Classification

- Using training data $(x_1, y_1), \dots, (x_n, y_n)$, construct a decision rule f that predict the label $\hat{y}$ of a new data point x
- We saw that the logistic classifier is linear:

$$\hat{y} = \mathrm{sign}(\langle \hat{\theta}, x \rangle + \hat{b})$$

  where $\mathrm{sign}\, z = 1$ if $z > 0$ and $\mathrm{sign}\, z = -1$ if $z < 0$, and where $\hat{\beta} \in \mathbb{R}^d$ and $\hat{b} \in \mathbb{R}$ is the intercept
- The aim of linear classification is to find $\hat{\theta}$ and $\hat{b}$ using the training data

Linearly separable data

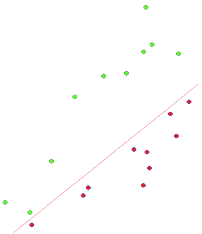
- Data is **linearly separable** if we can find a classification rule that does not make any error on training data.
- Namely, can we find θ and b such that

$$y_i(\langle \theta, x_i \rangle + b) \geq 0 \quad \text{for all} \quad i = 1, \ldots, n$$

- **Strict linear separability** is when can we find θ such that

$$y_i(\langle \theta, x_i \rangle + b) > 0 \quad \text{for all} \quad i = 1, \ldots, n$$

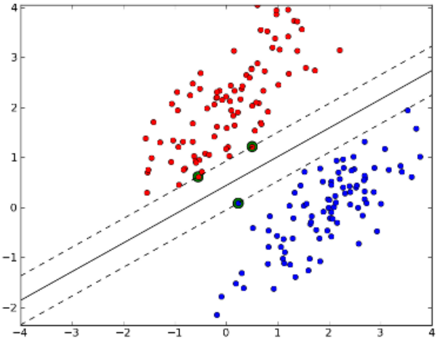
The hyperplane $\{x : \langle x, \theta \rangle + b = 0\}$ separates -1 and $+1$



Given strict linear separability, we can rescale θ and b and consider

$$y_i(\langle \theta, x_i \rangle + b) \geq 1 \quad \text{for all} \quad i = 1, \ldots, n$$

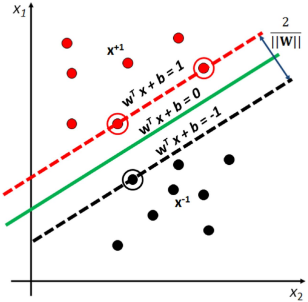
Points with label 1 are contained in the half-space $\langle \theta, x \rangle + b \geq 1$ and those with label -1 are contained in half-space $\langle \theta, x \rangle + b \leq -1$



The distance between the 1's hyperplane $\langle \theta, x\rangle + b = 1$ and the -1's hyperplanes $\langle \theta, x\rangle + b = -1$ is equal to

$$\frac{2}{\|\theta\|_2}.$$

This is called the **margin**. The margin measures how much we can separate the data apart.



Badly or margin data are called the **support vectors**

Slacks

The separability constraint

$$y_i(\langle \theta, x_i \rangle + b) \geq 1 \quad \text{for all} \quad i = 1, \ldots, n$$

is too strong. We relax it a little bit by introducting "slacks":

$$\begin{aligned} \underset{\theta \in \mathbb{R}^d, b \in \mathbb{R}, s_i \geq 0}{\operatorname{argmin}} \quad & \sum_{i=1}^{n} s_i \\ \text{subject to} \quad & y_i(\langle \theta, x_i \rangle + b) \geq 1 - s_i \quad \text{for all} \quad i = 1, \ldots, n \end{aligned}$$

The hinge loss

The problem

$$\begin{aligned} &\underset{\theta\in\mathbb{R}^d, b\in\mathbb{R}, s_i\geq 0}{\text{argmin}} && \sum_{i=1}^{n} s_i \\ &\text{subject to} && y_i(\langle\theta, x_i\rangle + b) \geq 1 - s_i \quad \text{for all} \quad i = 1, \ldots, n \end{aligned}$$

is an optimization problem called "linear programming". It can be interpreted differently:

$$\underset{\theta}{\text{argmin}}\, \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \langle x_i, \theta\rangle + b)$$

where

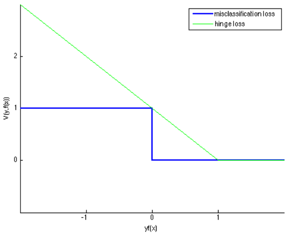
$$\ell(y, z) = (1 - yz)_+ = \max(1 - yz, 0)$$

is the **hinge** loss.

The hinge loss can be understood as a **relaxation** of the $0-1$ error (misclassification error)

$$\ell_{0-1}(y,z) = \mathbf{1}_{yz\leq 0}$$

Impossible to minimize the $0-1$ loss! Hinge loss gives an approximation to the number of errors made on the training set



Remark. Hinge is **convex** while $0-1$ loss is not

A solution to

$$\min_{\theta,b} \frac{1}{n} \sum_{i=1}^{n} \big(1 - y_i(\langle x_i, \theta \rangle + b)\big)_+$$

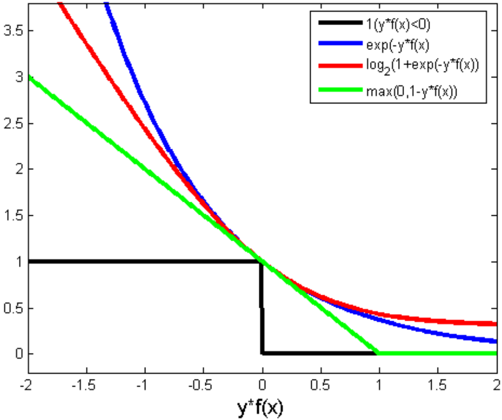
might not be unique, so we add a ridge penalization term:

$$\min_{\theta,b} \frac{1}{n} \sum_{i=1}^{n} \big(1 - y_i(\langle x_i, \theta \rangle + b)\big)_+ + \frac{\lambda}{2} \|\theta\|_2^2$$

where $\lambda > 0$ balances goodness-of-fit (measured by hinge loss) and energy of θ. The solution is now unique.

This is the **SVM** (Support Vector Machine) algorithm. More precisely, it is the **linear SVM**.

The losses we've seen so far for classification



$$\ell_{0-1}(y,z) = \mathbf{1}_{yz\leq 0} \quad \ell_{\text{hinge}}(y,z) = (1-yz)_+$$

$$\ell_{\texttt{logistic}}(y,z) = \log(1+e^{-yz}).$$

Logistic regression vs Linear SVM

Grandmother's recipes:

Logistic regression

- Logistic regression has a nice probabilistic interpretation
- Relies on the choice of the logit link function

SVM

- No model, only aims at separating points

No one is not better than the other in general. It depends on the data.

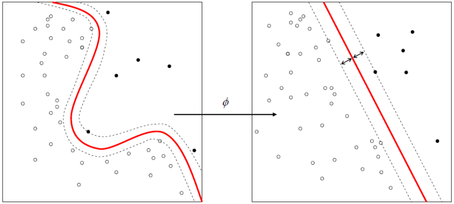
What is always important though is the **choice of the features** we work on

- Given features $x = [x_1, \ldots x_d] \in \mathbb{R}^d$, we can construct many more features.
- For instance, we can add second order polynomials of the features

$$x_j^2, x_j x_k \quad \text{for any} \quad 1 \leq j, k \leq d$$

- It increases the number of features, hence dimension of the parameter θ.
- Consider a **feature map** $\varphi(x)$ that adds all these new features

A decision boundary $\langle \theta, \varphi(x) \rangle + b = 0$ is not an hyperplane anymore



Beyond linear classification

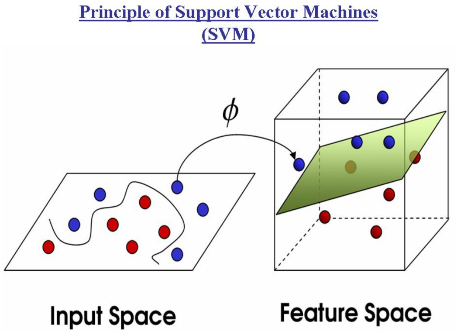
- It is a common belief that we can always **increase the dimension** of the feature space to make data **almost linearly separable**
- A youtube video explains this:

  `https://www.youtube.com/watch?v=3liCbRZPrZA`

You want to solve now

$$\min_{\theta,b} \frac{1}{n}\sum_{i=1}^{n}\left(1 - y_i(\langle\varphi(x_i),\theta\rangle + b)\right)_+ + \frac{\lambda}{2}\|\theta\|_2^2$$

where θ is much larger and where φ is a feature mapping



Kernels

For the polynomial mapping $\varphi : \mathbb{R}^2 \to \mathbb{R}^3$

$$\varphi : x = (x_1, x_2) \mapsto (x_1^2, \sqrt{2}x_1x_2, x_2^2).$$

We have

$$\langle \varphi(x), \varphi(x') \rangle = x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2 = \langle x, x' \rangle^2$$

More generally we can define the "polynomial kernel"

$$K(x, x') = (1 + \langle x, x' \rangle)^d.$$

This kernel satisfies

$$K(x, x') = \langle \varphi(x), \varphi(x') \rangle$$

for some feature mapping φ.

Kernels

- A **kernel** is a "new" inner product. It replaces $\langle x, x' \rangle$ by $\langle \varphi(x), \varphi(x') \rangle$ where φ is a feature mapping
- A kernel K is such that

$$K(x, x') = \langle \varphi(x), \varphi(x') \rangle$$

  for some feature mapping φ

Why is it important: the **kernel trick**. A theorem says that a solution $\hat{\theta}$ to

$$\min_{\theta, b} \frac{1}{n} \sum_{i=1}^{n} \big(1 - y_i(\langle \varphi(x_i), \theta \rangle + b)\big)_+ + \frac{\lambda}{2} \|\theta\|_2^2$$

writes

$$\hat{\theta} = \sum_{i=1} \hat{u}_i \varphi(x_i) + \hat{b}$$

for some "dual" parameters $\hat{u}_1, \ldots, \hat{u}_n \in \mathbb{R}$.

Actually computing a solution to

$$\min_{\theta,b} \frac{1}{n} \sum_{i=1}^{n} \big(1 - y_i(\langle \varphi(x_i), \theta \rangle + b)\big)_+ + \frac{\lambda}{2}\|\theta\|_2^2$$

only requires computing the inner products $\langle \varphi(x_i), \varphi(x_{i'}) \rangle$, hence only requires $K(x_i, x_{i'})$

Given a new $x \in \mathbb{R}^d$, the decision rule is $\langle \varphi(x), \hat{\theta} \rangle + \hat{b} \geq t$, which becomes

$$\sum_{i=1}^{n} \hat{u}_i \langle \varphi(x), \varphi(x_i) \rangle + \hat{b} \geq t$$

Only depends on K again!

Kernels

- Actually we never need to know about the feature mapping φ!
- Only need to be able to know K in the computations
- This is the **kernel trick**

Popular kernels are:

- Gaussian or RBM (Radial Basis Function) kernel:

$$K_\sigma(x, x') = \exp\Big(-\frac{\|x - x'\|_2^2}{2\sigma^2}\Big)$$

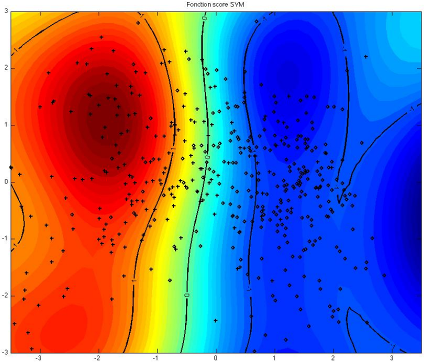
- The polynomial kernel

$$K(x, x') = (\langle x, x' \rangle + 1)^d$$

Many other kernels in specific domains (text, image, video, genomics, etc.), that are adapted to the "geometry" of specific data

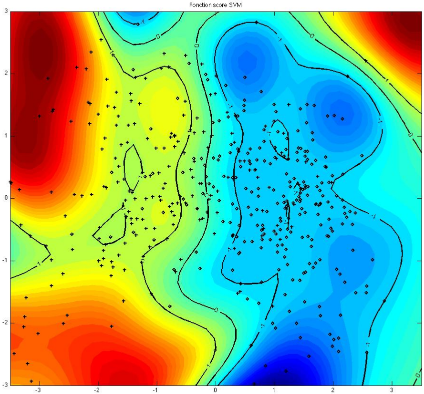
Decision rule with the RBF kernel: mixture of Gaussian functions

$$x \mapsto \sum_{i=1}^{n} \hat{u}_i \exp\left(-\frac{\|x - x_i\|_2^2}{2\sigma^2}\right) \geq t - \hat{b}$$



Decision rule with the RBF kernel: mixture of Gaussian functions

$$x \mapsto \sum_{i=1}^{n} \hat{u}_i \exp\left(- \frac{\|x - x_i\|_2^2}{2\sigma^2} \right) \geq t - \hat{b}$$



Decision rule with the RBF kernel: mixture of Gaussian functions

$$x \mapsto \sum_{i=1}^{n} \hat{u}_i \exp\left(-\frac{\|x - x_i\|_2^2}{2\sigma^2}\right) \geq t - \hat{b}$$