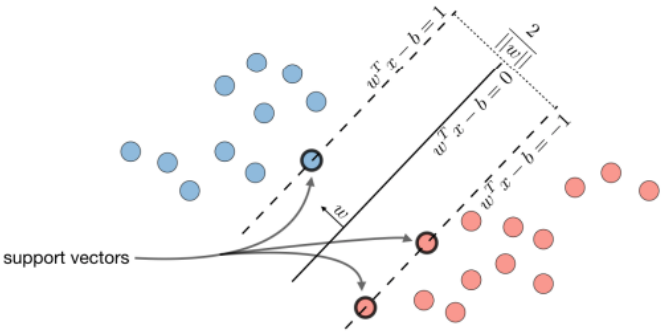
Optimal margin classifier

The optimal margin classifier h is such that:

$$h(x) = \text{sign}(w^T x - b)$$

where $(w, b) \in \mathbb{R}^n \times \mathbb{R}$ is the solution of the following optimization problem:

$$\min_{\frac{1}{2}||w||^2} \quad \text{such that} \quad y^{(i)}(w^T x^{(i)} - b) > 0$$



Hinge Loss

The hinge loss is used in the setting of SVMs and is defined as follows:

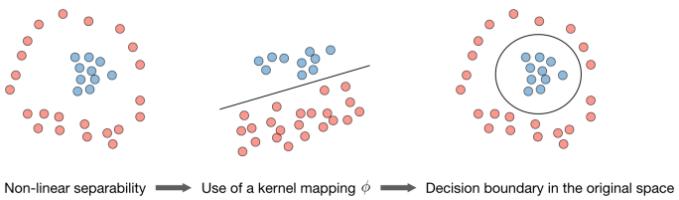
$$L(z, y) = [1 - yz]^+ = \max(0, 1 - yz)$$

Kernel

Given a feature mapping ϕ, we define the kernel K to be defined as:

$$K(x, z) = \phi(x)^T \phi(z)$$

In practice, the kernel K defined by $K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$ is called the Gaussian kernel and is commonly used.



4 Bayesian Classifier

Suppose we have K classes $C1, C2, \ldots, CK$ and a set of n training examples $(x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \in \mathbb{R}^{d \times 1}$ is a d-dimensional feature vector and $y_i \in \{1, 2, \ldots, K\}$ is the corresponding class label. Let $P(C_k)$ be the prior probability of class C_k, i.e., the probability that a randomly chosen example belongs to class C_k. Let μ_k and Σ_k be the mean vector and covariance matrix for class C_k. To classify a new example x, we compute the posterior probability of each class C_k given x using Bayes' rule:

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$

$$= \frac{\frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k)\right) P(C_k)}{\sum_{j=1}^{K} \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_j)^T \Sigma_j^{-1}(x - \mu_j)\right) P(C_j)}$$

where $|\Sigma_k|$ denotes the determinant of Σ_k, and Σ_k^{-1} is the inverse of Σ_k. The class with the highest posterior probability is then selected as the predicted class for x:

$$\hat{y} = \arg \max_{k \in \{1, 2, \ldots, K\}} P(C_k|x)$$

5 Decision Tree

6 K Nearest Neighbour

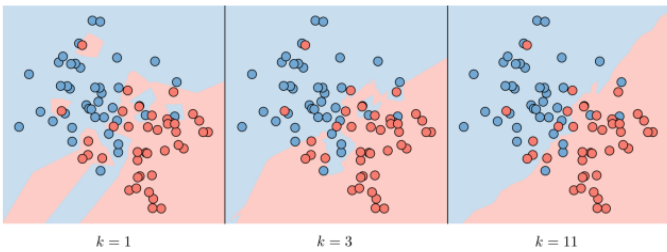
The k-nearest neighbors algorithm, commonly known as k-NN, is a non-parametric approach where the response of a data point is determined by the nature of its k neighbors from the training set. It can be used in both classification and regression settings.

Algorithm 1: K-nearest Neighbors (KNN) Algorithm in K-means Style

Input : Training examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, New example x, Number of neighbors K

Output : Predicted class $\hat{y}$

```
for i = 1 to n do
    Calculate distance: d_i = ||x - x_i||^2;
Sort distances d_1, d_2, ..., d_n in ascending order;
Initialize empty list neighbors;
for i = 1 to K do
    Find index j corresponding to d_i in the sorted list;
    Append training example (x_j, y_j) to neighbors;
Initialize empty dictionary class_votes;
forall (x_j, y_j) in neighbors do
    Increment class_votes[y_j] by 1;
Select predicted class: ŷ = arg max_y class_votes[y];
```



***