Classification

Learn from a dataset D an approximation of function f(x) that maps input x to a discrete class C_k (with k=1,...,k)

$$D = \{\langle x, C_k \rangle\} \Rightarrow C_k = f(x)$$

The target as to be encoded in a number, we need to model f, we need to encode C_k and we need to optimise the approximation.
I can find my target wit or without probabilistic approaches.

- Discriminant function
  - Model a parametric function that maps input to classes
  - Learn parameters from data
- Probablistic discriminative approach
  - Design a parametric model of $p(C_k|\mathbf{x})$
  - Learn model parameters from data
- Probabilistic generative approach
  - Model $p(\mathbf{x}|C_k)$ and class priors $p(C_k)$
  - Fit models to data
  - Infer **posterior** with Bayes' rule: $p(C_k|\mathbf{x}) = p(\mathbf{x}|C_k)p(C_k)/p(\mathbf{x})$

The second probabilistic approach is really difficult but if you can resolve it you can also generate new data points.

Discriminant Function

Function that map the input in one of the classes.
I cannot use a plain linear function. So we will use a generalized linear models:

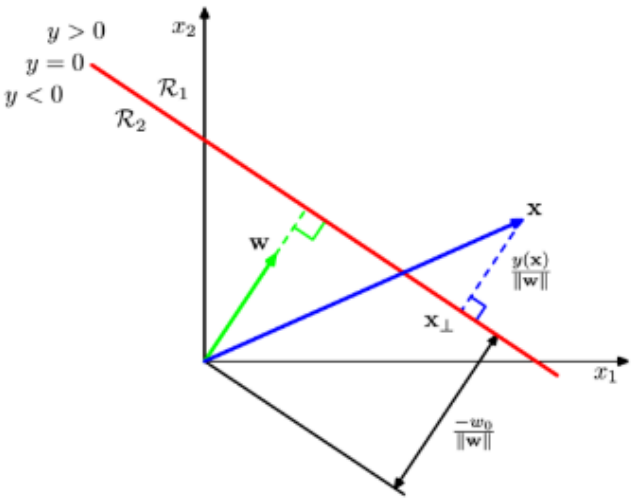
$$f(\mathbf{x}, \mathbf{w}) = f(\omega_0 + \sum_{j=1}^{D-1} \omega_j x_j) = f(\mathbf{x}^T\mathbf{w} + \omega_0)$$

- f(.) is not linear in w and it allows me to maps the inputs in the way I want to encode the class.
- f(.) partitions the input space into decision regions whose boundaries are called decision boundaries or decision surfaces
- these decision surfaces are linear function of x and w as they correspond to $\mathbf{x}^T\mathbf{w} + \omega_0 = const$
- generalized linear models are more complex to use with respect to linear models(both from a computational and analytical perspective)

Discriminant linear function for a two-class problem

$$f(\mathbf{x}, \mathbf{w}) = \begin{cases} C_1, & \text{if } \mathbf{x}^T\mathbf{w} + w_0 \geq 0 \\ C_2, & \text{otherwise} \end{cases}$$

- Properties
  - DS is $y(\cdot) = \mathbf{x}^T\mathbf{w} + w_0 = 0$
  - DS is orthogonal to $\mathbf{w}$
  - distance of DS from origin is $-\frac{w_0}{\|\mathbf{w}\|_2}$
  - distance of $\mathbf{x}$ from DS is $\frac{y(\mathbf{x})}{\|\mathbf{w}\|_2}$



This function is basically a linear function that gives a result regarding the input. If we call the argument of the model y(), by definition y()= 0 will be the equation of my decision boundary. It is a linear separator.
How can I use this function in a multiple classes problem?

1. One-versus-the-rest approach
2. One-versus-one

A simple solution for multiple classes

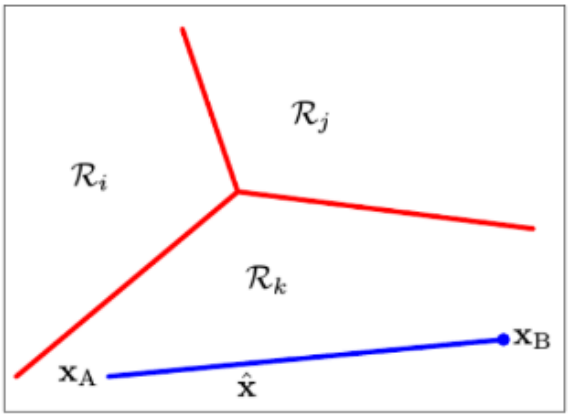
- A possible solutions is to use K linear discriminant functions:

$$y_k(\mathbf{x}) = \mathbf{x}^T\mathbf{w}_k + w_{k0}, \text{ where } k = 1, \ldots, K$$

  - Map $\mathbf{x}$ to class C_k if $y_k > y_j \quad \forall j \neq k$
  - No ambiguity
  - DS are **singly connected** and **convex**

Let $\mathbf{x}_A, \mathbf{x}_b \in \mathcal{R}_k$
Thus, $y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A)$ and $y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B)$

$\Rightarrow \forall\alpha\ (0 < \alpha < 1)$
$y_k(\alpha\mathbf{x}_A + (1-\alpha)\mathbf{x}_B) > y_j(\alpha\mathbf{x}_A + (1-\alpha)\mathbf{x}_B)$



Linear Basis Function Models

So far we considered models that work in the input space (i.e., with x). However, we can still extend models by using a fixed set of basis function $\phi(x)$. Basically, we apply a non-linear transformation to map the input space into a feature space. As a result, decision boundaries

that are linear in the feature space would correspond to nonlinear boundaries in the input space. This allows to apply linear classification models also to problem where samples are not linearly separable.
Some useful resolution can come from:

- Perceptron is a linear discriminant model proposed by Rosenblatt in 1958
  along with a sequential learning algorithm. Perceptron is devised for a two-class problem, where classes encoding is {-1,1}

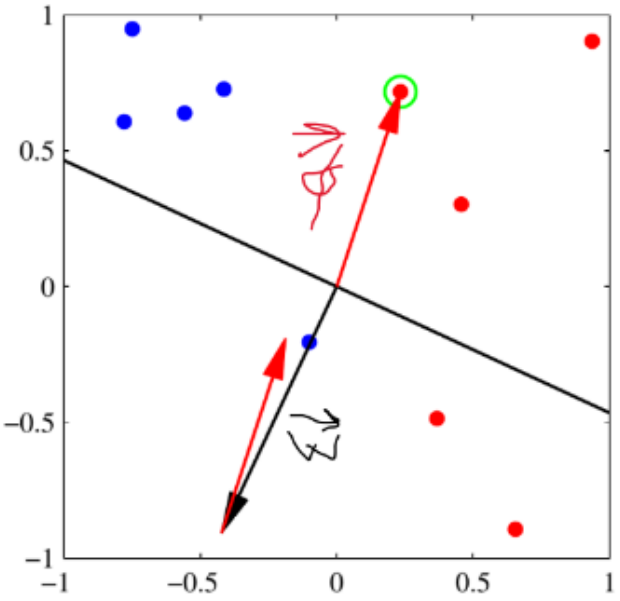
$$f(\mathbf{x}, \mathbf{w}) = \begin{cases} +1 \text{ if } \mathbf{w}^T\phi(\mathbf{x}) \geq 0 \\ nbsp; \quad nbsp; \quad nbsp; -1 \text{ otherwise} \end{cases}$$

  The idea is to optimise w to find a model that is perfect.
  Use as the loss the total sum of the misses points. The entire sum is going to be a positive matrix that I want to minimise/optimise.

Perceptron Algorithm

Given $\mathcal{D} = \{\mathbf{x}_i, \mathbf{t}_i\}$, where i=1,...,N

Initialize $\mathbf{w}_0$
k ← 0
repeat
  k ← k+1
  n ← k mod N
  if $\widehat{t_n} \neq t_n$ **then**
    $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + \boldsymbol{\phi}(\boldsymbol{x_n})t_n$
  endif
until convergence



I repeat until I don't have misplaced points anymore.
A property of this algorithm is that if there is a linear separation boundary in your data this algorithm will find it eventually(no guarantee on the number of steps needed to converge). This algorithm find one linear boundary separation depending on the initialization of the algorithm and the order of the scanning during the update.
Every update I am reducing the loss.

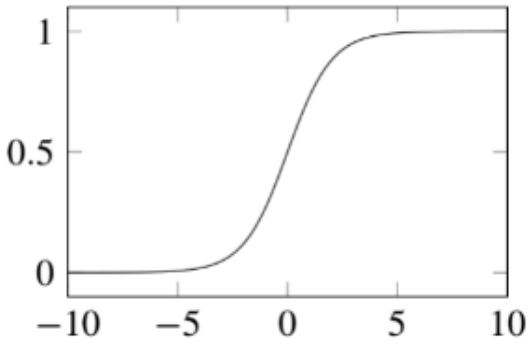
Probabilistic Discriminative Approaches

The new approach is a deterministic model. The idea is to model the problem with a probabilistic view, the probability of a point to be positive/negative. We have a probability to have a parametrized model to have a loss function to identify our problem.
It is not a regression model.

- In a discriminative approach we model directly the conditioned class probability:

$$p(C_1|\boldsymbol{\phi}) = \frac{1}{1+\exp(-\mathbf{w}^T\boldsymbol{\phi})} = \sigma(\mathbf{w}^T\boldsymbol{\phi})$$

  - where $\sigma(a) = 1/(1+\exp(-a)$ is the sigmoidal function
  - $\mathrm{p}(C_2|\boldsymbol{\phi}) = 1 - p(C_1|\boldsymbol{\phi})$
  - this model is knonw as **logistic regression**



We don't really use the logistic of the input but we are using a linear function as the input of the logistic model. I can quantify the probability.
I can fit to the data with a Maximum Likelihood and we compute the probability of seeing the data in our set. We then maximise the probability in regards of the weight vector. We just need to focus on a single point and then using this probability to compute on the other data.

Thus the gradient of the loss function is:

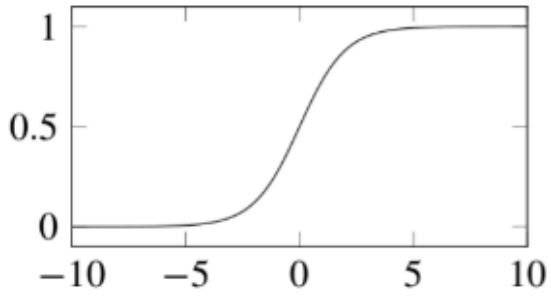
$$\nabla L(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\boldsymbol{\phi}_n$$

- due to the nonlinear logistic regression function it is not possible to find a closed-form solution
- however, the error function is **convex** and **gradient-based optimization** can be applied (also in an **online learning setting**)

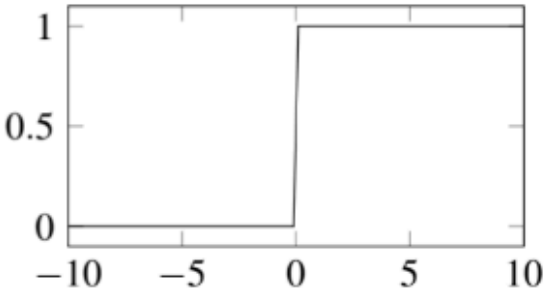
It works for a binary probabilistic problem but can be easily extended to a k-class problem as you only have to compute the probability of be a problem of the k-class.
I learn a weight vector for every class. i don't have two possible probability but I have to choose one on the key.

Logistic Regression and Perceptron Algorithm

- If we replace the logistic function with a step function...



$$y(\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x})}}$$



$$y(\mathbf{x}, \mathbf{w}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) > 0 \\ 0 & \text{otherwise} \end{cases}$$

- ... logisitc regression leads to the same **updating rule** of perceptron algorithm:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \left(y(\mathbf{x}_n, \mathbf{w}) - t_n \right) \boldsymbol{\phi}_n$$

The two techniques are exactly the same, the perceptron is a logistic model where we use the logistic function instead of the step function.