Optimization and Data Analytics, Course Notes

Alex Justesen Karlsen, 201404623

January 13, 2019

Contents

Optimization

Optimization problems can be classified as;

- Constrained Optimization vs. Unconstrained Optimization
- Continuous Optimization vs. Discrete Optimization
- Deterministic Optimization vs. Stochastic Optimization

Optimization is an important tool in making decisions and in analyzing physical systems. In mathematical terms, an optimization problem is the problem of finding the best solution from among the set of all feasible solutions. [1] Problems can be solved analytically (symbolic), when solutions do exist. Numerical methods can solve (some) problems, that can not be solved analytically by approximation. Optimization can be classified in to categories: Constrained or Unconstrained, Continuous or Discrete and Deterministic or Stochastic Optimization.

Most algorithm do not guarantee to find global optimum, there exist method that does, however these are computationally heavy and time exhaustive. Local search are often preferred, as the local solution are *"good enough"* and are not as time consuming.

Constrained Optimization vs. Unconstrained Optimization

Unconstrained Optimization Unconstrained optimization problems arise directly in many practical applications; they also arise in the reformulation of constrained optimization problems in which the constraints are replaced by a penalty term in the objective function.

Constrained Optimization Constrained optimization problems arise from applications in which there are explicit constraints on the variables. The constraints on the variables can vary widely from simple bounds to systems of equalities and inequalities that model complex relationships among the variables.

Constrained optimization problems can be furthered classified according to the nature of the constraints (e.g., linear, nonlinear, convex) and the smoothness of the functions (e.g., differentiable or non-differentiable) [2].

Continuous Optimization vs. Discrete Optimization

Models with discrete variables are discrete optimization problems; models with continuous variables are continuous optimization problems.

[1] NEOS. Accessed December 20, 2018. https://neos-guide.org/optimization-tree.
[2] NEOS. Accessed December 20, 2018. https://neos-guide.org/optimization-tree.

Continuous optimization problems tend to be easier to solve than discrete optimization problems; the smoothness of the functions means that the objective function and constraint function values at a point x can be used to deduce information about points in a neighborhood of x [3]. It is important as the continuity assumption allow us to use calculus methods.

Discrete optimization problems use discrete variables and tend to be harder to solve as these calculus method cannot be applied to function that are not continuous and form gaps and intervals.The travelling salesman problem, is a well-known discrete optimization problem, that tries to optimize the best route trough a set of discrete variables i.e. cities.

Deterministic Optimization vs. Stochastic Optimization

Deterministic Optimization *Always produces the same result from initial input (e.g. for same candidate solution x_0)* – Carl

In deterministic optimization, it is assumed that the data for the given problem are known accurately. However, for many actual problems, the data cannot be known accurately for a variety of reasons. The first reason is due to simple measurement error. The second and more fundamental reason is that some data represent information about the future (e. g., product demand or price for a future time period) and simply cannot be known with certainty.[4]

Stochastic Optimization Does not always produce the same result. Randomness is added to the algorithm e.g. random selection of candidate solutions or neighborhoods. Stochastic methods are presumably faster and more robust, as they have mechanism to avoid getting stuck in local minimums.

In optimization under uncertainty, or stochastic optimization, the uncertainty is incorporated into the model. Robust optimization techniques can be used when the parameters are known only within certain bounds; the goal is to find a solution that is feasible for all data and optimal in some sense. Stochastic programming models take advantage of the fact that probability distributions governing the data are known or can be estimated; the goal is to find some policy that is feasible for all (or almost all) the possible data instances and optimizes the expected performance of the model. [5]

Stochastic also means random. The opposite term is deterministic. Deterministic optimization given some input x is guaranteed to always output the same y. Stochastic methods does not provide such guarantee, however stochastic methods are presumably faster and more robust, as they have mechanism to avoid getting stuck in local minimums.

[3] NEOS. Accessed December 20, 2018. https://neos-guide.org/optimization-tree.
[4] NEOS. Accessed December 20, 2018. https://neos-guide.org/optimization-tree.
[5] NEOS. Accessed December 20, 2018. https://neos-guide.org/optimization-tree.

Mathematical Preliminaries

One-dimension

$$f : R^1 \rightarrow R^1$$

$FONC$:

$$f'(x) = 0$$

$FOSC$:

For a maximum (not entirely sure)

$$f''(x) > 0$$

For a minimum (not entirely sure)

$$f''(x) < 0$$

N-dimensions

$$f : R^n \rightarrow R^1$$

$SONC$:

$$\nabla f(x_1, x_2) = 0$$

∇f is the gradient of f, which is a vector of the partial derivatives of f.

$$\nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

$SOSC$:

Considering the definiteness of the Hessian matrix reveals optimum as max, min or saddle point, using an eigen value analysis.

The hessian matrix is a matrix of the second order derivatives of f.

$$D^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 x_2} \\ \frac{\partial^2 f}{\partial x_2 x_1} \end{bmatrix}$$

Linear Programming

Matrix Games

Geometric Method

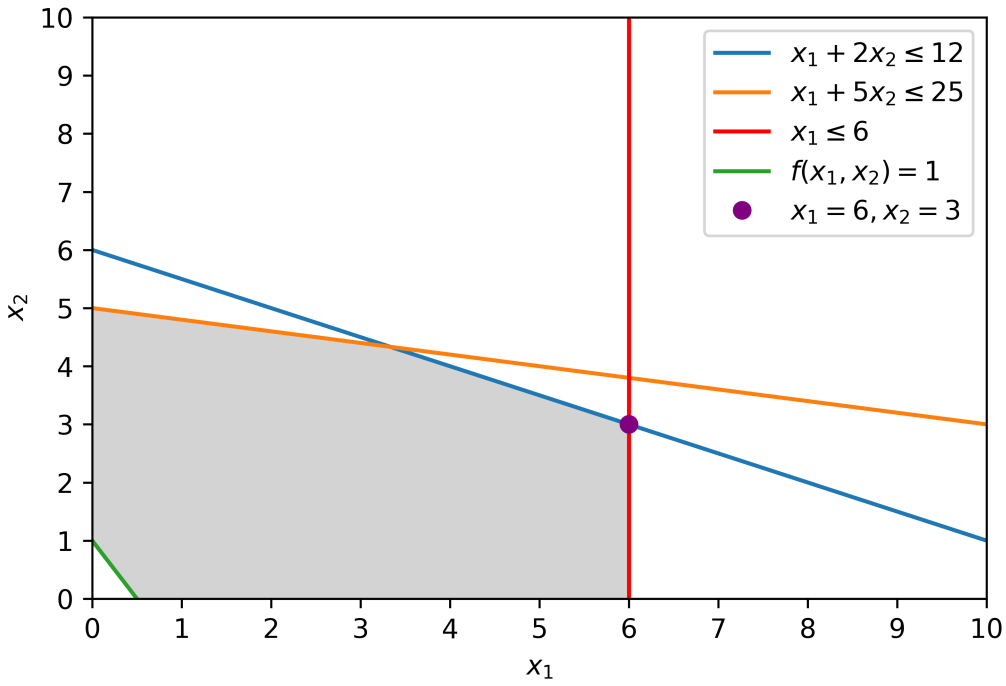

Figure 1: Geometric Method

Simplex

Simplex is a method for linear programming problem. It must be a **maximization** problem on standard form.

- *All variables must be non-negative* ≥ 0.

- *Constraints should be non-negative* $(\leq)$.

The objective function should have all the variables on the left-hand-side and is equal to zero e.g. $Z = 3x + 4y$ becomes $-3x - 4y + Z = 0$.

Simplex introduces slack variables to turn inequalities into equations e.g. $2x_1 + x_2 \leq 8$ becomes $2x_1 + x_2 + x_3 = 8$.

Once the above is satisfied the initial problem can be formulated on simplex tableau form:

x_1	x_2	$\hat{x}_3$	$\hat{x}_4$	$\hat{x}_5$	M	y
1	4	1	0	0	0	16
6	4	0	1	0	0	30
2	-5	0	0	1	0	6
-6	-5	0	0	0	1	0

Table 1: Initial simplex tableau

Similarly to Gaussian Elimination a pivot column must be chosen. The rest of the column should be reduced to zero and the pivot element should equal one.

The pivot column j should be the largest negative i.e. smallest value of the bottom row, which is the coefficients. The pivot row should be the smallest value. The pivot row i is determined by $\min(\frac{y_n}{x_n})$.

Duality Theorem Primal Problem P

$$maximize\, f(x) = c^T xAx \leq bx \geq 0$$

Dual Problem P'

$$minimize\, g(y) = b^T yA^T y \leq cy \geq 0$$

Constrained Optimization

$$minimize\, f$$
$$subject\, to\, \in \Omega$$

Where $x = (x_1, ..., x_n)$ optimization variables, or decision variables. $f: R^n \to R$, objective function. Ω, is the constraint set or feasible set, sometimes $\Omega = \{x \in R^n | f_i(x)$, where $i = 1, ..., m\}$. $f_i(x) R^n \to$

R where $i = 1, ..., m$.constraint functions. x optimal solution or minimizer, is the smallest value of f among $x = (x_1, ..., x_n)$ satisfying the constraints.

Solving Linear Equations

Linear System of Equations $Ax = b$ can be solved using Gaussian Elimination. For some system of equation no solution exists, and an approximate solution is the best we can get. We can typically project b onto the linear subspace spanned by A. The solution will have some error term, and we wish to find the projection with the least mean squared error.

Non-linear Constrained Optimization

The lagrange theorem:

$$F(x, \lambda) = f(x) - \lambda(h(x))$$

$$DF(x, \lambda) = \nabla f(x) - \nabla\lambda(h(x))$$

DF is the partial derivatives of x and λ

Solve the system of equations

$$DF(x, \lambda) = \nabla f(x) - \nabla\lambda(h(x)) = 0$$

If the solution has optimizers, these will be given as the solution to the system of equations.

Unconstrained Optimization

Unconstrained optimization has no constraints i.e. bound for it's optimal value. An optimal solution of the function is sought and there are different ways to obtain optimum numerically. We cannot be sure to find a global optimum and sometimes, a locally optimum is sufficient. Other methods have mechanism to that helps the algorithm obtain global optimums or an approximation hereof.

Root-Finding and Optimization

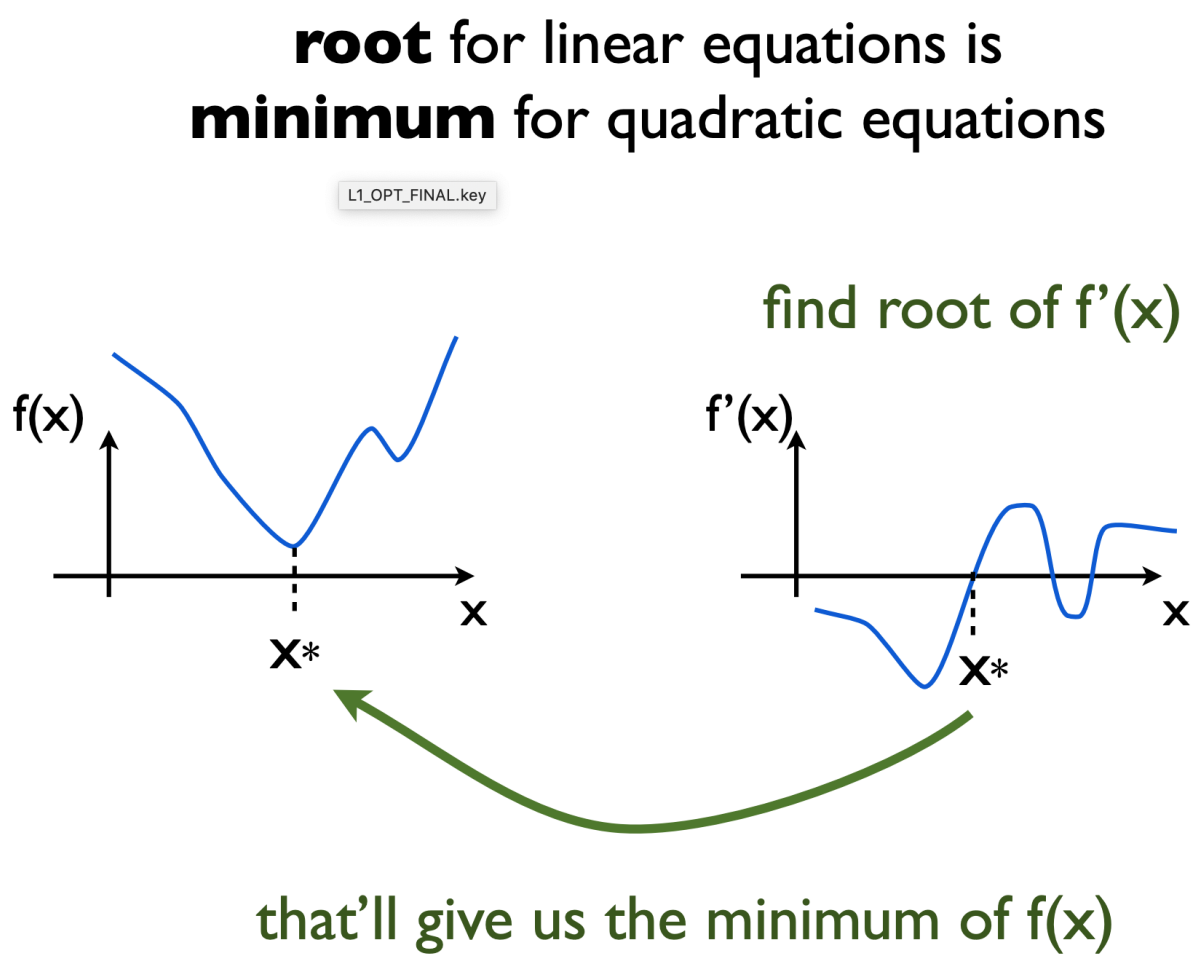

Figure 2: Root finding and optimization

1D Line Search Methods

1D Line search methods are methods that seek an optimum a long the line of a 1D function.

Golden Section Search

Class: Deterministic Golden section search are given an initial interval in which is should try to find an optimum. The interval is narrowed down by the golden ratio number $\phi = \sqrt{2}$. The algorithm converges, when there is no or sufficiently small progress of yet another iteration.

Fibonacci Search

Class: Deterministic Fibonacci search is golden section search with improved narrowing down mechanism. Instead of the golden ratio the method uses the Fibonacci series to shrink the search space.

Newtons Method

Class: Deterministic There are two basic approaches the tangent method and the secant method.

1D

Tangent method

$$x(t+1) = x(t) - \frac{g(x)}{g'(x)}$$

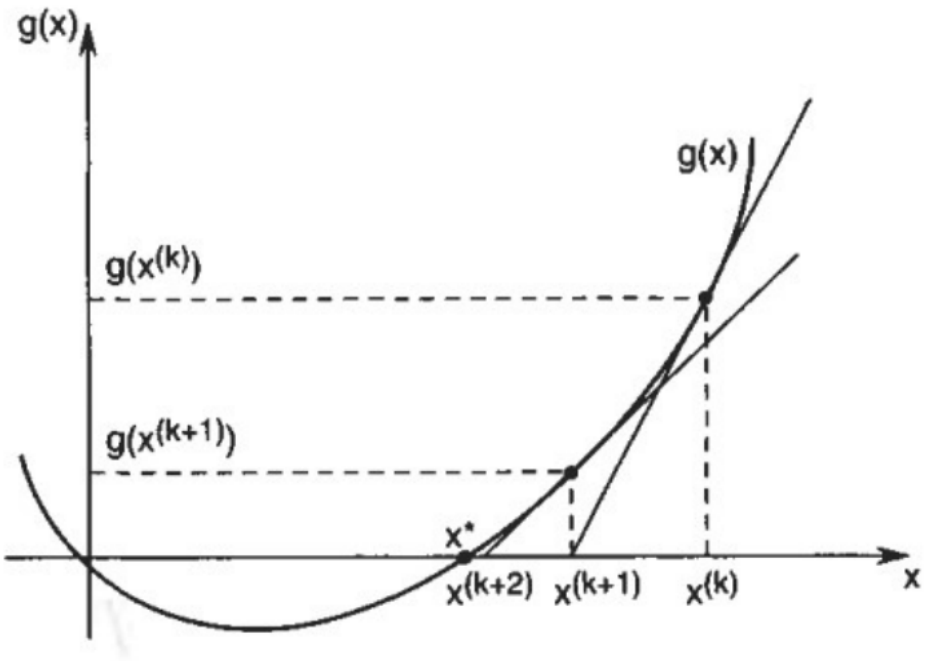

Figure 3: Newton 1D Tangent Method

Basically it computes the root of the tangent of $g(x)$, to get the next x.

Secant Method

$$x(t+1) = x(t) - \frac{x(t) - x(t-1)}{g(x(t)) - g(x(t-1))} g(x(t))$$

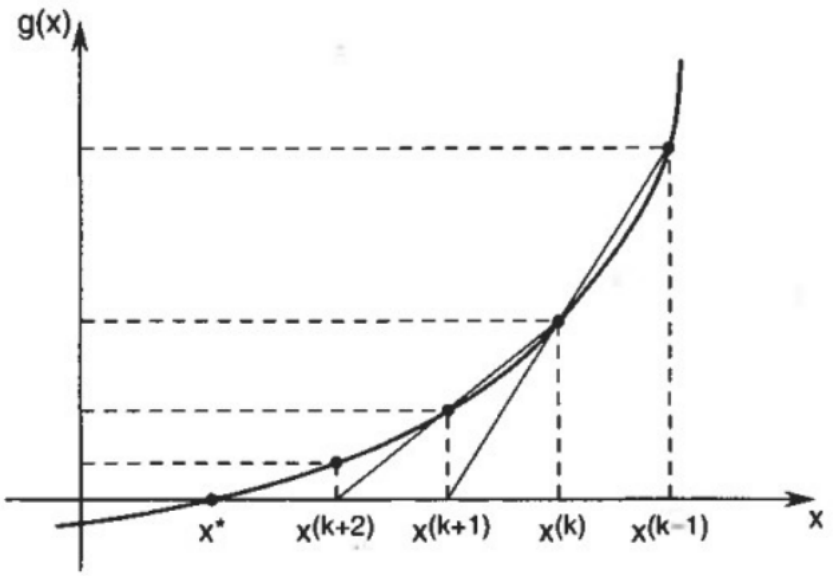

Figure 4: Newton 1D Secant Method

Basically it computes the secant of $g(x)$ minus the previous step $g(x-1)$.

Both methods converges when the value of improvement gets desirably small by each increment.

$$|x^{(k+1)} - x^{(k)}| < e$$

Or when the residual size gets small enough i.e. close to the root.

$$f(x^{(k+1)}) < e$$

Or when we have done too many iterations.

$$k > k_{max}$$

For optimization at $f'(x) = 0$ newtons method can be described as;

$$x^{(k+1)} = x^{(i)} \frac{f'(x^{(i)})}{f''(x^{(i)})}$$

nD

Expanding newtons method to n-dimension, basically just mean to turn x into a vector X. For root finding purposes the problem can be rewritten as;

$$X^{(k+1)} = X^{(i)} - J^{(i)-1} f(x^{(i)})$$

Where J is the Jacobian matrix.

Newton method for optimization, where the gradient $\nabla f(x) = 0$, in nD can be stated as;

$$X^{(k+1)} = X^{(i)} - H^{(i)-1} \nabla f(x^{(i)})$$

Where H is the Hessian matrix.

NB. Newton's method computes the first and second order derivative i.e. Jacobian and Hessian matrices in a point of f.

Newton's for non-linear least-square

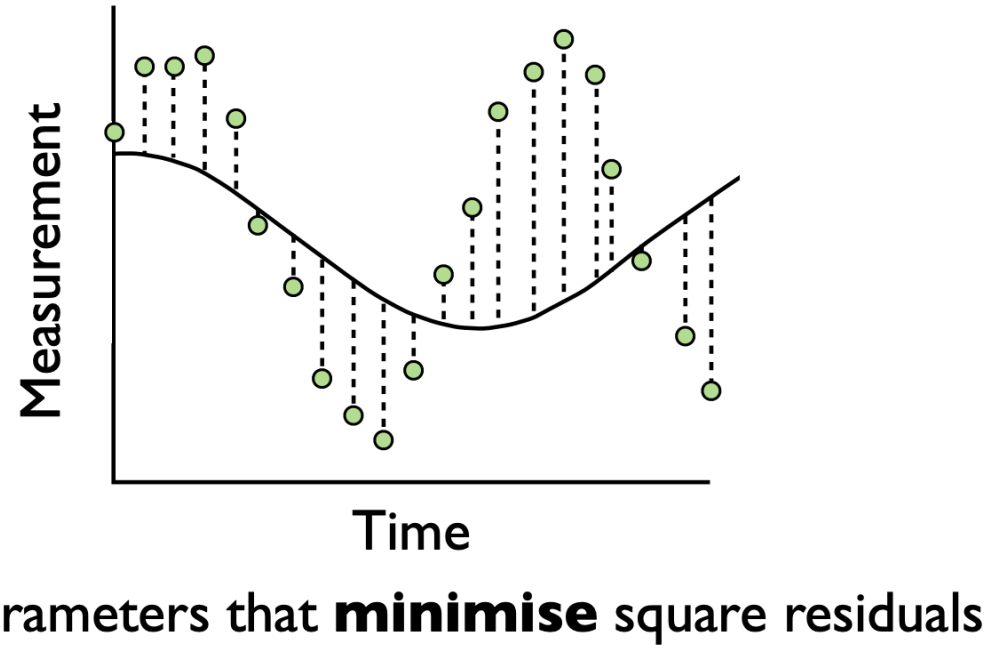

Figure 5: Newton Curve Fitting

Quasi-Newton

Quasi Newton's approximates Jacobian and Hessian matrices, in cases of unavailability or if they are too expensive to compute.

We replace the hessian with an approximation B.

$$X^{(k+1)} = X^{(i)} - B^{(i)-1}\nabla f(x^{(i)})$$

$$B = \frac{\nabla f(x_k + \Delta x) - \nabla f(x_k)}{\Delta x}$$

NB. It is actually just the secant method in nD.

As iteration steps increases B converges to true Hessian.

Quasi-Newton is faster and more robust than newton

Gradient Methods

Steepest Descent

The step-size is the movement in the direction of the gradient i.e. steepest descent, that minimizes f. The next step point is given as this point, and we once again move in the direction of the gradient, that minimizes the function f, this is done iteratively until convergence.

Class: Deterministic

$$\alpha_k = \arg\min_{\alpha>0} f(x^{(k)} - \alpha\nabla f(x)^k))$$

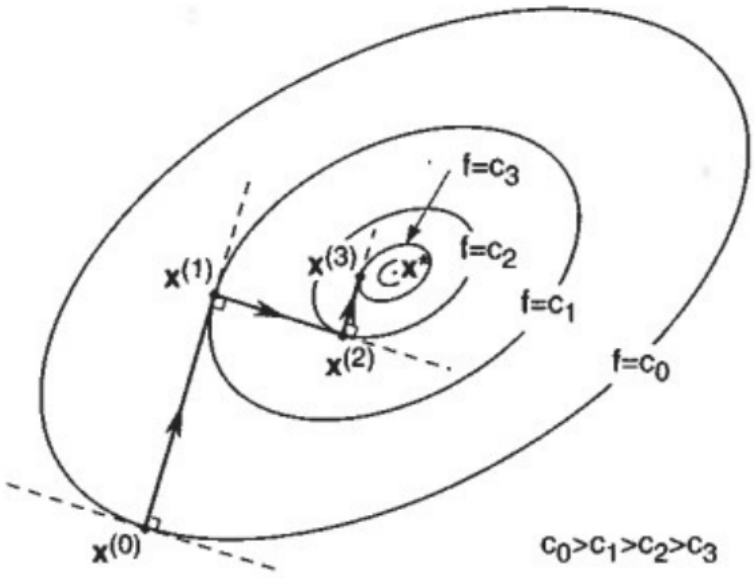

Figure 6: Steepest Descent

Quadratic Form

$$Q(t) = x^T Ax$$

Where A is a $n \times n$ symmetric matrix.

Then the steepest descent of the quadratic form is given by;

$$f(x) = \frac{1}{2}x^T Qx - b^T x$$

Where x and $g(x)$ is given by;

$$x^{(k+1)} = x^{(k)} \frac{g^{(k)T} g^{(k)}}{g^{(k)T} Q g^{(k)}} g^{(k)}$$

$$g^{(k)} = \nabla f(x^{(k)}) = Qx^{(k)} - b$$

Conjugate Methods

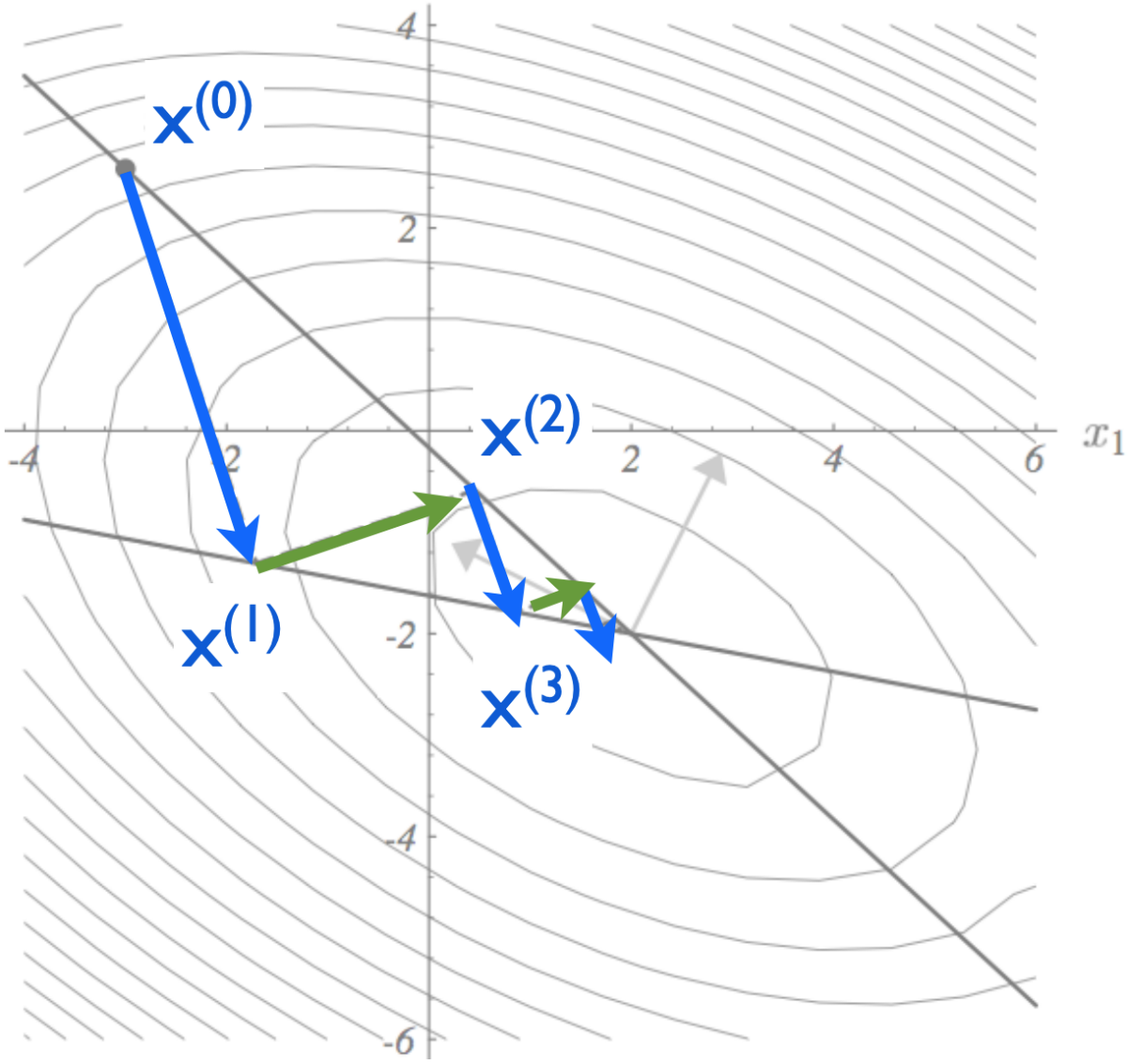

Figure 7: Gradient Descent Zig-Zag

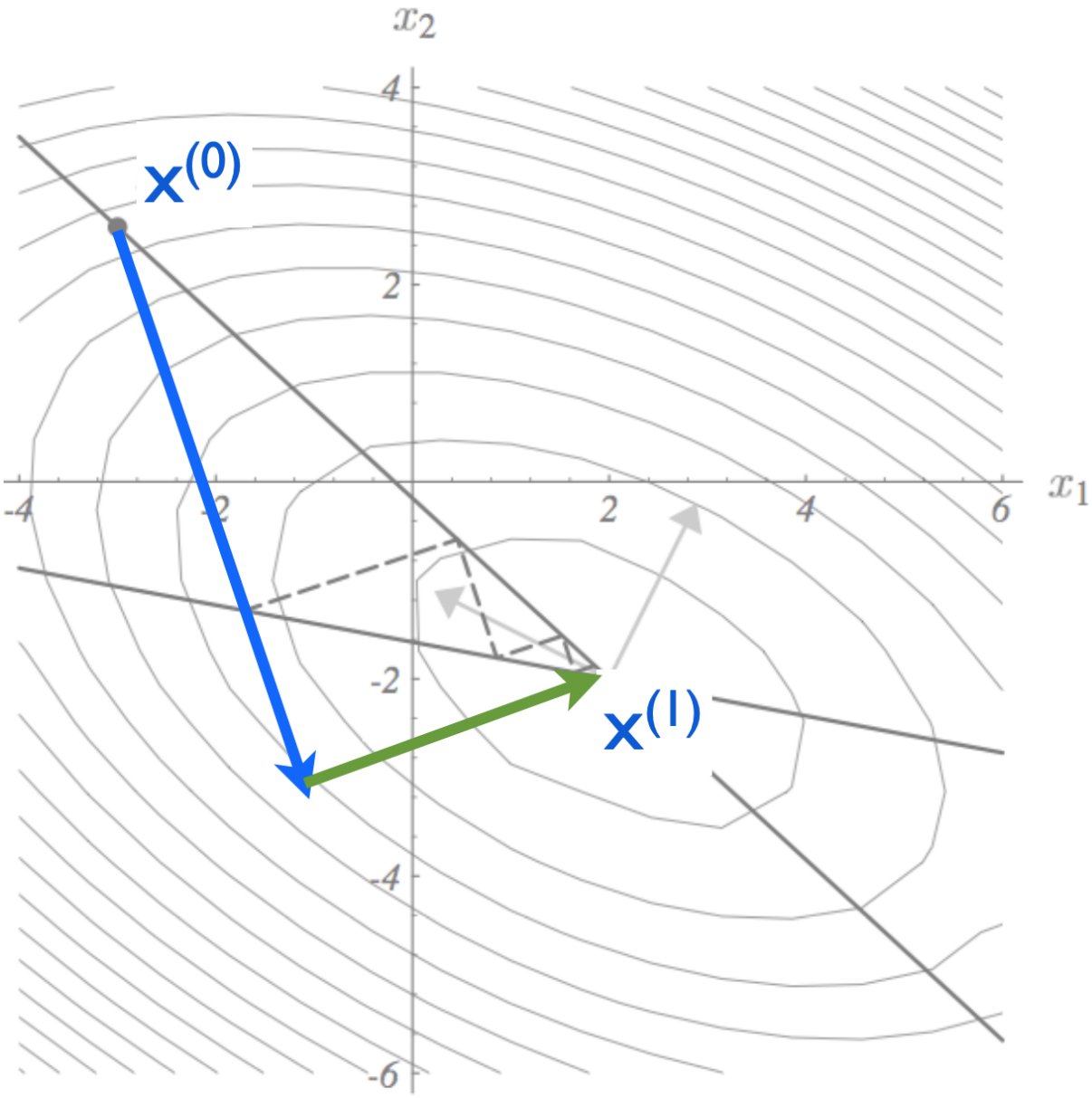

Figure 8: Conjugate Directions

Conjugate Descent

Class: Deterministic

Global Search Methods

Global Search Algorithms have countermeasures to avoid getting stuck in a local optimum and seek global optimum for the optimization problem.

Some of these global algorithms uses a population of candidate solutions and not just a single candidate solution at each iteration.

A population of candidate is a way of spreading out the search field of an optimization algorithm. Instead of a single candidate, that either search along a line or random steps from the former solution, the population searches in many areas of the function at the same time. Particle swam optimization is one such method. It spreads out a swarm of particles i.e. candidate solutions and in corporation they find an approximation of the global optimum.

Simulated Annealing

Simulated Annealing have countermeasures for getting stuck in local optimum, by accepting a worse candidate solution with some probability. Improving candidate solutions are always accepted. Simulated Annealing is inspired by metallurgy, where the blacksmith heath the metal material and lets it slowly cool off. It makes the metal easier to work with. Simulated annealing uses the term of hot and cold. When it's hot, the algorithm is more likely to take a worse candidate solution, as it get colder the probability decrease i.e. it gets less bendy.

```
Let s = s0
For k = 0 through kmax (exclusive):
T = temperature(k / kmax)
Pick a random neighbour, snew = neighbour(s)
If P(E(s), E(snew), T) >= random(0, 1):
s = snew
Output: the final state s
```

$$e^{(\frac{-(f(z_i)-f(x_i))}{T_i})} > r$$

If solution is worse (higher than for minimization), the lhs will be between $0...1$, r is a random integer between either 0 or 1. As the temperature cools off over time the lhs will get converge to zero, which essentially reduces the probability of taking a worse candidate solution over time.

Particle Swarm Optimization

Particle Swarm Optimization is inspired by swarms of insects finding good spots. Good spots are evaluated as fitness of some function $f(x, y)$. The solution seeks the global optimum or an approximation hereof.

The particle swarm has two update formulas one for particles velocity;

$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

and one for particle position;

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

The algorithm can be described as follows;

> For each p in P
>
> 1. Evaluate the fitness i.e. heat of particle p as $h(x_i, y_i)$.
> 2. If $h(\hat{x}_i, \hat{y}_i) < h(x_i, y_i)$ Then update the p best know position $\hat{x}_i$ (cognitive component) If $g(t) < h(x_i y_i)$ Then update the best known global position $g(t)$ (social component).
> 3. Update the particles individual velocity and position according to the update formulas in (i).

Genetic Algorithms

Uses a chromosomatic representation of the data. It could be a bit string explaining a step in a certain direction.

Using two bits it could be; Up: 00, Right: 01, Left: 10, Down 11.

A series of five steps could be represented like this;(00,01,01,00,10)

The algorithm is inspired by evolution theory and can be explained as follows;

```
Given initial population generation t0
1. Select intermediate population based on fitness (Selection)
2. use intermediate population to create offspring (Cross-over and
   Mutation)
3. population (t+1) is offspring
Repeat until stop criterion is met
```

Single-Point Cross-Over Single-point cross-over of gene (1) and gene (2) result in gene (3).

$$(1)\ 00010|10010$$

$$(2)\ 00100|00010$$

$$(3)\ 00010|00010$$

Cross-over point is indicated by | between index 5 and 6. First part is (1) second part is (2).

Mutation Step trough cromosome bit by bit. Randomly decide to flip with a probability e.g. 0.1%.*

(2) $000\mathbf{0}000010$

Index 4 has been flipped indicated with bold font.

Data Analytics (Machine Learning)

Machine Learning (ML) is a branch within Data Science to construct algorithms, that are self-improving based on statistical models. ML approaches can be classified into 3 main categories;

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

Supervised Learning

Is the learning task of mapping an input vector to an output response. Supervised learning uses data labelled by human experts. The learning phase tries to optimize the model to correctly map the input to the labeled output. The model are then used to predict the response of unlabeled data. Model are either used for regression or classification, depending on wether the data is continuous or discrete.

Unsupervised Learning

Is the learning task of finding commonalities in forms of structure and patterns in unlabeled raw test data. The algorithm are unsupervised i.e. without human help.

Reinforcement Learning

The training process is iteratively the model is presented an input and guesses an output. An expert corrects the guess only by binary feedback. Depending on the feedback the training either updates the model or continue the process.

Data Analytics Machine Learning is also known as Data analytics. Data Analytics is the discovery, interpretation and communication of meaningful patterns in data. Typically the analysis is four-fold; data preprocessing, data representation, representation preprocessing and model selection/training.

An example using image data: 1. Data preprocessing $\rightarrow$ image segmentation 2. Data representation $\rightarrow$ vectorizing 3. Representation preprocessing $\rightarrow$ centering 4. Model selection $\rightarrow$ classification

Regression

Is the procedure of fitting a model to describe some training data. The model can be used to make predictions given some input test data. The model predict some quantitative value for the input data.

$$Y = M(X)$$

Where Y is the output vector, M is the model and X is the input vector.

Classification

Is the procedure of fitting a decision function to discriminates classes in training data. The model can be used to make prediction, that classifies between qualitative classes.

$$Y = C(X)$$

Where Y is the output label, C is the classifier and X is the input vector.

Unsupervised Learning Techniques

Unsupervised learning is the learning process or finding patterns, structures and similarities in unlabeled data. It is widely used in data mining.

To find these patterns and discriminate some classes in data from other classes, similarity measures is used determine the degree of which the data has the same structure.

Distance-based similarity

Similarity can in some applications, be describe as the distance $d(x_i, x_j)$ between two vectors.

Similarity function based on distance function can have the forms;

$$s(x_i, x_j) = \frac{\sigma}{d(x_i, x_j)}$$

$$s(x_i, x_j) = e^{-\frac{d(x_i, x_j)}{\sigma}}$$

In the above any distance function between two vector-pairs.

Euclidean Distance

$$d_E(x_i, x_j) = \|x_i - x_j\|_2 = \sqrt{\sum_{d=1}^{D}(x_{id} - x_j d)^2}$$

Manhattan Distance

$$d_M(x_i, x_j) = \sum_{d=1}^{D} |x_{id} - x_j d|$$

Minkowski Distance

$$d(x_i, x_j) = \left(\sum_{d=1}^{D} |x_{id} - x_j d|^q \right)^{\frac{1}{q}}$$

Angular-based similarity

$$s(x_i, x_j) = \frac{x_i^T x_j}{\|x_i\|_2 \|x_j\|_2}$$

K-means clustering algorithm

Given a data set of size N

```
Generate k centroids vectors randomly.

Do:
    Assign each sample to the nearest centroid for the sample.

    Update the centroid to become the mean vector of the samples.

Until no change
```

Algorithm 2: K-Means clustering

1: Initialize $\boldsymbol{\mu}_k,\ k = 1, \ldots, K$
2: **Do**
3: Assign all vectors $\mathbf{x}_i,\ i = 1, \ldots, N$ to a cluster by:
4: $l^* = \arg\min_l \|\mathbf{x}_i - \boldsymbol{\mu}_l\|_2^2$
5: Update the cluster mean vectors by:
6: $\boldsymbol{\mu}_k^* = \frac{1}{N_k} \sum_{\mathbf{x}_i \in D_k} \mathbf{x}_i,\ \ k = 1, \ldots, K$
7: **until** no change in $\boldsymbol{\mu}_k^*,\ k = 1, \ldots, K$

Figure 9: KMeans Pseudo Code

Fuzzy K-means clustering

K-means clustering uses har-assignment, by assigning a sample to one or another cluster distinctively. Fuzzy or soft assignment allow us to assign a sample partly to one or the other. The membership is a percentage of how much we assign each sample to every cluster. ## Dimensional Reduction Techniques

Principal Component Analysis

Decision Theory

Decision Theory is about *Minimize expected loss* of a decision e.g. classification or regression.

An Email Spam filter is a good example. We can classify mails as ham or spam. The loss of getting a mail, that was actually spam is not very costlt, however filtering an important mail can be potentially harmful.

A loss matrix could look like:

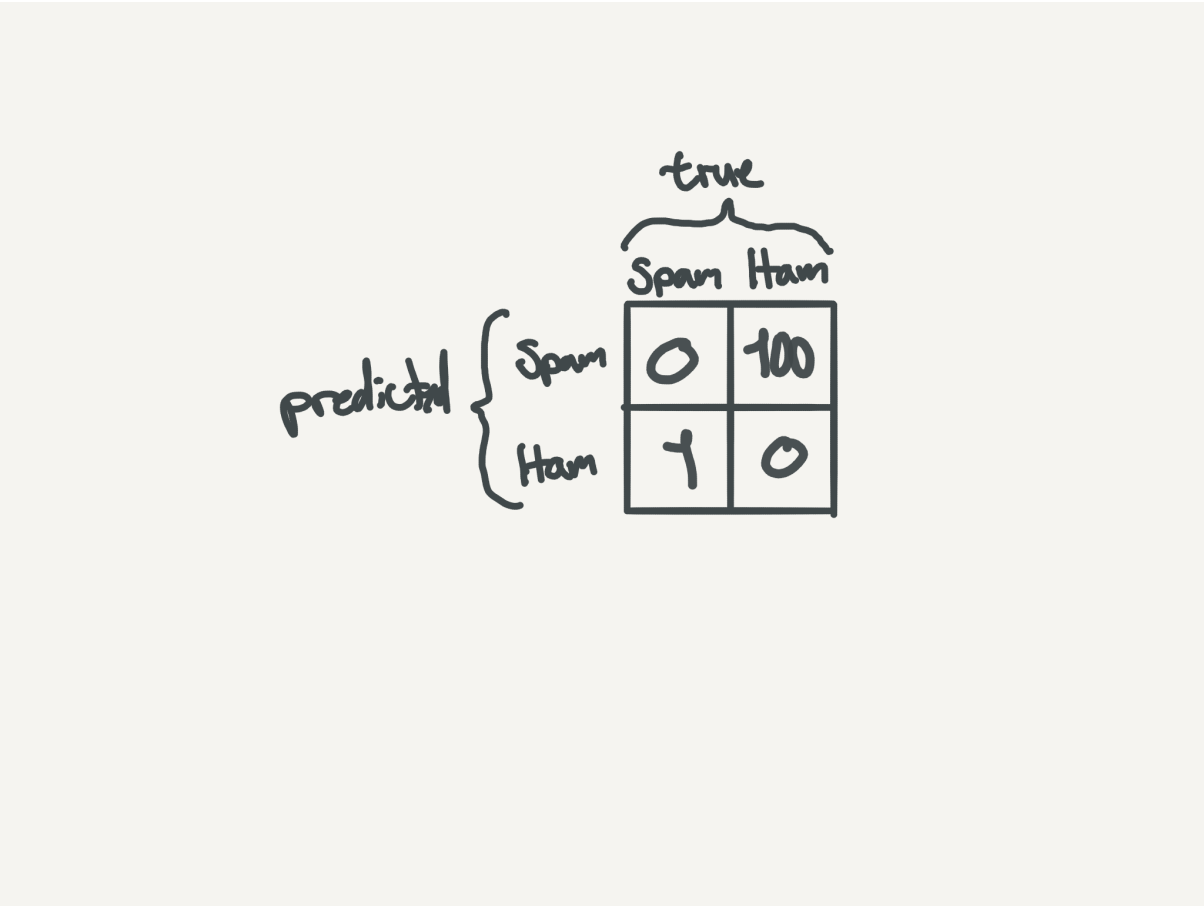

Figure 10: Spam Filter loss function

The are different kind of loss functions e.g.

- "1-0" loss function, that either classifies correctly (l=0) or incorrectly (l=1).
- Squared loss function, that squared the loss, typically used in regressions.

Statistically we wish the smallest loss on average. Using "1-0" loss function we classify based on the conditional probability $p(y|x)$. The optimization problem can be described as;

$$\hat{y} = \arg\min_{y} p(y|x)$$

Is also called maximum likelihood classification.

State of nature

Probability-Based Learning

Assuming a classification problem with K possible outcomes;

$$C = \{c_1 ... c_k\}$$

C is the set of classes.

$P(c_k)$ denotes the prior probability of the outcomes.

The probability of all comes are denoted as;

$$\sum_{k=1}^{K} P(c_k) = 1$$

We do NOT classify based on the priori, if $P(c_1) > P(c_2)$, then all would be classified as the c_1.

$P(c_k|x)$ denotes the conditional probability, that states the probability of class c_k given the observation x.

The joint probability of $c_k 1$ and x is

$$P(c_k, x) = P(c_k|x)P(x) = P(x|c_k)P(c_k)$$

From this we can obtain Bayes' formula

$$P(c_k|x) = \frac{p(x|c_k)P(c_k)}{p(x)}$$

The error can then be defined as

$$P(error|x) = \begin{cases} P(c_1|x), if\ x\ is\ misclassified\ to\ c_2 \\ P(c_2|x), if\ x\ is\ misclassified\ to\ c_1 \end{cases}$$

Which is given by;

$$P(error) = \int_{-\infty}^{\infty} P(error, x)dx = \int_{-\infty}^{\infty} P(error|x)p(x)dx$$

Thus, we can define Bayes' rule as decision rule:

Decide c_1 if $P(c_1|x) > P(c_2|x)$, else decide c_2.

The decision function is obtained by finding x for $P(c_1|x) = P(c_2|x)$.

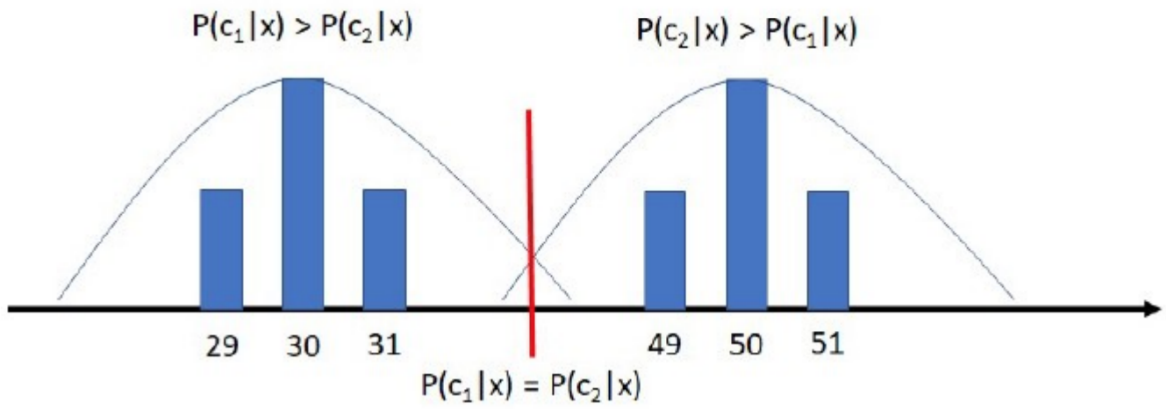

Figure 11: Bayes Decision Function

More than two classes??

Risk-Based Decision Functions

Suppose given observation x , we take the action α_i classifying the sample to class i.

We define the loss function $\lambda(\alpha_i|c_k)$, which expresses the loss incurred taking action α_i, given the correct class is c_k.

The risk of α_i for observation x can be defined as:

$$R(\alpha|x) = \sum_{k=1}^{K} \lambda(\alpha_i|c_k)P(c_k|x)$$

Gaussian Decision Functions

We make assumptions about the distribution of samples data is Gaussian. In reality it can take any distribution, however many has proven to be gaussian. The central limit theorem tells, that aggregating a large number from a lot of small independent disturbance will turn out to be gaussian.

When sampling a variable a lot of disturbance is collected as well. The expected value of a continuous variable can be determined as;

$$\varepsilon[f(x)] = \int_{-\infty}^{\infty} f(x)p(x)dx$$

The normal distribution $p(x)$ is given by:

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

For discrete variables of set D, it can be expressed as;

$$\varepsilon[f(x)] = \sum_{x \in D} f(x) P(x) dx$$

where $P(x)$ is the Probability Mass Function (PMF) of x.

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

μ is the mean and σ is the standard deviation.

$$\mu = \varepsilon[x] = \int_{-\infty}^{\infty} x \cdot p(x) dx$$

$$\sigma^2 = \varepsilon[(x-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 \cdot p(x) dx$$

The following model visualizes the normal distribution. 95% are within 95% of the interval of $2\mu + \sigma$ (i.e. confidence interval).

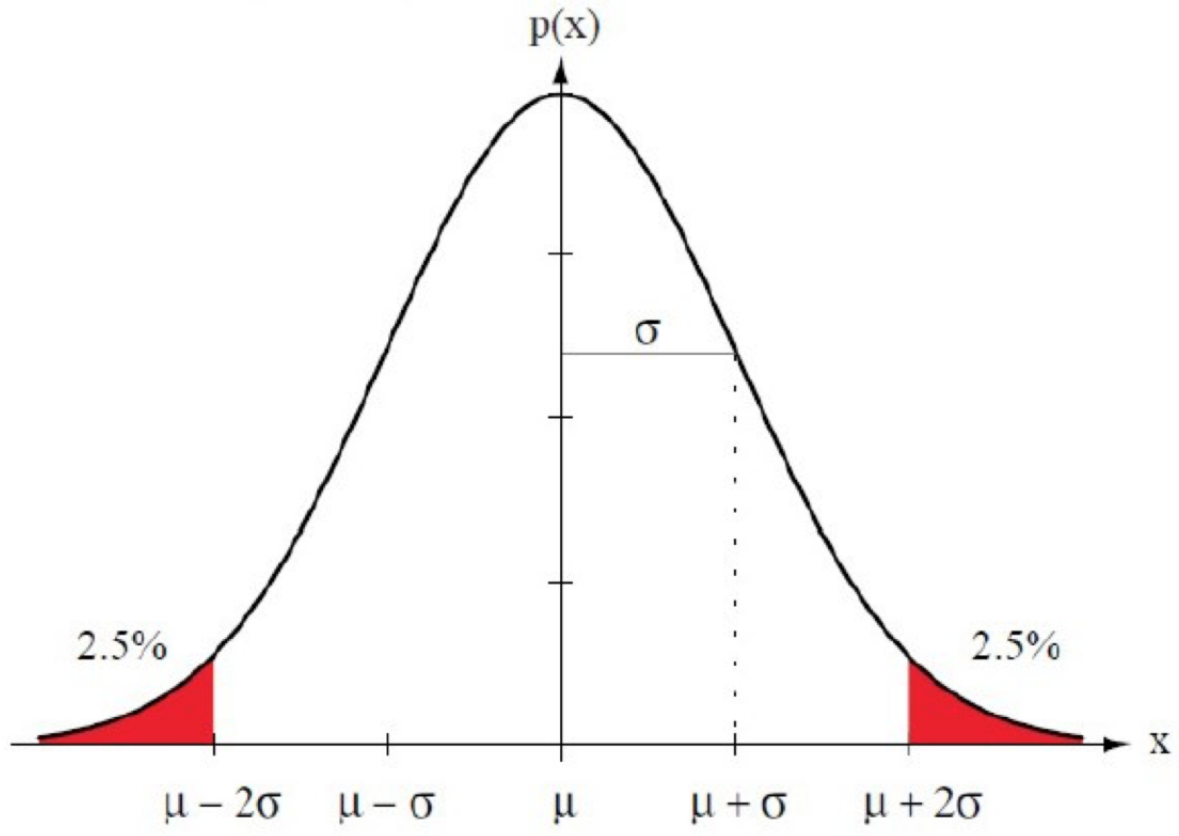

Figure 12: Normal Distribution

Normal distribution are also denoted $\mathcal{N}(\mu, \sigma^2)$.

Multi-variate normal distribution

In higher dimensions, we have a D-dimensional vector x;

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix}$$

The normal distribution is denoted $\mathcal{N}_k(\mu, \sigma^2)$, where $p(x)$ can be expressed as;

$$p(x) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

where Σ is the covariance matrix, that is defined as;

$$\Sigma = \varepsilon[(x-\mu)(x-\mu)^T] = \int (x-\mu)(x-\mu)^T p(x)dx$$

Σ can be calculated using 1D operations.

$$\Sigma_{ij} = \varepsilon[(x_i - \mu_i)(x_j - \mu_j)]$$

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \dots & \Sigma_{1D} \\ \vdots & \ddots & \vdots \\ \Sigma_{D1} & \dots & \Sigma_{DD} \end{bmatrix}$$

The covariance matrix have some important properties;

- Σ_{ii} is the variance of dimension i.
- Σ_{ij} is the covariance of i and j.

1. If $\Sigma_{ij} = 0$ for $i \neq j$, then i and j are statistically independent.
2. If $\Sigma_{ij} = 0$ for $i \neq j$, then the multi-variate normal distribution degenerates to the product of k normal distributions.

It can often be used to define a decision function taking into account the different scaling of the various dimensions and theri co-variance.

$$d_m(x-\mu) = (x-\mu)^T \Sigma^{-1} (x-\mu)$$

Data Whitening Is a linear transformation with a whitetning matrix W, that transforms a vector of random variables X with a known covariance matrix into a nex vector Y whose covariance matrix is the identity matrix, meaning the data are uncorrelated and each have a variance of 1.

Eigenvalues decomposition:

$$\Sigma = U \Lambda U^T$$

Where U is a matrix whose column are formed by the eigenvectors and Λ is a diagonal matrix with the corresponding eigenvalues.

There are infinitely many option of W. Commonly used are ZCA i.e. Mahalanobis whitening;

$$W = U\Lambda^{-\frac{1}{2}}$$

The transformation can be denoted; Wikipedia:

$$Y = WX$$

Alexandros:

$$Y = W^T X$$

The transformations has much resemblance with PCA. It is an linear transformation using eigen vectors. PCA is an orthogonal transformation, thus rotating the data. PCA uses $W = U\Lambda$.

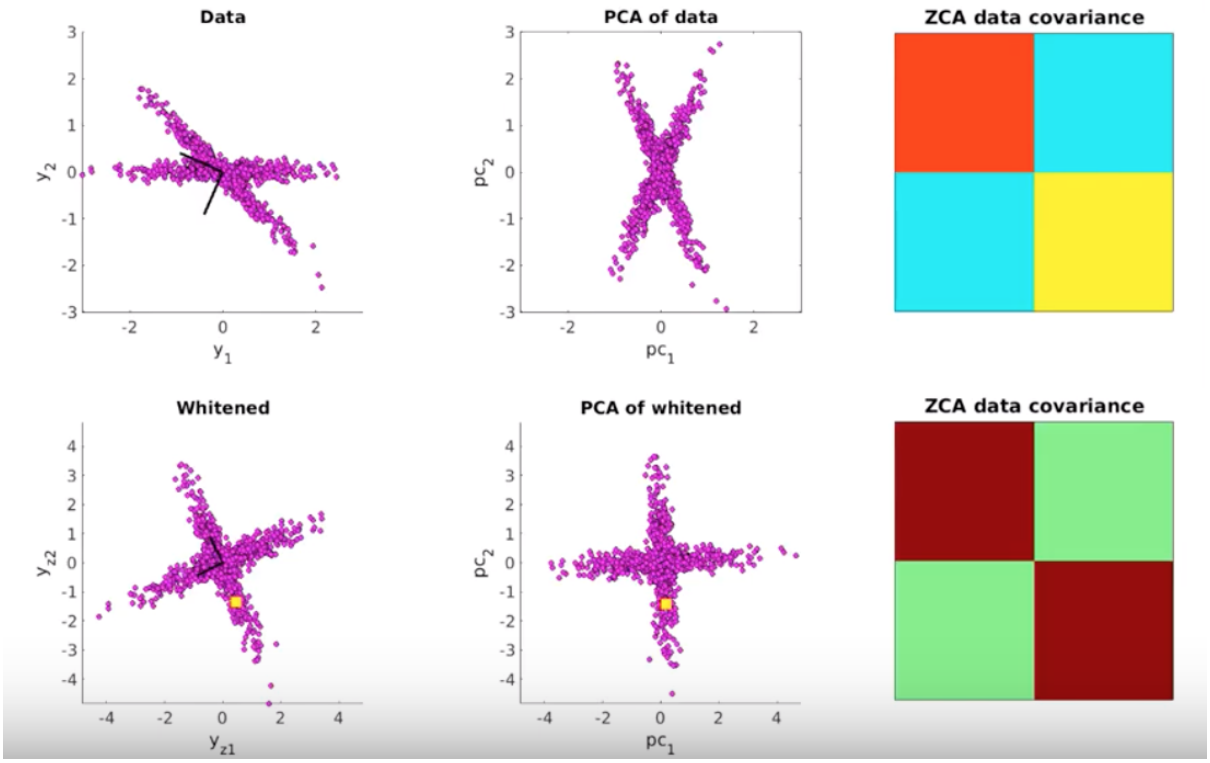

Figure 13: Whitetning

PCA can be used to reduce the dimensionality. Whitening is used to remove correlation in data by shrinking large data direction and expanding small directions. Large data directions tend to reflect low spatial frequencies, thus whitening can increase spatial precision. Whitening separates data by expanding small dimension assuming all the dimensions are of equal importance.

Decision Functions

Consider a two-class classification problem:

$$p(x|c_1) \sim \mathcal{N}(\mu_1, \Sigma_1)$$

$$p(x|c_2) \sim \mathcal{N}(\mu_2, \Sigma_2)$$

Using Bayes' rule, we will;

Decide c_1 if $P(c_1|x) > P(c_2|x)$, else decide c_2.

If $g(\cdot)$ is a monotonic function, then

$$P(c_1|x) > P(c_2|x) \rightarrow g(P(c_1|x)) > g(P(c_2|x))$$

We do this to ease our calculations. We replace using Bayes formula;

Decide c_1 if $P(c_1|x)P(c_1) > P(c_2|x)P(c_2)$, else decide c_2. (note: divide by p(x) dissappears on both sides.) or Decide c_1 if $f(x|c_1) > f(x|c_2)$, else decide c_2.

We simplify the computation of the exponential function by $g(x) = ln(x)$, then we have;

$$f(x|c_k) = -\frac{1}{2}(x-\mu)^T\Sigma_k^{-1}(x-\mu) - \frac{D}{2}ln(2\pi) - \frac{1}{2}ln(|\Sigma_k|) + ln(P(c_k))$$

Special Case In case $\Sigma_k = \sigma^2 I$, the deteminant $|\Sigma_k| = \sigma^{2D}$ and the inverse $\Sigma_k^{-1} = \frac{1}{\sigma^2}I$. The decision rule then becomes;

$$-\frac{1}{2\sigma^2}(x-\mu_1)^T(x-\mu_1) + ln(P(c_1)) > -\frac{1}{2\sigma^2}(x-\mu_2)^T(x-\mu_2) + ln(P(c_2))$$

Discrete Values Integrals become summation;

$$\int p(x|c_k) \rightarrow \sum_x P(x|c_k)$$

Bayes' formula involves propalities instead of propability densities;

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

Where

$$P(x) = \sum_{k=1}^{K} P(x|c_k)P(c_k)$$

Maximum Likelihood Estimation

We assume our collection of samples to be of a gaussian distribution and that each training sample drawn are idenpendent identical distributed (i.i.d.) random variables.
Suppose the training set D contains n samples, and θ is the sample we wish to estimate. The likelihood is defined as;

$$p(D|\theta) = \prod_{k=1}^{n} p(x_k|\theta)$$

The estimate $\hat{\theta}$ is by definition the value, that maximizes $p(D|\theta)$.

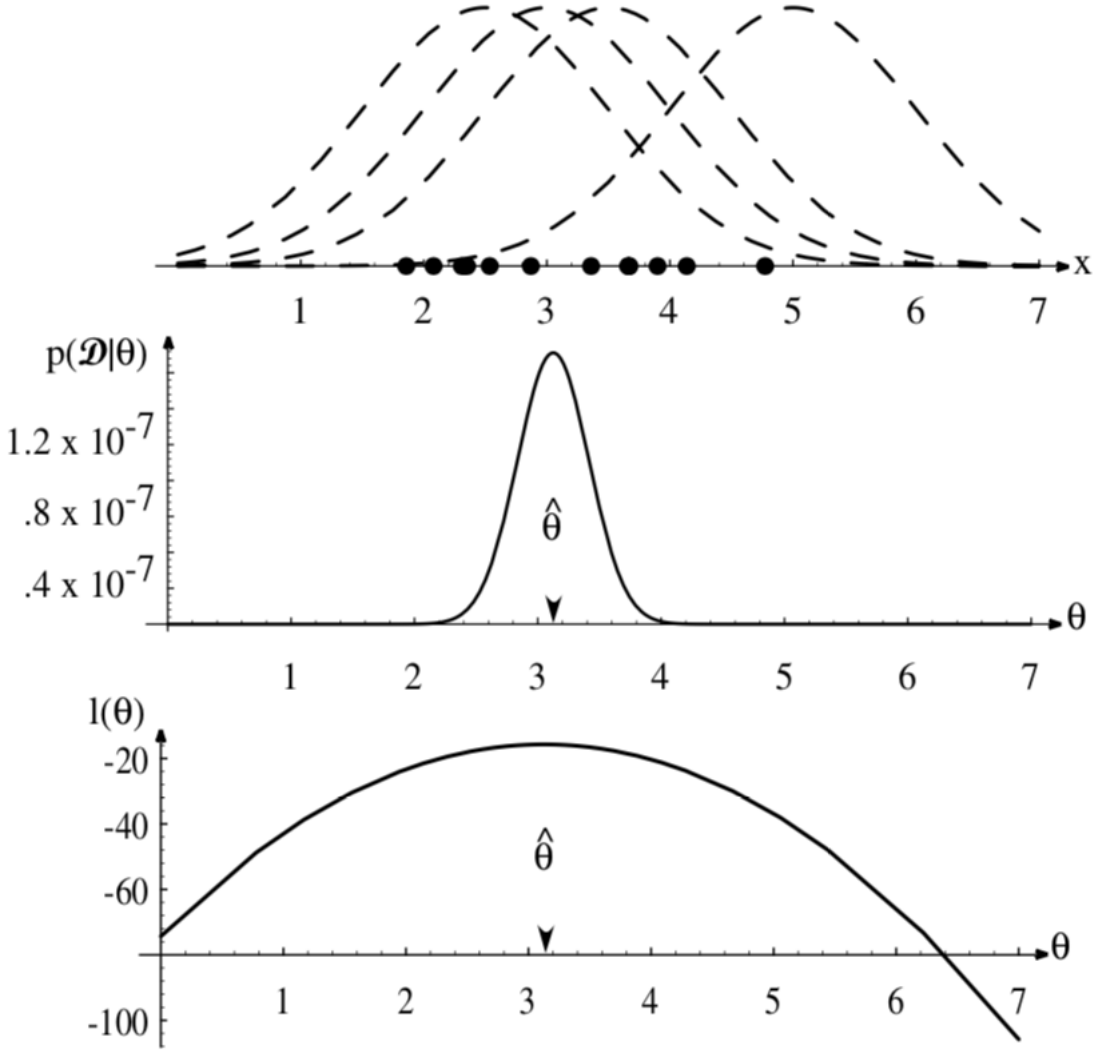

Figure 14: likelihood

Figure shows all the possible distributions of the sampled data, an optimal solution and the same optimal solution option by the log-likelihood.

Taking the natural logarithm generate a monotonic function, that simplifies computations. The operation can be written as;

$$\hat{\theta} = \arg\max_{\theta} ln(p(D|\theta))$$

It can be derived, that the mean μ and standard deviation σ can be estimated from the collection by the general formulas.

Univariate

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{k=1}^{n}(x_k - \mu)^2$$

Multivariate

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{k=1}^{n}(x_k - \mu)^2$$

Typically we do not know these variables and must estimate them.

Baysian Estimation

We do not seek a true value for θ, but a random variable instead. Training allow us to convert a distribution into a posterior probability density.

Nearest Prototype Classifiers

Nearest Centroid

The nearest centroid classifier belong to the class of similarity or distance based algorithms. The model is trained by computing the centroid i.e. mean vector of each class based on the training data. The model is tested by computing the distance for unlabelled samples to the centroids and assigning the sample to the class of the nearest centroid. The are different distances measures to use e.g. Euclidean,

Manhattan, Minkowski, etc. The most typical one is the euclidean distance. The euclidean distance is also defined as the L2 norm and is the straight line distance between to points.

K-nearest neighbor

K-Nearest Neighbors is a rather simple but a computational expensive algorithm. The training phase is simply just storing the multi-dimensional training samples. The test phase on the other hand is expensive. It takes unlabelled sample vectors and computes the distance to all the samples in the model, and assigns the label that is most frequent among the k nearest samples.

Linear Methods

Fischer Discriminant Analysis

Is a linear projection, that seeks to maximize the discrimination of two classes. It has a lot of similarities with PCA, but instead of creating component that maximizes the total variance, LDA creates component, that projects onto the surface of the distance between two class means.

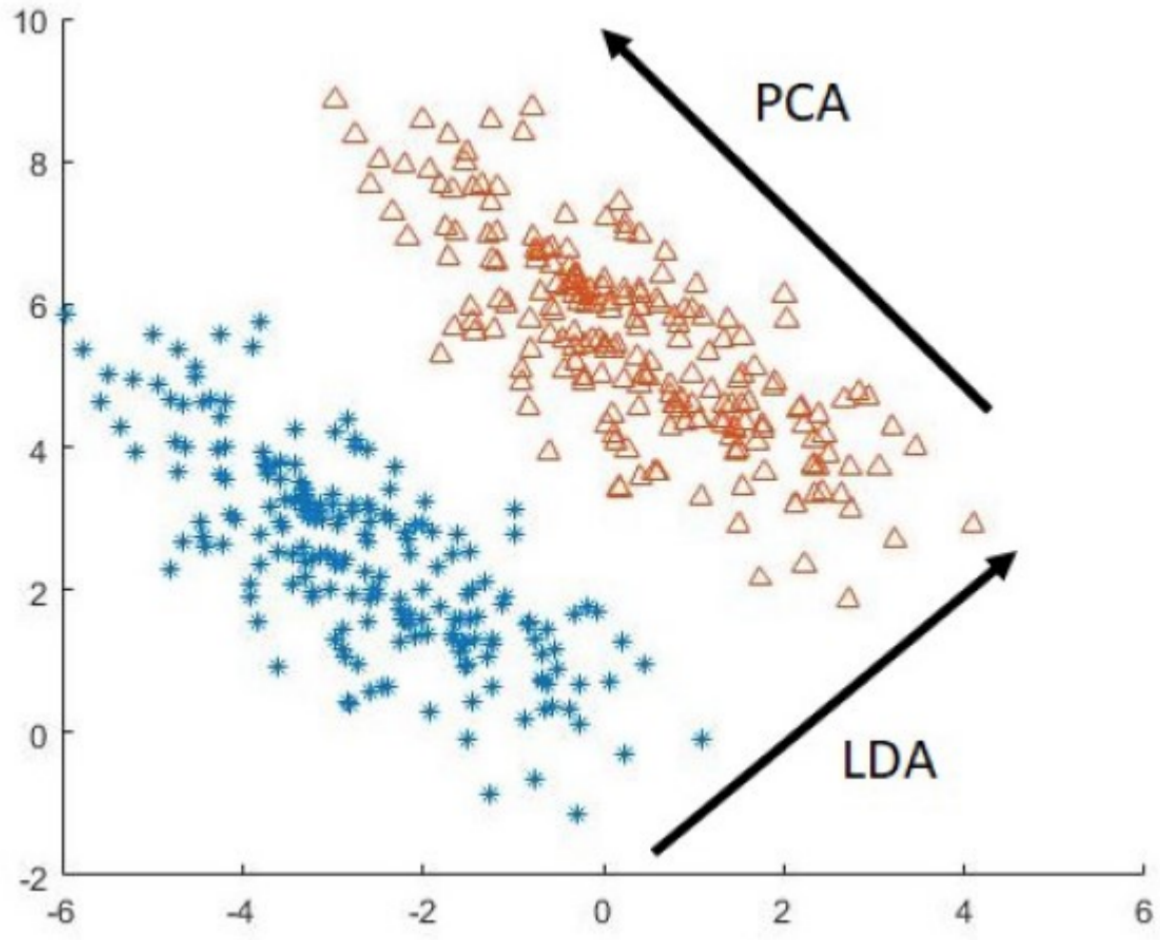

Figure 15: LDA vs. PCA

Linear Discriminant Analysis

Extends Fischer LDA into multi-class classification i.e. more than two classes.

Linear Discriminant Functions (Rules)

Maximum likelihood: Assigns x to the group that maximizes population (group) density.

Bayes Discriminant Rule: Assigns x to the group that maximizes $\pi_i f_i(x)$, where π_i represents the prior probability of that classification, and $f_i(x)$ represents the population density.

Fisher's linear discriminant rule: Maximizes the ratio between $SS_{between}$ and SS_{within}, and finds a linear combination of the predictors to predict group.

Generalized Discriminant Function

Is a classification technique for cases, that are non-linear separable. It is an augmented version of the latter. It uses feature expansion i.e. augmentation of the feature vector to create a higher dimensional representation, where the data is separable. It then create a decision hyperplane, and projects it back down to lower dimensions. The result is non-linear decision function.

Perceptron

A perceptron is a single layer neural network and the simplest form of a neural network. It takes a feature vector as input and produces a classification as output. It does so by multiplying each feature by a weight. The multiplied features are summed up to a weighted sum, that is given to the activation function. The activation function produces the output of the perceptron. The are different classification function e.g. the binary step function, that either produces a 1 or a 0 depending on the input. The training phase is an iterative approach to update the weights of the inputs by evaluating the response with a cost function. The cost function in our case is the mean squared error. The learning or training phase is an optimization process to minimize this cost function. The optimization function is a stochastic steepest gradient descent algorithm. The algorithm iteratively calculates the steepest decent from a given point and converges when finding minimas.

Kernel-Based Learning

Support Vector Machines

Support vector machine is a technique to create a decision boundary by a hyperplane discriminating classes. The support vector machine tries to create a hyperplane, that maximizes the margin between classes.

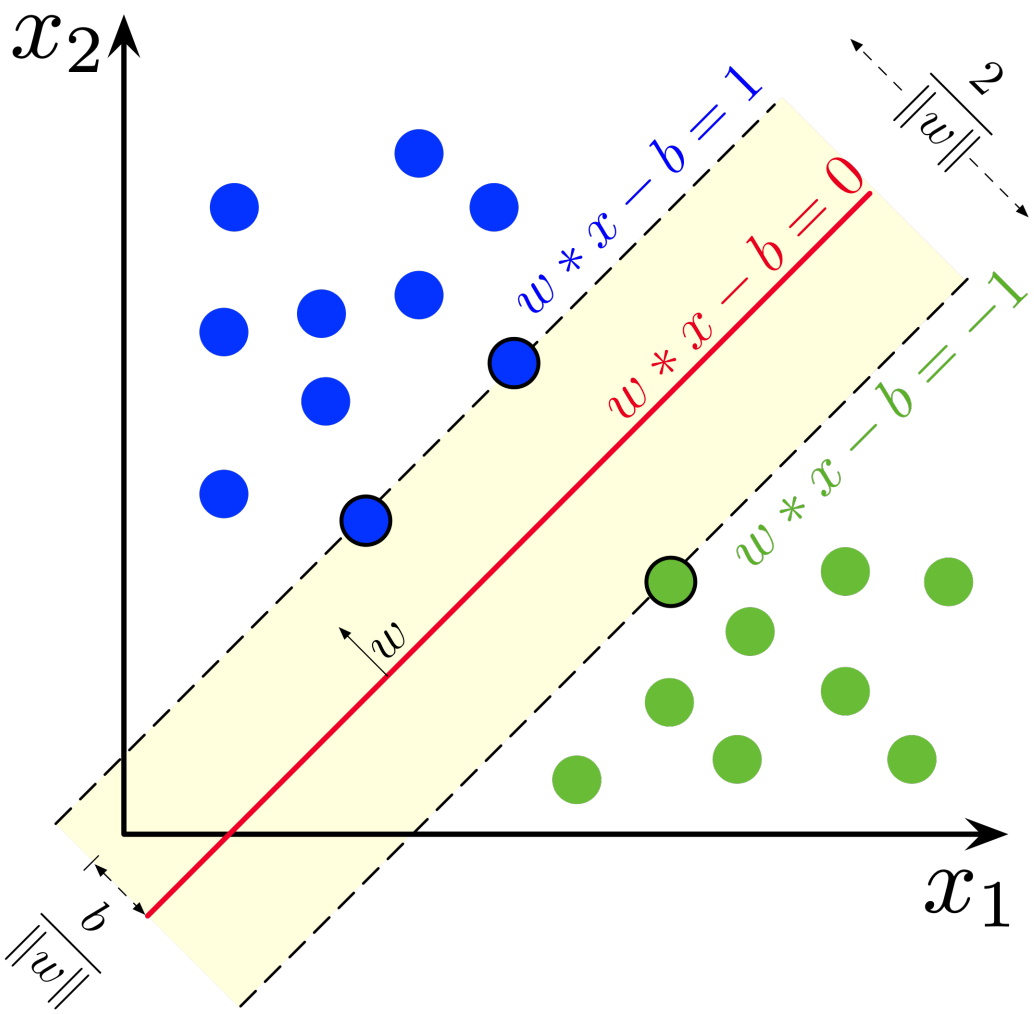

Figure 16: Support Vector Machine

Least-Mean-Square Regression

Kernel Discriminant Analysis

The kernel tricks is a lightweight procedure to perform, what feature expansion to higher dimensions does, however without ever transforming the data. We take advantage of the existence of some specialized functions, that allow us to compute the inner product of two vectors in higher dimension.

Neural Networks

Multi-layered Neural Networks are a widely used supervised learning technique. It extends the perceptron by adding hidden layers between the input layer and the output layer. All nodes in the hidden layers have a weight and activation function just as the single layer perceptron. The training phase consists of two sub-phases; feedforward and Back-propagation.

Feedforward is the process of each node in a layer passing its result to the next layer in the network until it reaches the output layer. The error is then computed using a loss function.

Backpropagation is the process of determining the contribution to the error at each node, as all nodes in the network affect the outcome. The network tries to optimize the error or cost function by updating the weight at each layer. This is done repeatedly until the network converges to a state where all training data has been correctly classified.

The testing phase is inputting unlabelled data to the network's input layer. The layers then apply their effect and the outcome is the prediction.

Selecting number of neurons in hidden layers rules of thumb;

> - The number of hidden neurons should be between the size of the input layer and the size of the output layer.
> - The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
> - The number of hidden neurons should be less than twice the size of the input layer.
>
> *– Jeff Heaton*