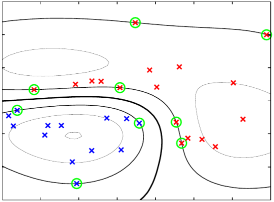
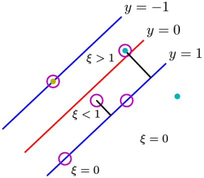
Machine Learning

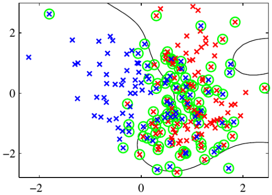
Support Vector Machines

Marcello Restelli

May 13, 2017







Support Vector Machines (SVMs)

- They have a long history, but they were invented in the present form in the **late 90's**
- One of the **best** methods for **classification**
- It is one of the most **mathematical** and **difficult** topic in machine learning
    - Learning theory
    - Kernel theory
    - Constrained optimization
- As a consequence people use SVMs as **black boxes**
- We will not see all the details, but give a **basic understanding** of how SVMs works and what are the **important parameters**

What Is a Support Vector Machine?

1. A **subset** of training examples **x** (the support vectors)
2. A vector of weights for them **a**
3. A similarity function $K(x, x')$ (the **kernel**)

Class prediction for a new example x_q ($t_i \in \{-1, 1\}$):

$$f(x_q) = \text{sign}\left(\sum_{m \in \mathcal{S}} \alpha_m t_m k(x_q, x_m) + b\right),$$

where $\mathcal{S}$ is the set of **indices** of the **support vectors**

- It is a very **smart** way of doing **instance based learning**
- They are usually presented as a **generalization** of the **perceptron**
- What's the **relation** between perceptrons and instance-based learning?

The Perceptron Revisited

- What **similarity function** makes the weighted kNN work like the perceptron?
  - The **dot product**:

$$f(\mathbf{x}_q) = \text{sign}\left[\sum_{j=1}^{M} w_j \phi_j(\mathbf{x}_q)\right]$$

  - but

$$w_j = \sum_{n=1}^{N} \alpha_n t_n \phi_j(\mathbf{x}_n)$$

  - so

$$f(\mathbf{x}_q) = \text{sign}\left[\sum_{j=1}^{M}\left(\sum_{n=1}^{N} \alpha_n t_n \phi_j(\mathbf{x}_n)\right)\phi_j(\mathbf{x}_q)\right] = \text{sign}\left[\sum_{n=1}^{N} \alpha_n t_n (\boldsymbol{\phi}(\mathbf{x}_q) \cdot \boldsymbol{\phi}(\mathbf{x}_n))\right]$$

- The sum over the **features** has been rewritten as a sum over the **samples**
- So, the **perceptron** can be seen as a special case of **instance-based learning**

Another View of SVMs

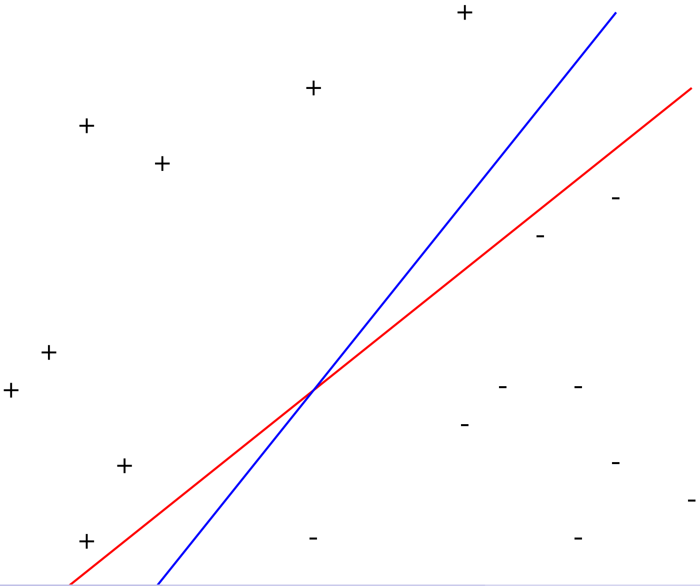
- Take the **perceptron**
- Replace the **dot product** with arbitrary **similarity function**
- Now you have a much more **powerful learner**
- Kernel matrix: $k(\mathbf{x}, \mathbf{x}')$
- If a symmetric matrix K is positive semi-definite (i.e., has non-negative eigenvalues), then $k(\mathbf{x}, \mathbf{x}')$ is still a dot product, but in a transformed space:
$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$
- Also guarantees **convex** weight optimization problem: **no local optima!!!**
- Very **general trick**

Learning SVMs

So how do we:

- Choose the kernel?
  - Black art
- Choose the examples?
  - Side effect of choosing weights
- Choose the weights?
  - **Maximize the margin**

Margin Example



Maximize the Margin

- We want to maximize the margin, that is **maximize the distance** of the **closest point** to the hyperplane

The Weight Optimization Problem

- **Margin** = $\min_n t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b)$
- The maximum margin solution is found by solving:

$$\mathbf{w}^* = arg\max_{\mathbf{w},b}\left(\frac{1}{\|\mathbf{w}\|_2}\min_n(t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b))\right)$$

- Direct solution is **complex**, we need to consider an **equivalent** problem **easier** to be solved
- **Fix margin**, minimize weights

$$\begin{array}{rl}\textbf{Minimize} & \frac{1}{2}\|\mathbf{w}\|_2^2 \\ \textbf{Subject to} & t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) \geq 1, \text{ for all } n\end{array}$$

Constrained Optimization Basics

- $$\begin{aligned} \textbf{Minimize} \quad & f(\mathbf{w}) \\ \textbf{Subject to} \quad & h_i(\mathbf{w}) = 0, \text{ for } i = 1, 2, \dots \end{aligned}$$
- If f and h_i are linear we have **linear programming**, but here we are interested in **quadratic programming**
- At solution $\mathbf{w}^*$, $\nabla f(\mathbf{w}^*)$ must lie in subspace spanned by $\{\nabla h_i(\mathbf{w}^*) : i = 1, 2, \dots\}$
- **Lagrangian function:**

$$L(\mathbf{w}, \boldsymbol{\lambda}) = f(\mathbf{w}) + \sum_i \lambda_i h_i(\mathbf{w})$$

- The λ_is are the **Lagrange multipliers**
- Solve $\nabla L(\mathbf{w}^*, \boldsymbol{\lambda}^*) = 0$

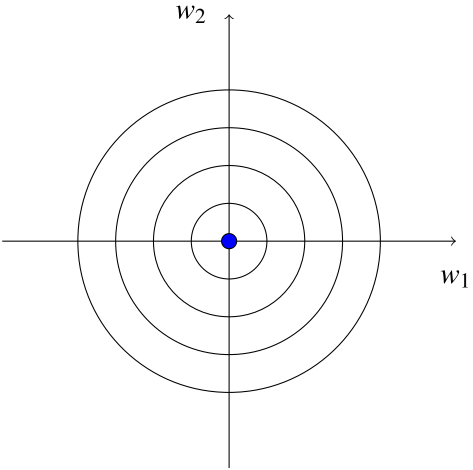
Example

Minimize $\frac{1}{2}(w_1^2 + w_2^2)$

Subject to $w_1 + w_2 = 1$

$$L = \frac{1}{2}(w_1^2 + w_2^2) + \lambda(w_1 + w_2 - 1)$$

$$\nabla L = 0 \rightarrow \begin{cases} w_1 + \lambda = 0 \\ w_2 + \lambda = 0 \\ w_1 + w_2 - 1 = 0 \end{cases}$$



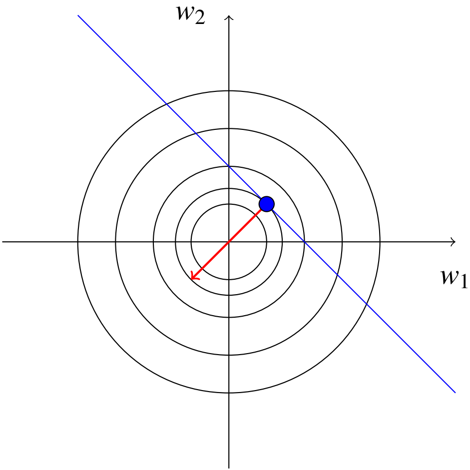
Example

$$\textbf{Minimize} \quad \frac{1}{2}(w_1^2 + w_2^2)$$
$$\textbf{Subject to} \quad w_1 + w_2 = 1$$

$$L = \frac{1}{2}(w_1^2 + w_2^2) + \lambda(w_1 + w_2 - 1)$$

$$\nabla L = 0 \rightarrow \begin{cases} w_1 + \lambda = 0 \\ w_2 + \lambda = 0 \\ w_1 + w_2 - 1 = 0 \end{cases}$$

$$\text{Solution:} \begin{cases} w_1 = w_2 = \frac{1}{2} \\ \lambda = -\frac{1}{2} \end{cases}$$



Example

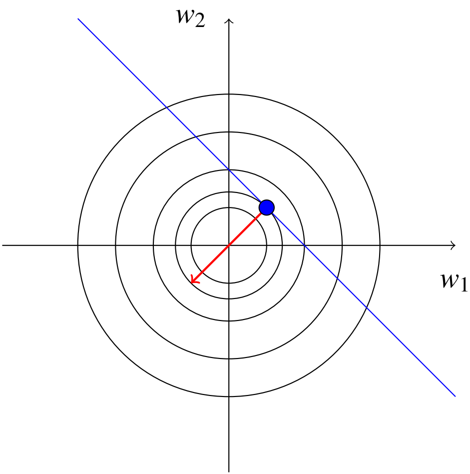
$$\textbf{Minimize} \quad \frac{1}{2}(w_1^2 + w_2^2)$$

$$\textbf{Subject to} \quad w_1 + w_2 = 1$$

$$L = \frac{1}{2}(w_1^2 + w_2^2) + \lambda(w_1 + w_2 - 1)$$

$$\nabla L = 0 \rightarrow \begin{cases} w_1 + \lambda = 0 \\ w_2 + \lambda = 0 \\ w_1 + w_2 - 1 = 0 \end{cases}$$

$$\text{Dual: } \begin{cases} w_1 = -\lambda \\ w_2 = -\lambda \\ L = -\lambda^2 - \lambda \end{cases} \quad \rightarrow \lambda = -\frac{1}{2}$$



Inequality Constraints

- $$\begin{array}{lll} \textbf{Minimize} & f(\mathbf{w}) & \\ \textbf{Subject to} & g_i(\mathbf{w}) \leq 0, & \text{for } i = 1, 2, \dots \\ & h_i(\mathbf{w}) = 0, & \text{for } i = 1, 2, \dots \end{array}$$
- Lagrange multipliers for inequalities: α_i
- **KKT Conditions** (necessary conditions):

$$\begin{aligned} \nabla L(\mathbf{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\lambda}^*) &= 0 \\ h_i(\mathbf{w}^*) &= 0 \\ g_i(\mathbf{w}^*) &\leq 0 \\ \alpha_i^* &\geq 0 \\ \alpha_i^* g_i(\mathbf{w}^*) &= 0 \end{aligned}$$

- **Complementarity**: Either a constraint is active ($g_i(\mathbf{w}^*) = 0$) or its multiplier is zero ($\alpha_i^* = 0$)

Primal and Dual Problems

- Problem over $\mathbf{w}$ (weights over the features) is the **primal**
- Solve equations for $\mathbf{w}$ and **substitute**
- Resulting problem over $\boldsymbol{\lambda}$ is the **dual**
- **If it's easier**, solve the dual instead of primal
- In SVMs
  - Primal problem is over **feature weights**
  - Dual problem is over **instance weights**
- The solution over the dual problem will have a lot of **zero weights**

Dual Representation

- Let's consider the **Lagrangian function**

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_{n=1}^{N} \alpha_n (t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) - 1)$$

- Putting the gradient w.r.t. $\mathbf{w}$ and b to zero we get

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n t_n \phi(\mathbf{x}_n), \qquad 0 = \sum_{n=1}^{N} \alpha_n t_n$$

- We can **rewrite** the Lagrangian as follows

$$\textbf{Maximize} \quad \tilde{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

$$\textbf{Subject to} \quad \alpha_n \geq 0, \qquad \text{for } n = 1, \ldots, N$$

$$\sum_{n=1}^{N} \alpha_n t_n = 0$$

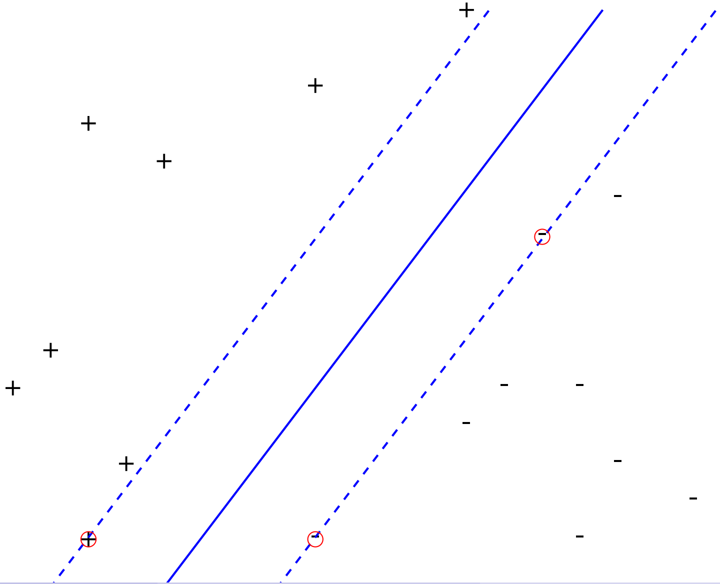
SVM prediction

- The classification of **new points** with the train model is:

$$y(\mathbf{x}) = \mathrm{sign}\left(\sum_{n=1}^{N} \alpha_n t_n k(\mathbf{x}, \mathbf{x}_n) + b\right)$$

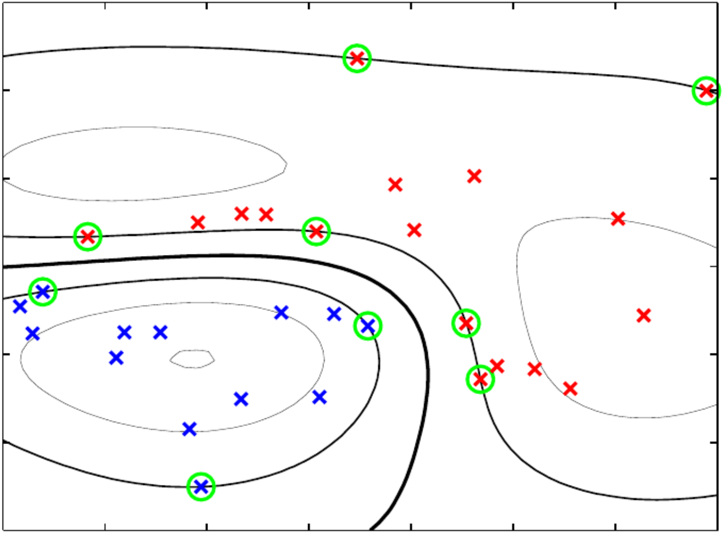
- where $b = \dfrac{1}{N_{\mathcal{S}}} \displaystyle\sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} \alpha_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right)$
- Notice that $N_{\mathcal{S}}$ (the number of support vectors) is usually **much smaller** than N

Maximize the Margin



Maximize the Margin

SVM using **Gaussian kernel function**



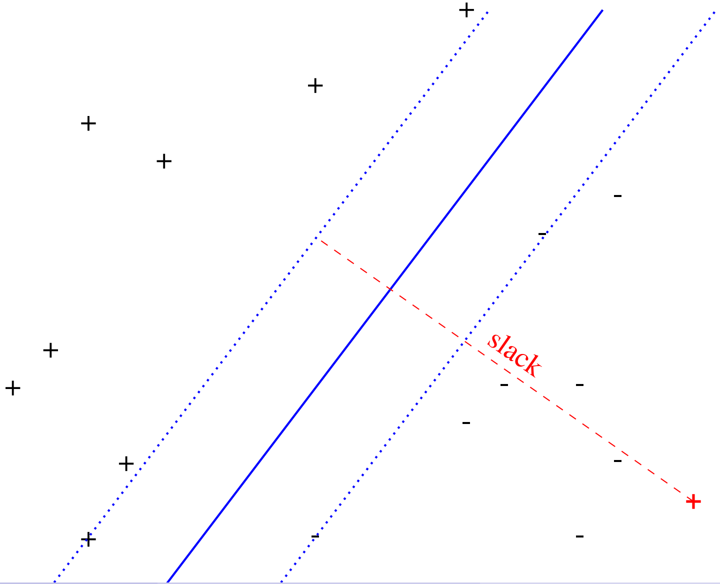
Curse of Dimensionality and SVMs

- What happens when the number of dimensions increases?
    - The number of support vectors **increases too**
- In high dimensional problems the percentage of support vectors can become **significant**
- **Scalability** becomes an issue
- This affects the **generalization guarantees**

Solution Techniques

- Use **generic** quadratic programming solver
- Millions of samples means millions of constraints!
- Use **specialized** optimization algorithm
- E.g., SMO (Sequential Minimal Optimization)
  - **Simplest** method: update one α_i at a time
  - But this **violates constraints**
  - Iterate until convergence:
    1. Find example x_i that violates KKT conditions
    2. Select second example x_j heuristically
    3. Jointly optimize α_i and α_j

Handling Noisy Data



Handling Noisy Data

- Introduce **slack variables** ξ_i
- We allow to **violate** the margin constraint, but we add a **penalty**
-

$$\begin{aligned} \textbf{Minimize} \quad & \|\mathbf{w}\|_2^2 + C \sum_i \xi_i \\ \textbf{Subject to} \quad & t_i(\mathbf{w}^T x_i + b) \geq 1 - \xi_i, && \text{for all } i \\ & \xi_i \geq 0, && \text{for all } i \end{aligned}$$

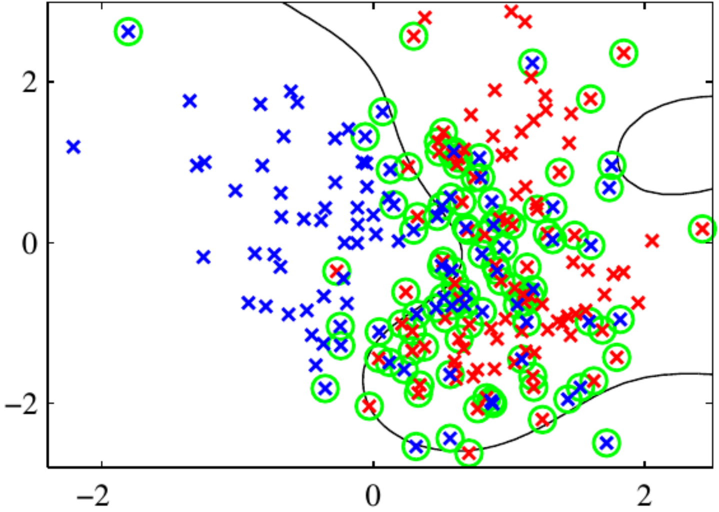
- C is a coefficient that allows to **tradeoff bias-variance**
- C is chosen by **cross validation**

Dual Representation

$$
\begin{aligned}
\textbf{Maximize} \quad & \tilde{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \\
\textbf{Subject to} \quad & 0 \leq \alpha_n \leq C, \qquad \text{for } n = 1, \dots, N \\
& \sum_{n=1}^{N} \alpha_n t_n = 0
\end{aligned}
$$

- **Support vectors** are points associated to $\alpha_n > 0$
- If $\alpha_n < C$ the point lies **on the margin**
- If $\alpha_n = C$ the point lies **inside the margin**, and it can be either **correctly classified** ($\xi_i \leq 1$) or **misclassified** ($\xi > 1$)

Example



Bounds

- **Margin bound**
  - Bound on VC dimension **decreases** with margin
  - The larger the margin, less capacity to overfit, less VC dimension
  - Margin bound is **quite loose**
- **Leave-one-out bound:**
  - $L_h \leq \dfrac{\mathbb{E}[\text{Number of support vectors}]}{N}$
  - It can be **easily computed**
  - We do **not** need to run SVM multiple times

Other SVM Uses

- SVMs are **not** used only for classification
- SVMs can be used for
  - Regression
  - Ranking
  - Feature selection
  - Clustering
  - Semi-supervised learning