The SVM classifier

Lecture 14 , Oct 22ed, Martin Radfar, CSE391:
Data Science

Be more concerned with your character than
your reputation.
John Wooden

The Slides are either from or adapted from slides and lecture notes
provided by A. Zisserman from Oxford or

http://www.robots.ox.ac.uk/~az/lectures/ml

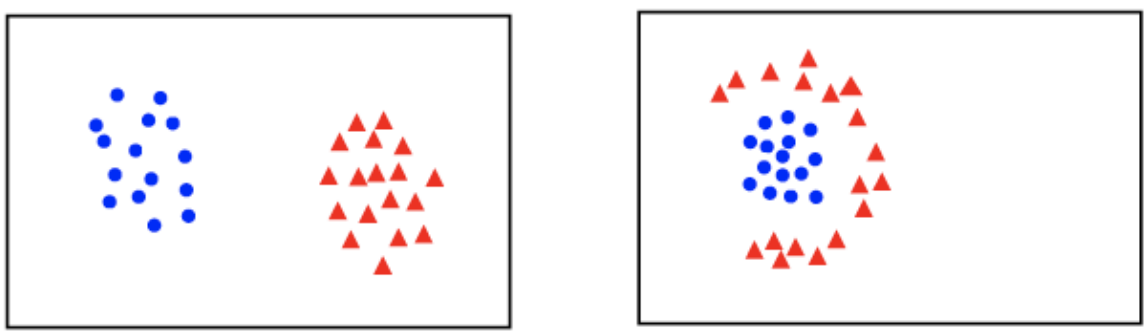
http://cs229.stanford.edu/notes/cs229-notes3.pdf

Binary Classification

Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \dots N$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(\mathbf{x})$ such that

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

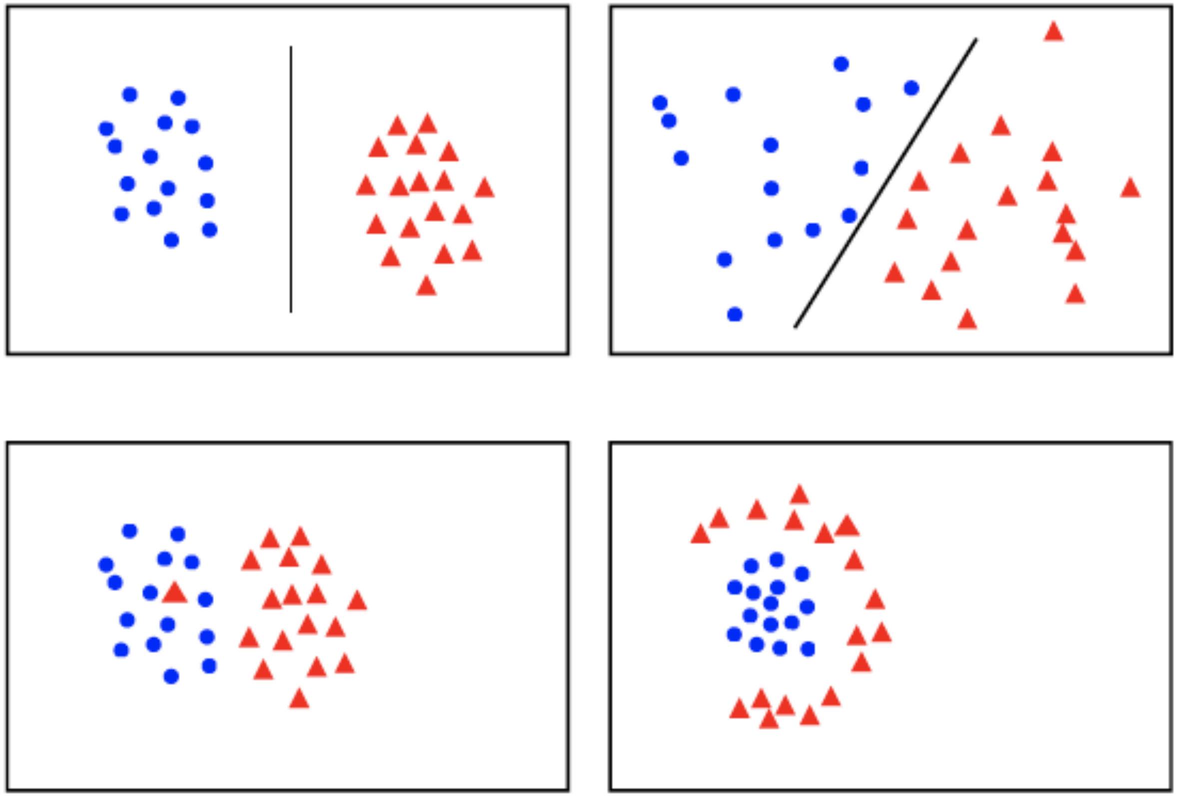
i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification.



Linear separability

linearly
separable

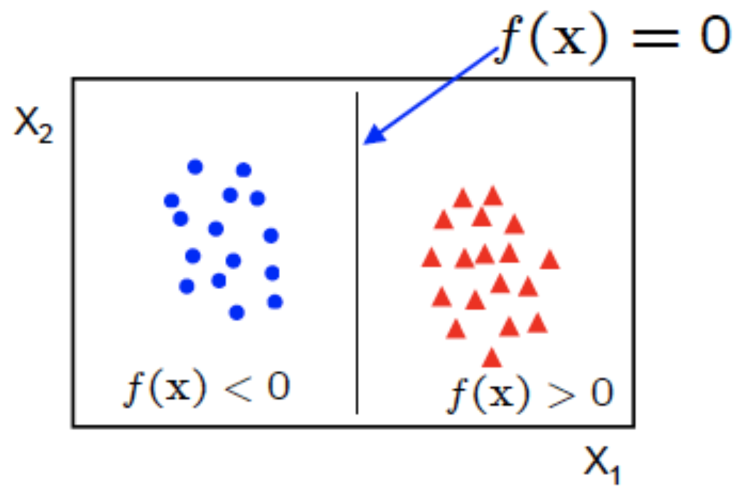
not
linearly
separable



Linear classifiers

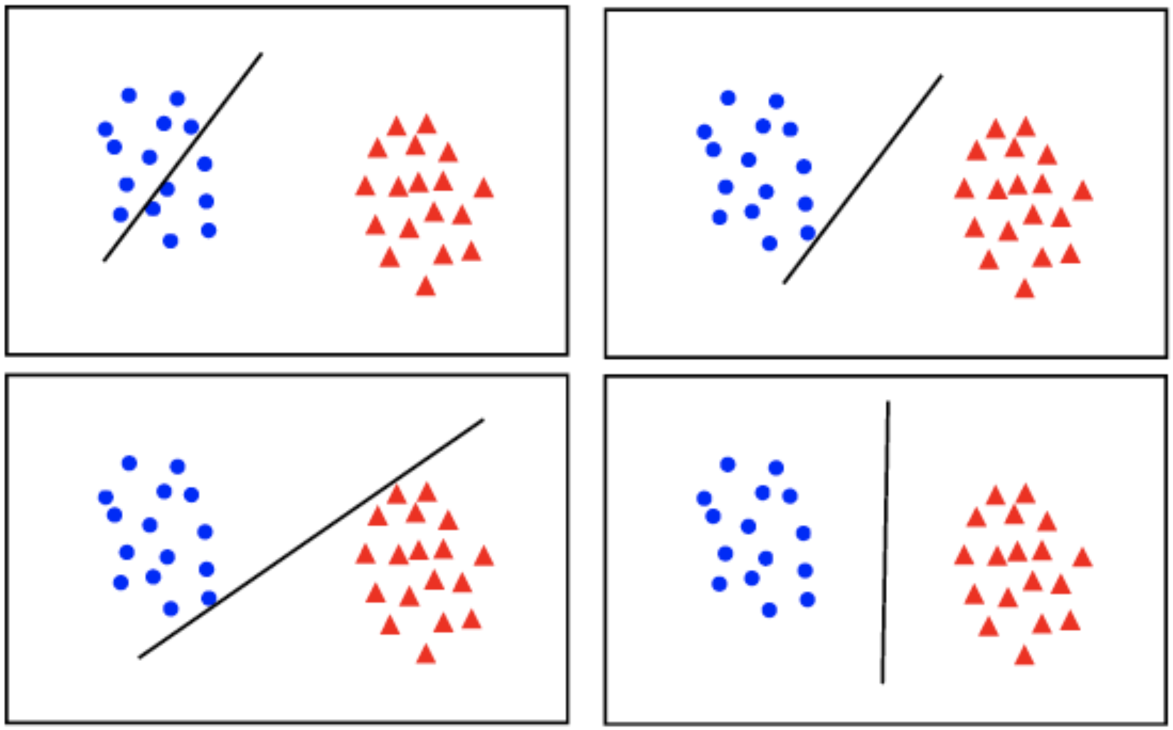
A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



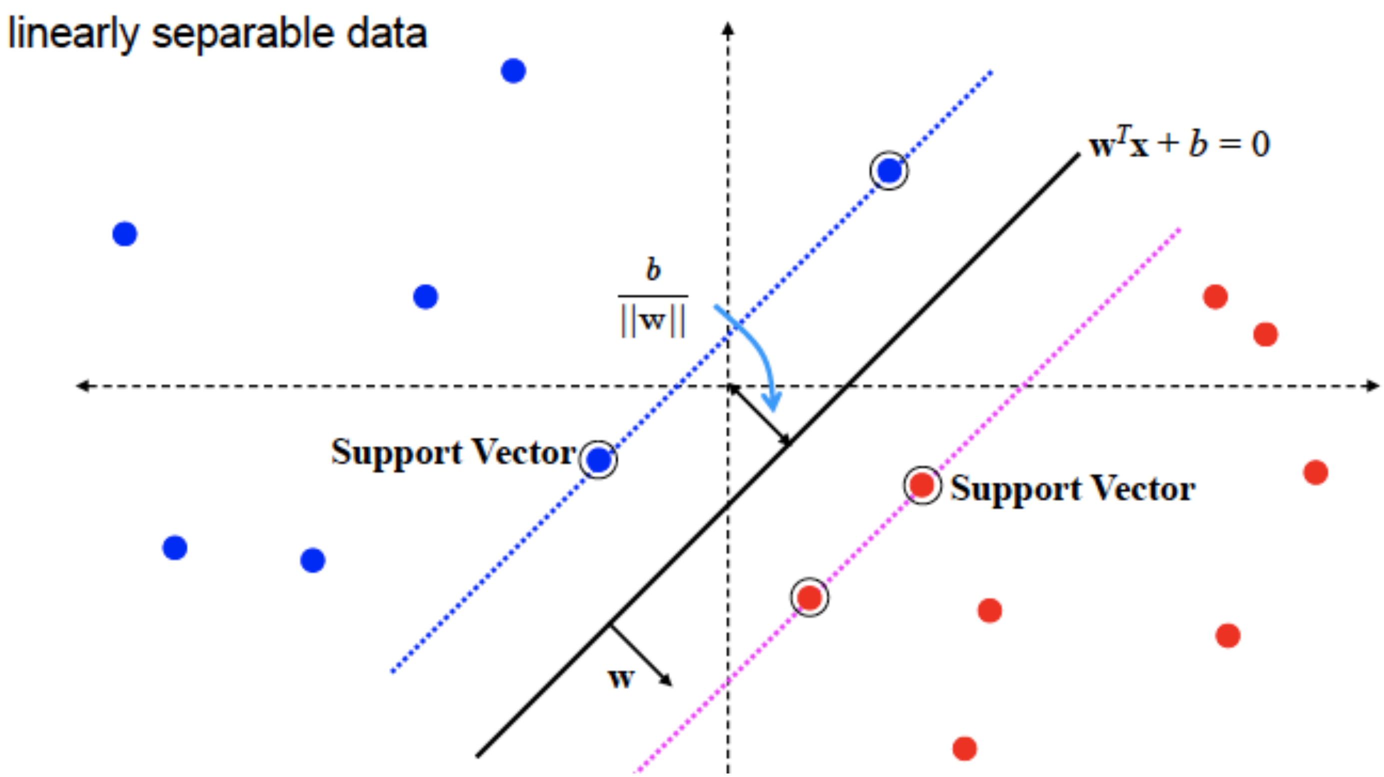
- in 2D the discriminant is a line
- $\mathbf{w}$ is the normal to the line, and b the bias
- $\mathbf{w}$ is known as the weight vector

What is the best w?



• maximum margin solution: most stable under perturbations of the inputs

Support Vector Machine

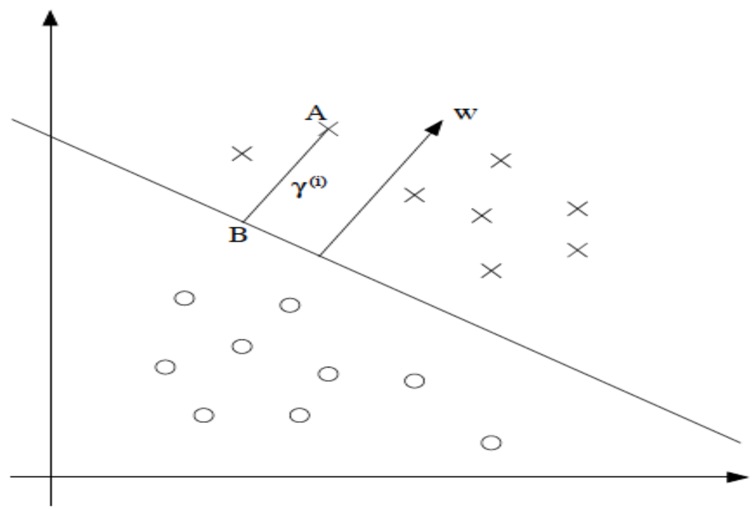


How find the parameters of SVM?

Given a training set $S = \{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$, we also define the function margin of (w, b) with respect to S to be the smallest of the functional margins of the individual training examples. Denoted by $\hat{\gamma}$, this can therefore be written:

$$\hat{\gamma} = \min_{i=1,\ldots,m} \hat{\gamma}^{(i)}.$$

Next, let's talk about **geometric margins**. Consider the picture below:



More generally, we define the geometric margin of (w, b) with respect to a training example $(x^{(i)}, y^{(i)})$ to be

$$\gamma^{(i)} = y^{(i)} \left(\left(\frac{w}{||w||} \right)^T x^{(i)} + \frac{b}{||w||} \right).$$

How find the parameters of SVM?

For now, we will assume that we are given a training set that is linearly separable; i.e., that it is possible to separate the positive and negative examples using some separating hyperplane. How we we find the one that achieves the maximum geometric margin? We can pose the following optimization problem:

$$\begin{aligned} \max_{\gamma,w,b} \quad & \gamma \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq \gamma, \quad i = 1, \ldots, m \\ & ||w|| = 1. \end{aligned}$$

I.e., we want to maximize γ, subject to each training example having functional margin at least γ. The $||w|| = 1$ constraint moreover ensures that the functional margin equals to the geometric margin, so we are also guaranteed that all the geometric margins are at least γ. Thus, solving this problem will result in (w, b) with the largest possible geometric margin with respect to the training set.

Since multiplying w and b by some constant results in the functional margin being multiplied by that same constant, this is indeed a scaling constraint, and can be satisfied by rescaling w, b. Plugging this into our problem above, and noting that maximizing $\hat{\gamma}/||w|| = 1/||w||$ is the same thing as minimizing $||w||^2$, we now have the following optimization problem:

$$\begin{aligned}\min_{\gamma,w,b} \quad & \frac{1}{2}||w||^2 \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \ldots, m\end{aligned}$$

We've now transformed the problem into a form that can be efficiently solved. The above is an optimization problem with a convex quadratic objective and only linear constraints. Its solution gives us the **optimal margin classifier**. This optimization problem can be solved using commercial quadratic programming (QP) code.[1]

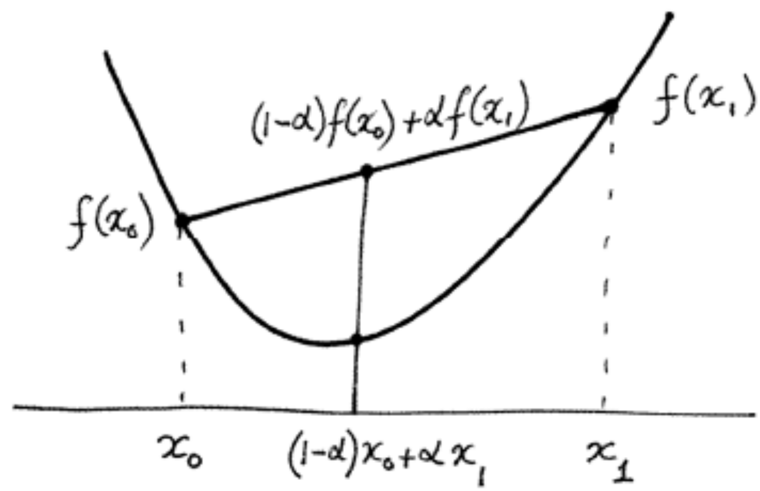
Convex functions

D – a domain in $\mathbb{R}^n$.

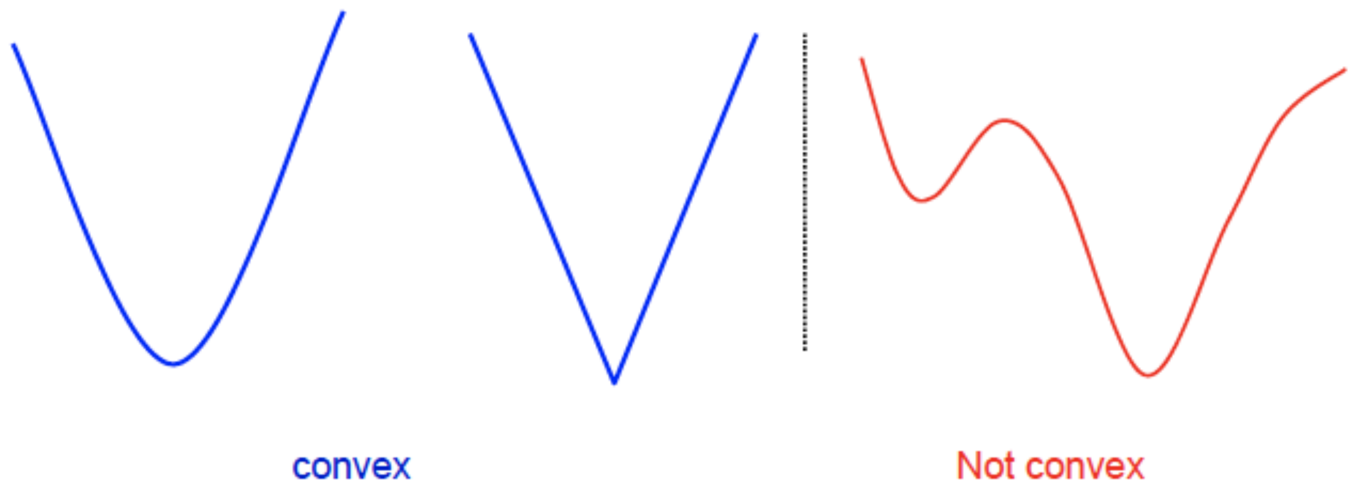
A convex function $f : D \to \mathbb{R}$ is one that satisfies, for any $\mathbf{x}_0$ and $\mathbf{x}_1$ in D:

$$f((1-\alpha)\mathbf{x}_0 + \alpha\mathbf{x}_1) \leq (1-\alpha)f(\mathbf{x}_0) + \alpha f(\mathbf{x}_1) \ .$$

Line joining $(\mathbf{x}_0, f(\mathbf{x}_0))$ and $(\mathbf{x}_1, f(\mathbf{x}_1))$ lies above the function graph.

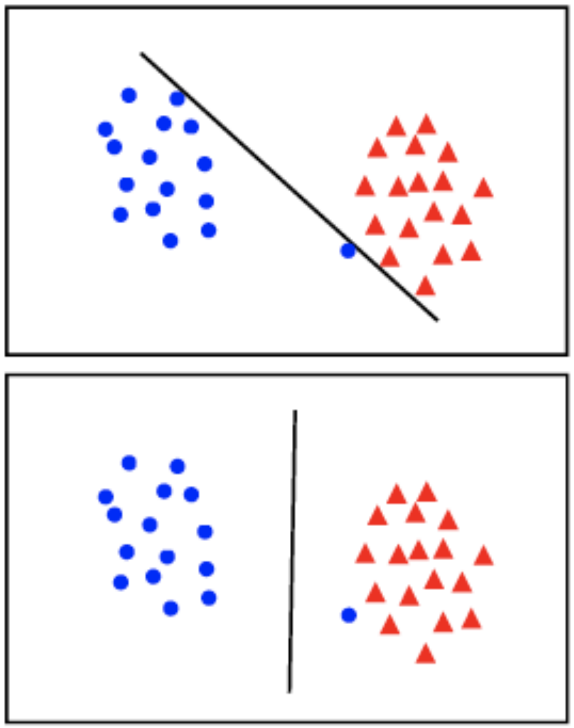


Convex function examples



A non-negative sum of convex functions is convex

Linear separability again: What is the best w?



• the points can be linearly separated but there is a very narrow margin

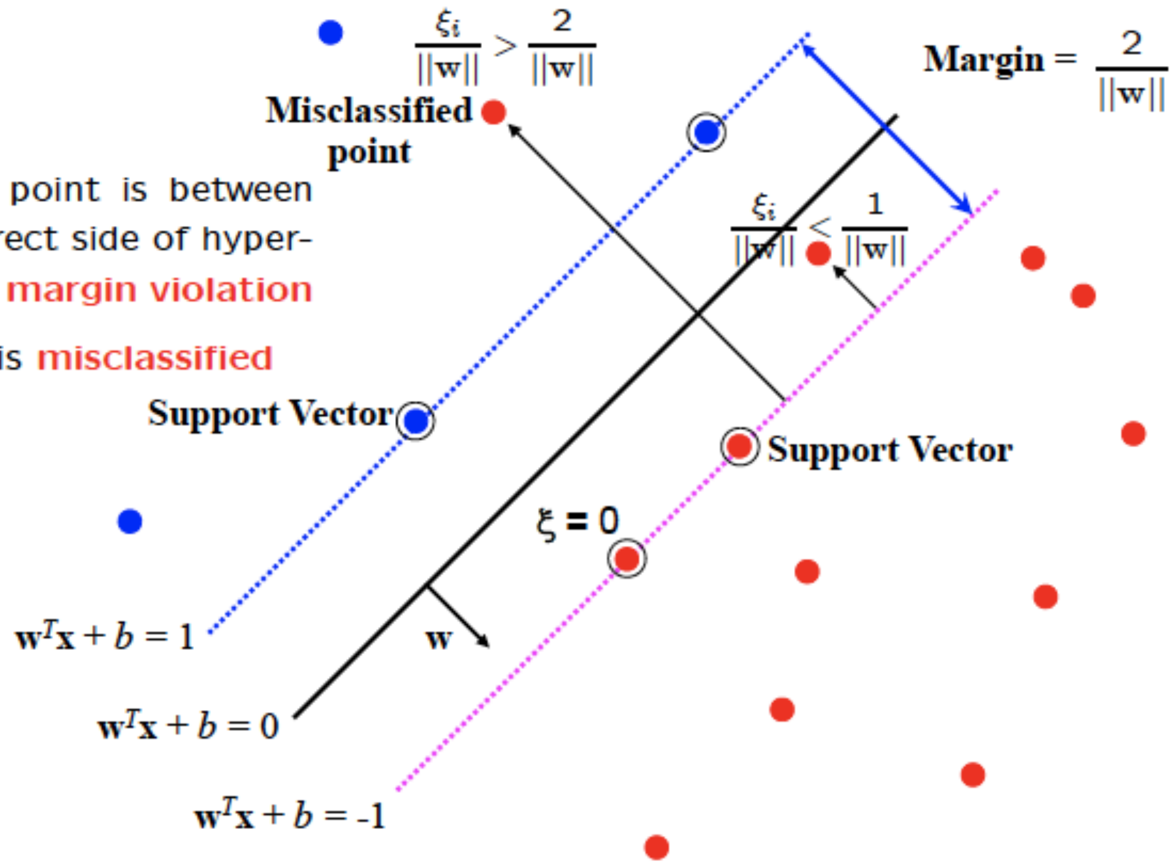
• but possibly the large margin solution is better, even though one constraint is violated

In general there is a trade off between the margin and the number of mistakes on the training data

Introduce "slack" variables

$$\xi_i \geq 0$$

- for $0 < \xi \leq 1$ point is between margin and correct side of hyper-plane. This is a margin violation
- for $\xi > 1$ point is misclassified



What is the purpose for using slack variable in SVM?

- The standard SVM classifier works only if you have a well separated categories. To be more specific, they need to be *linearly separable.* It means there exist a line (or hyperplane) such that all points belonging to a single category are either below or above it. In many cases that condition is not satisfied, but still the two classes are pretty much separated except some small training data where the two categories overlap. It wouldn't be a huge error if we would draw a line (somewhere in between) and accept some level of error - having training data on the wrong side of the marginal hyperplanes. **How do we measure the error?**The answer is: *slack variables*. For each training data point we can define a variable that measures the distance of the point to its *marginal hyperplane*(dahsed line in the figure), lets call it $\xi{*}i$. Whenever the point is on the wrong site of the marginal hyperplane we quantify the amount of error by the ratio between $\xi{*}i$ and half of the margin, i.e. distance between separating hyperplane and marginal hyperplane (M in the figure). Points on the correct site are not quantified as errors. This is a geometrical interpretation of slack variables ξi. You can now go back to the initial SVM problem and maximize the margin in the presence of errors. The larger the error that you allow for, the wider the margin (numerical illustration at the end.

- SRC : [Dariusz Kajtoch]

The optimization problem becomes

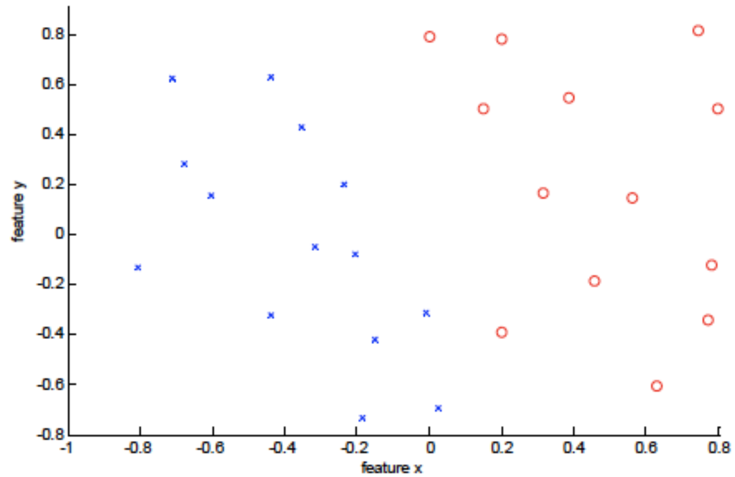
$$\min_{\mathbf{w}\in\mathbb{R}^d, \xi_i \in \mathbb{R}^+} ||\mathbf{w}||^2 + C \sum_i^N \xi_i$$

subject to

$$y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1 \dots N$$

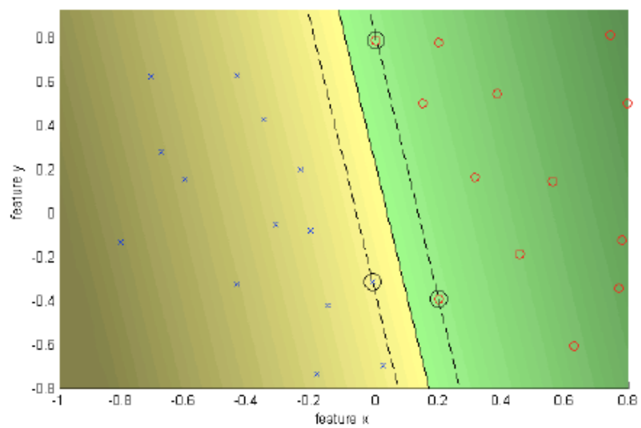
- Every constraint can be satisfied if ξ_i is sufficiently large
- C is a regularization parameter:
  - small C allows constraints to be easily ignored $\rightarrow$ large margin
  - large C makes constraints hard to ignore $\rightarrow$ narrow margin
  - $C = \infty$ enforces all constraints: hard margin
- This is still a quadratic optimization problem and there is a unique minimum. Note, there is only one parameter, C.

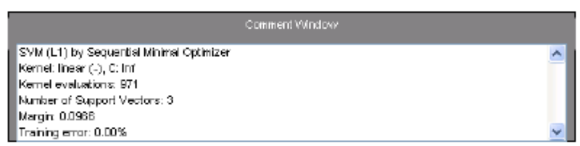
Example



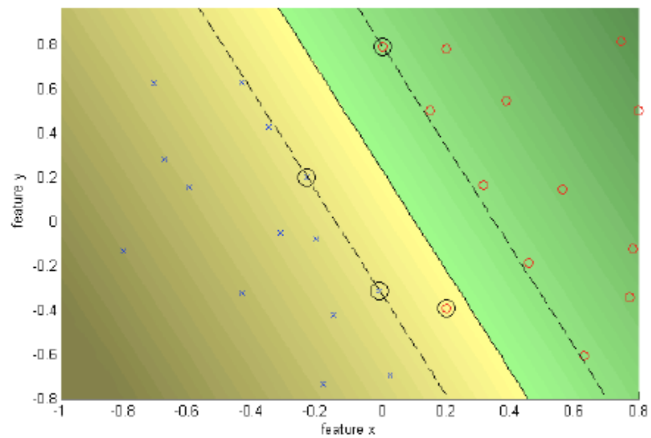
- data is linearly separable
- but only with a narrow margin

C = Infinity   hard margin





C = 10   soft margin



Comment Window

SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: 10.0000
Kernel evaluations: 2845
Number of Support Vectors: 4
Margin: 0.2285
Training error: 3.70%

Application: Pedestrian detection in Computer Vision

Objective: detect (localize) standing humans in an image

- cf face detection with a sliding window classifier

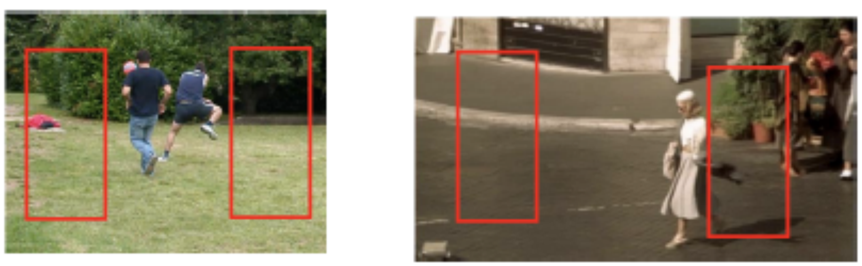
- reduces object detection to binary classification

- does an image window contain a person or not?

Training data and features

- Positive data – 1208 positive window examples

- Negative data – 1218 negative window examples (initially)



Algorithm

Training (Learning)

- Represent each example window by a HOG feature vector

$$\mathbf{x}_i \in \mathbb{R}^d, \text{ with } d = 1024$$

- Train a SVM classifier

Testing (Detection)

- Sliding window classifier

$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$

Using Kernel in SVM: motivation

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}.$$

Eq9

Before moving on, let's also take a more careful look at Equation (9), which gives the optimal value of w in terms of (the optimal value of) α. Suppose we've fit our model's parameters to a training set, and now wish to make a prediction at a new point input x. We would then calculate $w^T x + b$, and predict $y = 1$ if and only if this quantity is bigger than zero. But using (9), this quantity can also be written:

$$\begin{aligned} w^T x + b &= \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \right)^T x + b && (12) \\ &= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b. && (13) \end{aligned}$$

Using Kernel in SVM: motivation

Rather than applying SVMs using the original input attributes x, we may instead want to learn using some features $\phi(x)$. To do so, we simply need to go over our previous algorithm, and replace x everywhere in it with $\phi(x)$.

Since the algorithm can be written entirely in terms of the inner products $\langle x, z \rangle$, this means that we would replace all those inner products with $\langle \phi(x), \phi(z) \rangle$. Specifically, given a feature mapping ϕ, we define the corresponding **Kernel** to be

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

How to intuitively explain what a kernel is?

Kernel is a way of computing the dot product of two vectors $\mathbf{x}$ and $\mathbf{y}$ in some (possibly very high dimensional) feature space, which is why kernel functions are sometimes called "generalized dot product".

Suppose we have a mapping $\varphi : \mathbb{R}^n \to \mathbb{R}^m$ that brings our vectors in $\mathbb{R}^n$ to some feature space $\mathbb{R}^m$. Then the dot product of $\mathbf{x}$ and $\mathbf{y}$ in this space is $\varphi(\mathbf{x})^T\varphi(\mathbf{y})$. A kernel is a function k that corresponds to this dot product, i.e. $k(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x})^T\varphi(\mathbf{y})$.

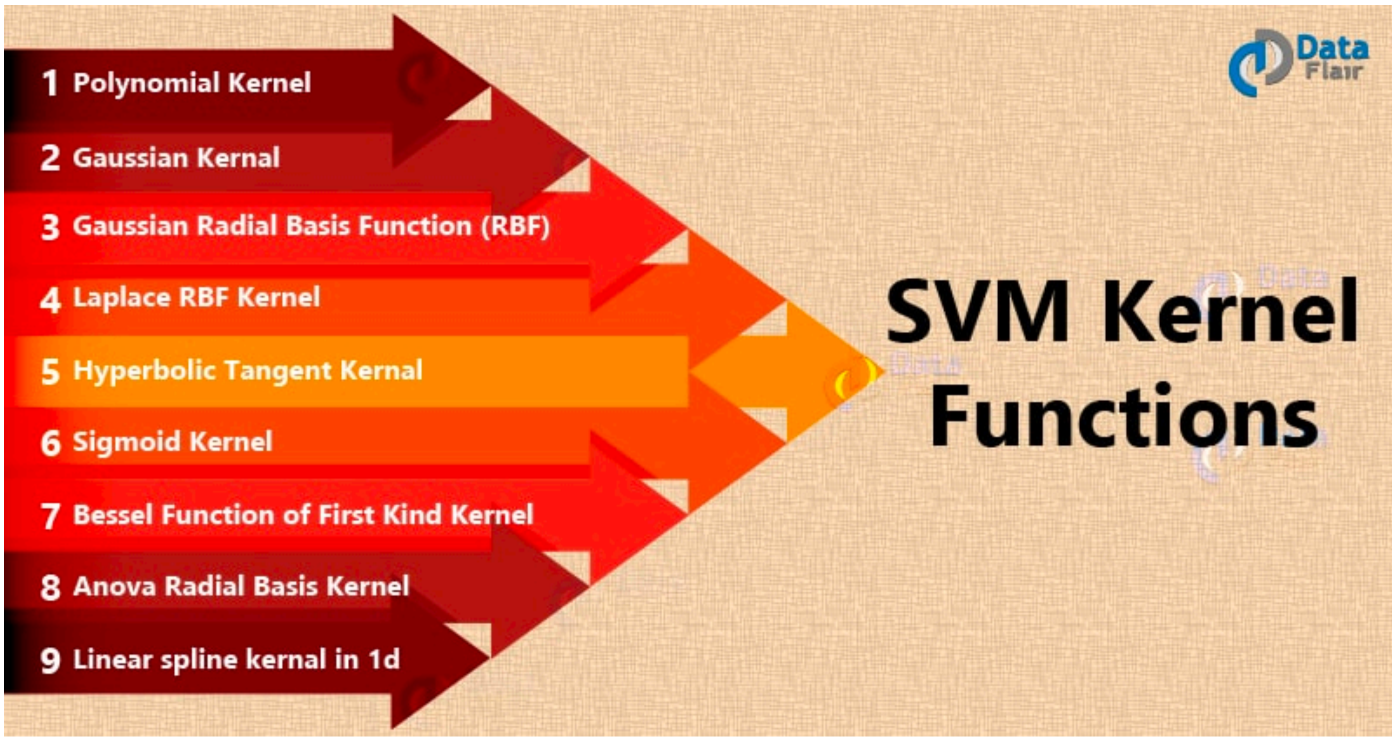
Why is this useful? Kernels give a way to compute dot products in some feature space without even knowing what this space is and what is φ.

For example, consider a simple polynomial kernel $k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T\mathbf{y})^2$ with $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$. This doesn't seem to correspond to any mapping function φ, it's just a function that returns a real number. Assuming that $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$, let's expand this expression:

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T\mathbf{y})^2 &= (1 + x_1\, y_1 + x_2\, y_2)^2 = \\
&= 1 + x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2
\end{aligned}
$$

Note that this is nothing else but a dot product between two vectors $(1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$ and $(1, y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1y_2)$, and $\varphi(\mathbf{x}) = \varphi(x_1, x_2) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$. So the kernel $k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T\mathbf{y})^2 = \varphi(\mathbf{x})^T\varphi(\mathbf{y})$ computes a dot product in 6-dimensional space without explicitly visiting this space.

SVM Kernel Functions



Examples of SVM Kernels

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma\|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

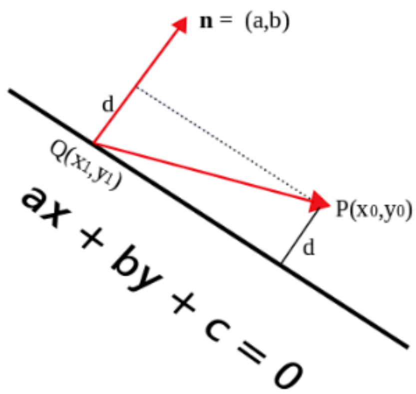
$$k(x, y) = \tanh(\alpha x^T y + c)$$

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma\|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

$$k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa \mathbf{x_i} \cdot \mathbf{x_j} + c)$$

$$k(x, y) = \exp\left(-\frac{||x - y||}{\sigma}\right)$$

Supplemental note: Distance from a point to a line



Let *P* be the point with coordinates (x_0, y_0) and let the given line have equation $ax + by + c = 0$. Also, let $Q = (x_1, y_1)$ be any point on this line and **n** the vector (a, b) starting at point *Q*. The vector **n** is perpendicular to the line, and the distance *d* from point *P* to the line is equal to the length of the orthogonal projection of $\overrightarrow{QP}$ on **n**. The length of this projection is given by:

$$d = \frac{|\overrightarrow{QP} \cdot \mathbf{n}|}{\|\mathbf{n}\|}.$$

Now,

$\overrightarrow{QP} = (x_0 - x_1, y_0 - y_1)$, so $\overrightarrow{QP} \cdot \mathbf{n} = a(x_0 - x_1) + b(y_0 - y_1)$ and $\|\mathbf{n}\| = \sqrt{a^2 + b^2}$,

thus

$$d = \frac{|a(x_0 - x_1) + b(y_0 - y_1)|}{\sqrt{a^2 + b^2}}.$$

Since *Q* is a point on the line, $c = -ax_1 - by_1$, and so,[8]

$$d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}.$$