Classification

Francesco Locatello
based on slides by Alina Dubatovka, Benjamin Gallusser
and Milica Petrovic

Nov 11 – 15, 2019

Classification – The setting

Goal: Assign a d-dimensional input vector $\mathbf{x}$ to one of K classes $y_k, k = 1, \ldots, K$

- divide the input space $\mathbf{X}$ into decision regions
- the decision boundaries (or surfaces) can be of any shape
- However, for mathematical simplicity, we would like to use decision boundaries of the form $\mathbf{w}^T\mathbf{x} = 0$

Classification – The setting

Generalized linear functions

In linear regression, the model for y was a linear function of input x and parameter w (weights). However, in classification, we need discrete numbers or probabilities as output. Idea: Apply some non-linear function f to $\mathbf{w}^T\mathbf{x}$: $\quad y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x})$.
Examples:

- take the sign: $y(\mathbf{x}) = \operatorname{sgn}(\mathbf{w}^T\mathbf{x})$
- sigmoidal functions like $y(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}} = \sigma(\mathbf{w}^T\mathbf{x})$

Important note: Even though the function $y(\mathbf{x})$ is non-linear in parameter $\mathbf{w}$, the decision surface is still linear as it is based only on the dot product.

Classification approaches

- **Discriminative**:
  - **probabilistic**: model class posterior $\mathbb{P}[Y|\mathbf{X}]$ and decide based on Bayes decision criteria (e.g. Logistic regression)
  - **non-probabilistic**: construct a discriminant function that directly assigns input $\mathbf{x}$ to a class k, without estimating underlying probability distributions (e.g. Perceptron, Fisher's LDA)
- **Generative probabilistic**: Model both class prior probabilities $\mathbb{P}[Y]$ and class-conditional probability densities $\mathbb{P}[\mathbf{X}|Y]$ to get the posterior

Linear Classifiers

Plain vanilla classifier: splitting a dataset into two classes (a *dichotomy*) using a *linear* separating hyperplane

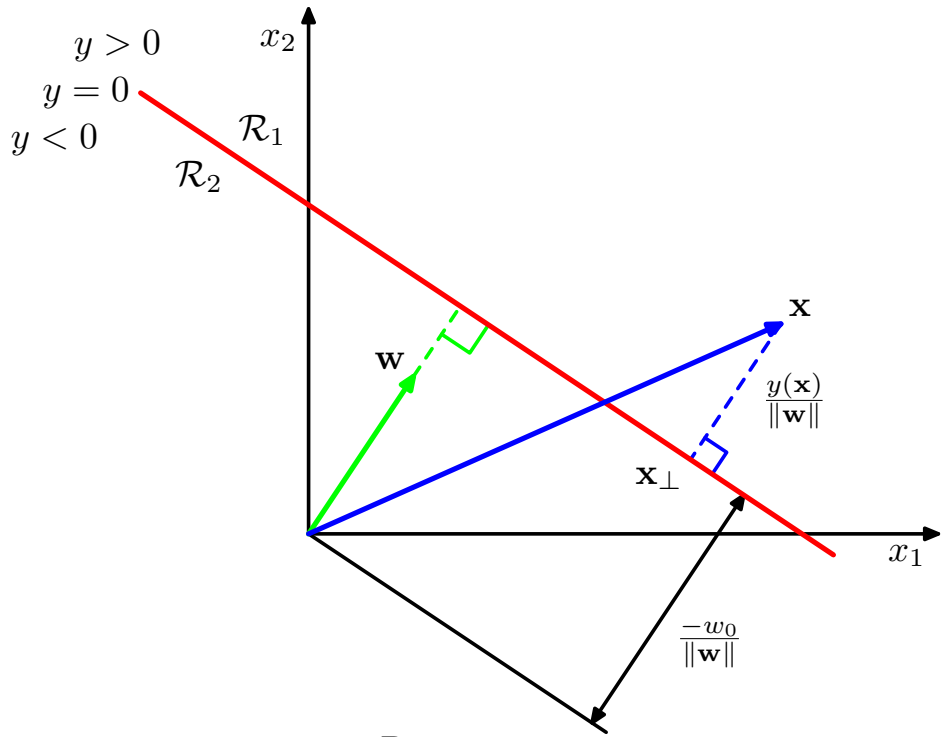
$$f(\mathbf{x}) = w_0 + \sum_{d=1}^{D} w_d x_d = w_0 + \mathbf{w}^T \mathbf{x} \tag{1}$$

- Labelled data point is pair $(\mathbf{x}_{(n)}, y_{(n)})$, with $y \in \{-1, +1\}$
- $\hat{y}_{(n)} = \mathrm{sgn}(f(\mathbf{x}_{(n)}))$ is classifier output

Extensions to $k > 2$ classes and non-linear decision surfaces often use this as the starting point.

Separating Hyperplane

Affine hyperplane geometry:



$$f(\mathbf{x}) = w_0 + \sum_{d=1}^{D} w_d x_d = w_0 + \mathbf{w}^T \mathbf{x}.$$

Homogeneous Coordinates

Two tricks to simplify notation:

1. Subsume w_0: augment feature vector $\mathbf{x} \in \mathbb{R}^D$ into $\tilde{\mathbf{x}} := (\mathbf{x}, 1) \in \mathbb{R}^{D+1}$

$$w_0 + \sum_{d=1}^{D} w_d x_d = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}$$

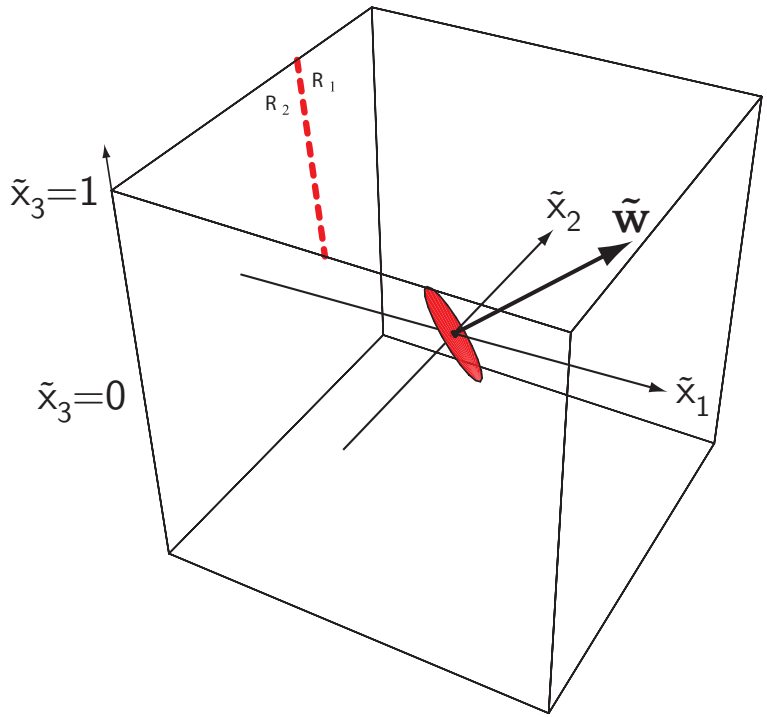
   with $\tilde{\mathbf{w}} = (\mathbf{w}, w_0)$.

2. Replace two sided test with one inequality: $\mathbf{x}_{(n)}$ correctly classified if

$$y_{(n)} f(\mathbf{x}_{(n)}) > 0.$$

Geometry of Homogeneous Coordinates

Augmented feature space:



- Data point $(x_1, x_2, 1)$ and weight vector $\tilde{\mathbf{w}}$ live in $\mathbb{R}^3$
- Shifting decision boundary in $\mathbb{R}^2$ means rotating $\tilde{\mathbf{w}}$ around origin

Points and (Hyper)-planes

End result:

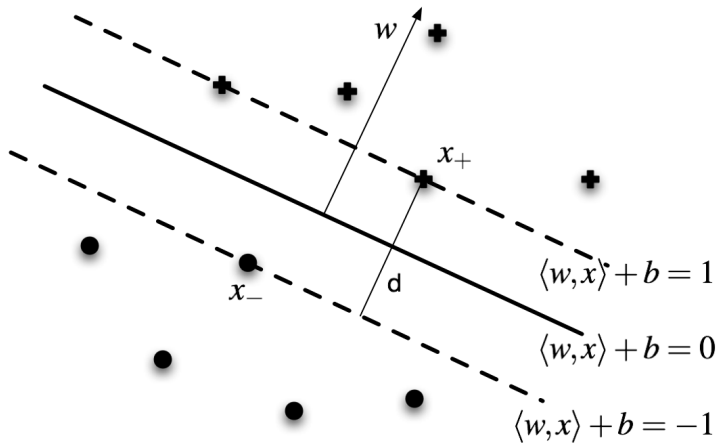
- Data point (x_1, x_2) is described by $\mathbf{x} = (x_1, x_2, 1)$ in homogeneous coordinates.
- A plane $ax_1 + bx_2 + c$ is described by a vector $\mathbf{w} = (a, b, c)$.
- Point $(x_1, x_2, 1)$ is on plane a (a, b, c) when $\mathbf{x}^T\mathbf{w} = 0$.
  If not on plane, side depends on the sign of $\mathbf{x}^T\mathbf{w}$.
- Our goal is:
  given a number of points $\{\mathbf{x_i}\}_{i=1}^n$ and a class corresponding to each point $\{y_i\}_{i=1}^n$ $(y_i \in \{-1, +1\})$, find a good separating plane.

  Two solutions considered: Perceptrons, SVMs

SVM

Why Maximize The Margin?

- Intuitively, it feels the safest.
- For a small error in the separating hyperplane, we do not suffer too many mistakes.
- Empirically, it works well.
- VC theory indicates that it is the right thing to do. I.e. good generalization properties.
- There is one global maximum, i.e. the problem is convex.



Deriving The Maximum Margin

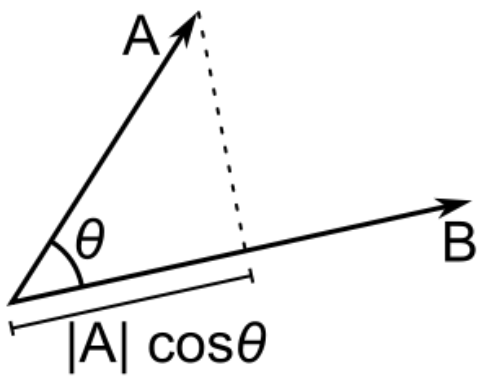
Recall:

- For two vectors $\mathbf{A}$ and $\mathbf{B}$, their inner product $\langle \mathbf{A}, \mathbf{B} \rangle$ or $\mathbf{A}^T\mathbf{B}$ is given by:

$$\mathbf{A}^T\mathbf{B} := \|\mathbf{A}\|\|\mathbf{B}\| \cos(\angle \mathbf{AB}).$$

- If $\mathbf{A}$ and $\mathbf{B}$ are two vectors, the projection ($\mathbf{C}$) of $\mathbf{A}$ on $\mathbf{B}$ is the vector that has the same slope as $\mathbf{B}$ with the length:

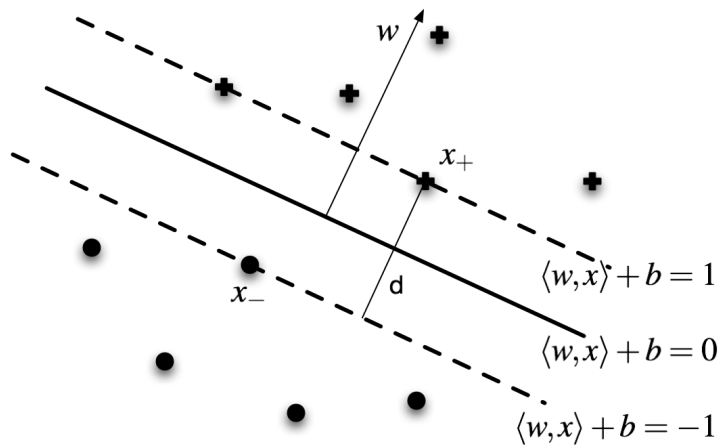
$$\|\mathbf{C}\| = \|\mathbf{A}\| \cos\theta$$



Deriving The Maximum Margin

Consider the figure on the right,

1. express the perpendicular distance between the two dotted lines d in terms of the vectors $\mathbf{x}_+$ and $\mathbf{x}_-$.



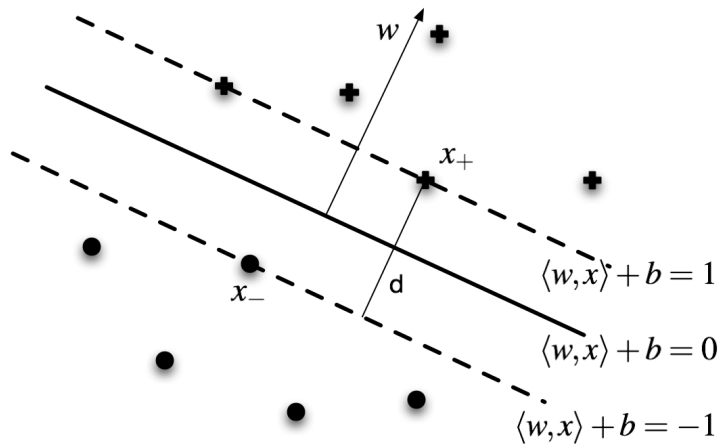
1. Let $\mathbf{x}$ be the vector $\mathbf{x} = \mathbf{x}_+ - \mathbf{x}_-$, then the margin d can be expressed as

$$d = \|\mathbf{x}\| \cos\theta$$

where θ is the angle between $\mathbf{x}$ and the vector perpendicular to the hyperplane.

Deriving The Maximum Margin

2. Express d in terms of an inner product using the vector $\mathbf{w}$ which is the normal vector to the hyperplane.



2. d can be expressed as

$$\|\mathbf{x}\| \cos\theta = \frac{\|\mathbf{x}\|\|\mathbf{w}\| \cos\theta}{\|\mathbf{w}\|} = \frac{\mathbf{w}^T\mathbf{x}}{\|\mathbf{w}\|} = \frac{\mathbf{w}^T\mathbf{x}_+ - \mathbf{x}_-}{\|\mathbf{w}\|}$$

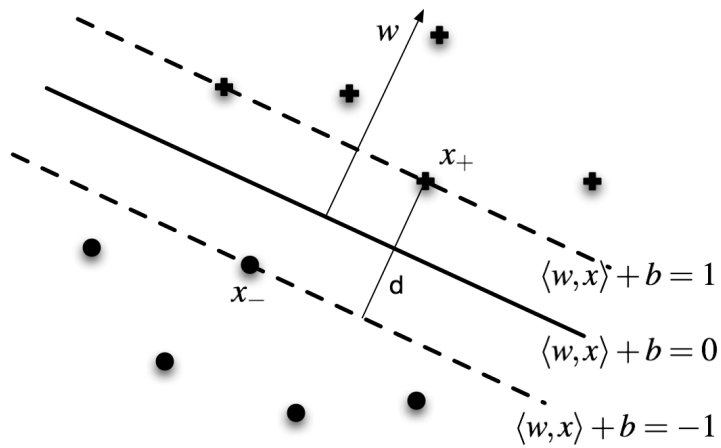
Deriving The Maximum Margin

3. Using the equations of the dotted lines, show that $d = \frac{2}{\|\mathbf{w}\|}$.

Conclude that

$$\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|}$$

maximizes the margin.



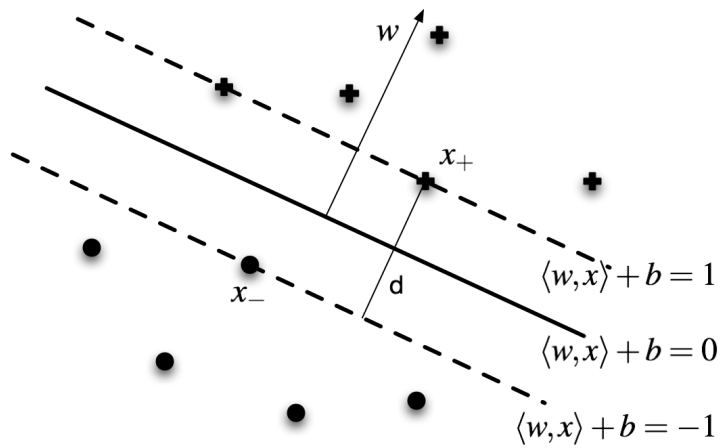
3. Substituting the lines equations leads to

$$d = 1 + b - (-1 + b) = 2$$

Deriving The Maximum Margin

4. What is the equation of the half space on the top right of the dotted line $\mathbf{w}^T\mathbf{x} + b = 1$?

5. What is the equation of the half space on the bottom left of the dotted line $\mathbf{w}^T\mathbf{x} + b = -1$?



4. The "upper" halfspace is described by: $\mathbf{w}^T\mathbf{x} + b > 1$.

5. The "lower" halfspace is described by: $\mathbf{w}^T\mathbf{x} + b < -1$.

Put all the above derivations together to obtain the hard margin support vector machine.

Hard Margin SVM

The objective of the hard margin support vector machine becomes:

$$\begin{array}{ll} \text{minimize} & \|\mathbf{w}\| \\ \text{subject to} & y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 \quad \text{for } i = 1, \ldots, n. \end{array}$$

This is hard to solve directly!

Hard Margin SVM Lagrangian

Primal form:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\mathbf{w}^T\mathbf{w} \\ &\text{subject to} && y_i(\mathbf{w}^T\mathbf{x_i} + w_0) \geq 1 \quad \text{for } i = 1, \ldots, m. \end{aligned}$$

* For convenient of derivation we use $\frac{1}{2}\mathbf{w}^T\mathbf{w}$

Lagrangian:

$$L(\mathbf{w}, w_0, \alpha) = \tfrac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{m} \alpha_i [y_i(\mathbf{w}^T\mathbf{x_i} + w_0) - 1]$$

Finding stationary points (Deriving with respect to the primal variables):

1. $\frac{\partial L(\mathbf{w}, w_0, \alpha)}{\partial \mathbf{w}} \rightarrow$ What is $\frac{\partial \mathbf{w}^T\mathbf{w}}{\partial \mathbf{w}}$??
2. $\frac{\partial L(\mathbf{w}, w_0, \alpha)}{\partial w_0}$

Vector Calculus: Derivatives

The partial derivative of a function $g : R^n \rightarrow R$ of a vector $w \in R^n$, with respect to the vector itself is given by:

$$\frac{\partial g(\mathbf{w})}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial g(\mathbf{w})}{\partial w_1} \\ \frac{\partial g(\mathbf{w})}{\partial w_2} \\ \vdots \\ \frac{\partial g(\mathbf{w})}{\partial w_n} \end{bmatrix}.$$

$\mathbf{w}^T\mathbf{w} = w_1^2 + w_2^2 + \ldots + w_n^2$. The partial derivative of $\mathbf{w}^T\mathbf{w}$ is given by

$$\frac{\partial \mathbf{w}^T\mathbf{w}}{\partial \mathbf{w}} = 2\mathbf{w}$$

Hard Margin SVM Dual

Finding extreme points:

$$\frac{\partial L(\mathbf{w}, w_0, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x_i} = 0 \rightarrow w = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x_i}$$

$$\frac{\partial L(\mathbf{w}, w_0, \alpha)}{\partial w_0} = -\sum_{i=1}^{m} \alpha_i y_i = 0$$

Substituting $w = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x_i}$ and $\sum_{i=1}^{m} \alpha_i y_i = 0$ into $L(\mathbf{w}, w_0, \alpha)$:

$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^T \mathbf{x_j} - \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^T \mathbf{x_j} + \sum_{i=1}^{m} \alpha_i$$

$$= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^T \mathbf{x_j}$$

Hard Margin SVM Dual

The Hard Margin SVM Dual becomes:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^T \mathbf{x_j}$$

subject to:

$$\forall i . \alpha_i \geq 0$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

How is this better than before?

Hard Margin SVM Dual

How is this better than before?

- It is easier to solve, e.g. libSVM uses it
- We can recover the optimal $\mathbf{w}$ via

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x_i}$$

- The classification reduce to

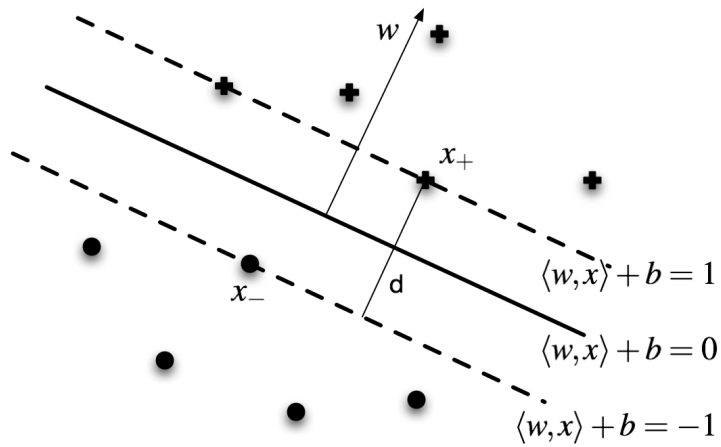
$$\mathbf{w}^T \mathbf{x} + w_0 = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x_i}^T \mathbf{x} + w_0$$

- Intuitively we expect most of the α_i to be zero.

Support Vectors

* The separating hyperplane is determined only by the points which are closest to it (the support vectors).
The points which do not lie on the boundary do not contribute to the classification

In theory, you may have millions of data points, but only three support vectors.



Soft Margin SVM Motivation

- Non-separable datasets - in real life most datasets are in fact non separable

  Non linearly separable Use kernel trick
  Non separable introduce slack variables

- Sensitivity to outliers

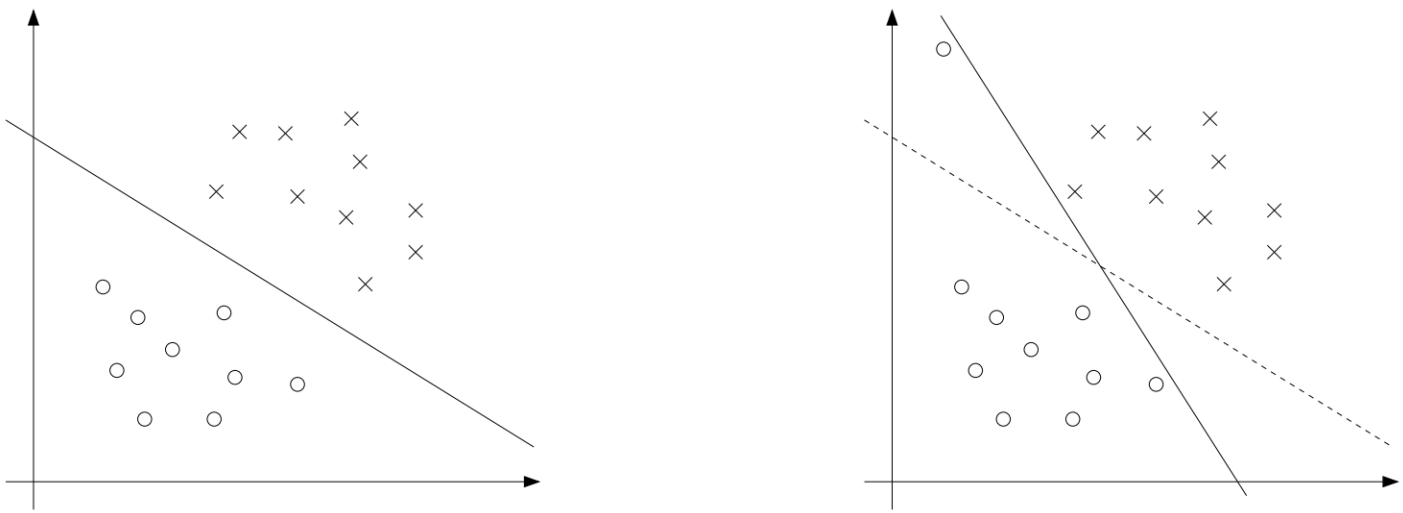


Figure from: http://cs229.stanford.edu/notes/cs229-notes3.pdf

Soft Margin SVM

Key Ideas:

1. Introduction of slack variables
   (see optimization techniques for more details on slack variables).
2. Penalize the misclassified examples

Hard Margin SVM: The margin term is defined by the hard constraints:

$$\forall i \; y_i \mathbf{w}^T \mathbf{x}_i \geq 1,$$

Soft margin SVM: soften the constraints:

$$\forall i \; y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i \; , \qquad \xi_i \geq 0$$

This new constraint permits a functional margin that is less than 1.

Note: Slack variables apply only to training data. Classification of test points depends only on which side of the hyperplane they are on.

Soft Margin SVM

Penalty Cost: we still prefer a hyperplane which correctly classifies the data.

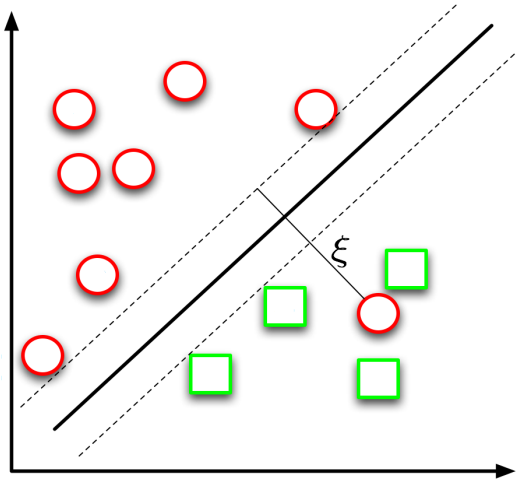
Introduce $C\xi_i$ penalty for each data point i which falls:

- Within the margin ($0 < \xi_i \leq 1$)
- On the wrong side of the hyperplane ($\xi_i > 1$).

$\rightarrow$ The penalty is proportional to the amount by which the example is misclassified.

Target: Minimize the sum of the total penalties ξ_i over all i (This is an upper bound for the training classification error).

Soft Margin SVM



Soft margin SVM becomes

$$\min_{\mathbf{w},\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i$$

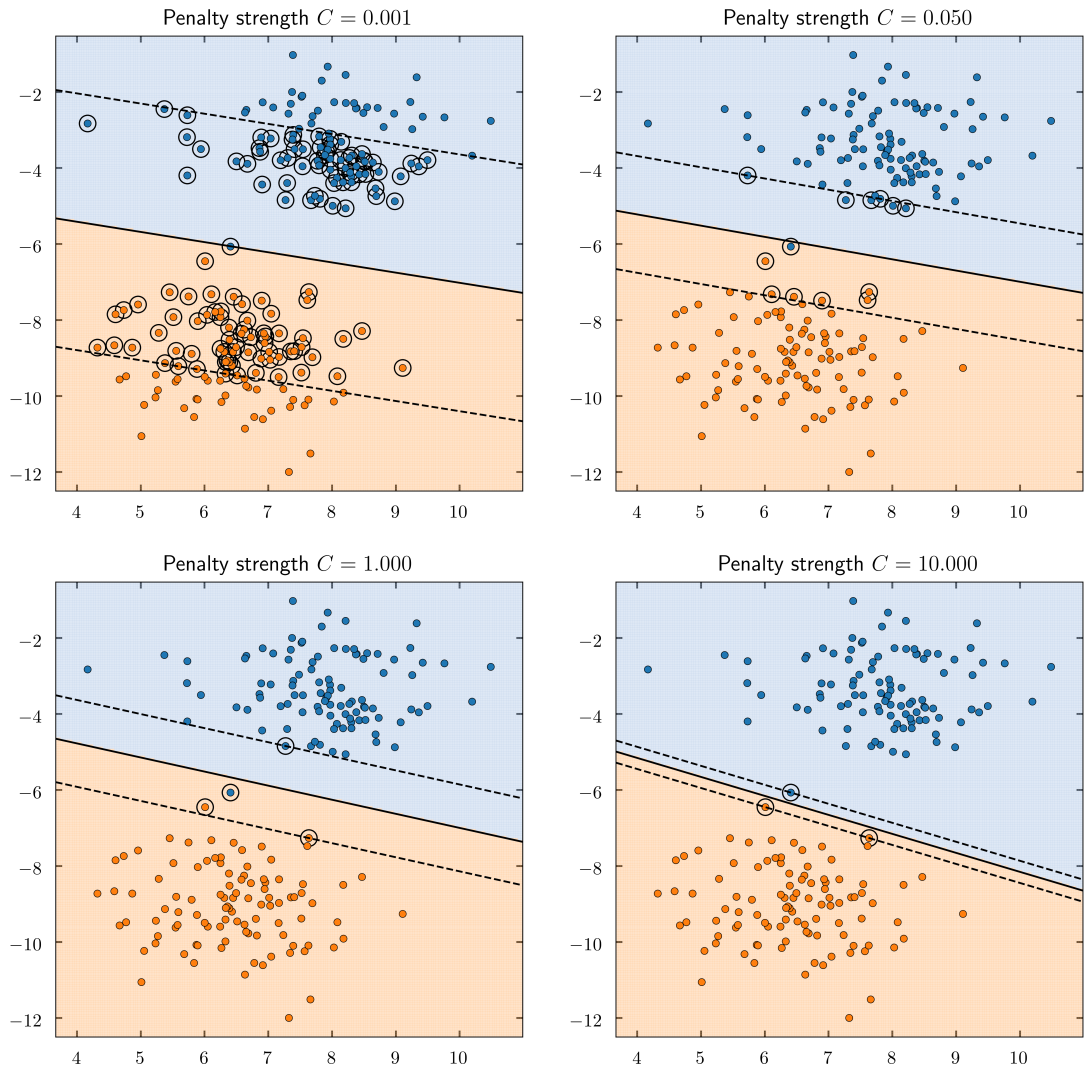
Subject to:

$$y_i\mathbf{w}^T\mathbf{x}_i \geqslant 1-\xi_i$$
$$\xi_i \geqslant 0 \text{ for all } i = 1,\ldots,m.$$

- There are many ways to combine two cost terms into a cost function: Every expression that becomes larger as either of the two terms increases is valid.
- Soft margin SVM: Choose the simple way, add the terms.

  cost of solution = margin costs + C * slack costs (in L_1 norm)
- C is a constant which controls the amount of constraint violations vs. margin maximization. The value of C is found using cross validation.

Penalty Strength



Plot from: Dariusz Kajtoch

SVM Summary

- By definition, SVM is the maximum margin classifier defined in terms of the support vector approach.
- Real-world SVM implementations usually combine three techniques:

  1. Maximum margin classifier (this is where convex optimization comes in)
  2. Soft margin technique (slack variables)
  3. Kernel trick

- Three very simple reason why SVMs are so popular:

  1. Of proven merit.
  2. Lots of experience, literature etc exists.
  3. Several easy-to-use, freely accessible, well-tested implementations are available (libsvm, svmlite, shogun etc.)