Support Vector Machines

Machine Learning

Daniele Loiacono

POLITECNICO MILANO 1863

Outline and References

- Outline
  - Separable Problems [PRML 7.1]
  - Non-Separable Problems [PRML 7.1.1]
  - Training [PRML 7.1.1]
  - Multi-Class SVM [PRML 7.1.3]
  - SVM for regression [PRML 7.1.4]

- References
  - Pattern Recognition and Machine Learning, Bishop [PRML]

Sparse Kernel Machines

- A significant limitation to kernel methods is that we need to compute the kernel function for each samples in the training set, that is often computationally unfeasible
- To deal with this issue, **sparse kernel methods** find **sparse** solutions, that rely only on a **subset** of the training samples
- The most well known among these methods are:
  - Support Vector Machines
  - Relevance Vector Machines

We went from the parametric approach (OLS, Ridge regression, ..) to nonparametric approach (kernel methods). This because the parametric approach is not feasible when the number of features is too high (or ∞). The nonparametric approach is okay also with $M = \infty$. However

Sparse Kernel Machines

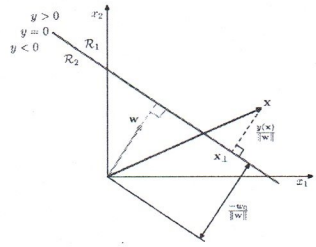
- A significant limitation to kernel methods is that we need to compute the kernel function for each sample in the training set, that is often computationally unfeasible
- To deal with this issue, **sparse kernel methods** find **sparse** solutions, that rely only on a **subset** of the training samples
- The most well known among these methods are:
  - **Support Vector Machines** ← our focus
  - Relevance Vector Machines

Separable Problems

Do you remember the perceptron?

linear classification approach that identify a linear boundary in the features space that allows to classify between +1 and -1

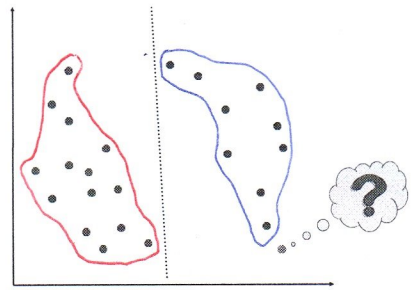
$$f(\mathbf{x}, \mathbf{w}) = \begin{cases} +1, & \text{if } \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + b \geq 0 \\ -1, & \text{otherwise} \end{cases}$$



- Properties
  - DS is $y(\cdot) = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + b = 0$
  - 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 $\frac{y(\mathbf{x})}{\|\mathbf{w}\|_2}$

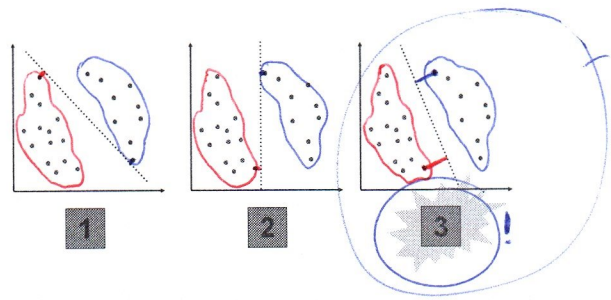
"distance"
(because is >0 in the part of +1 and it's <0 in the part of -1)

The perceptron in action



Are all the solution equivalent?

For the perceptron these solutions are equivalent:



This solution has a better margin. The closest points (in both cases) have the biggest distance from the classifier (biggest among the other examples). This makes the solution more robust (data can have noise).

Maximum Margin Classifier

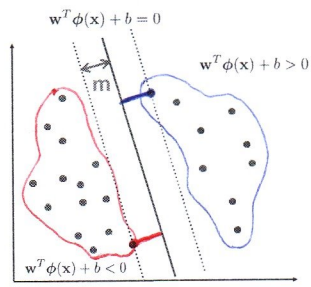
- The margin will be:

$$margin = \min_n \frac{t_n(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) + b)}{||\mathbf{w}||}$$ *

- Thus, the optimal **hyperplane** will be:

$$\arg\max_{\mathbf{w},b} \left\{ \min_n \left[\frac{t_n(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) + b)}{||\mathbf{w}||} \right] \right\}$$

- Unfortunately, solving this optimization problem would be very complex...



notice that the distance of $\underline{x}_n$ from the function is just $y(\underline{x}_n)/\|\underline{w}\|_2 = (\underline{w}^T\underline{\phi}(\underline{x}_n)+b)/\|\underline{w}\|_2$ however since it's not a proper distance (it can be <0) we multiply it by the target t_n ⟹ now it's a real distance (if $t_n = -1$ the dist is <0 and so the multipl. brings something positive)

Equivalent constrained optimization problem

- **Canonical** hyperplane
  - There are infinite equivalent solutions:

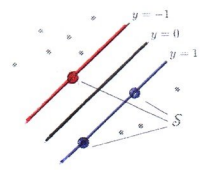
$$\kappa\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + \kappa b \quad \forall \kappa > 0$$

  - We will consider only solutions such that:

$$t_n(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) + b) = 1 \quad \forall \mathbf{x}_n \in \mathcal{S}$$

- Equivalent **quadratic programming** problem

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



The points that are the closest to the discriminant boundary are called SUPPORT VECTORS. They're the only ones that really matter in the process. (more importantly, in the prediction)

* using this, the margin is equal to: $1/\|\underline{w}\|_2$

and so, to maximize the margin we have to minimize $\|\underline{w}\|_2$

(however we don't want to compute it (solve it) directly because we don't want to compute $\underline{\phi}(\underline{x}_n)$ → because we want a KERNEL METHOD APPROACH)

Derivation of dual problem

See [PRML - Appendix E]

(Procedure for constrained optimization problems)

- We can solve the constrained optimization problem using **Lagrance multipliers**:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(t_i \left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b\right) - 1\right) \qquad (\alpha_i \geq 0)$$

- We need to maximize $\mathcal{L}$ with respect to α and minimize it with respect to $\mathbf{w}$ and b (check optimization)
- We can compute the gradient w.r.t. $\mathbf{w}$ and b and derive dual representation

$$\frac{\partial}{\partial \mathbf{w}}\mathcal{L} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i t_i \boldsymbol{\phi}(\mathbf{x}_i) \qquad (*)$$

$$\frac{\partial}{\partial b}\mathcal{L} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i t_i = 0$$

We use the Lagrange multipliers for a change of variables (kernel trick for support vector machines)

Dual Problem

- We can now rewrite the optimization problem as:

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

$$\text{Subject to} \quad \alpha_n \geq 0 \text{ and } \sum_{n=1}^{N} \alpha_n t_n = 0, \quad \text{for } n = 1, \ldots, N$$

We again have a quadratic programming problem, however this time we have the kernels (we don't have to compute $\underline{\phi}(\cdot)$!)

  - where the explicit feature mapping does not appear explicitly anymore

Discriminant Function and Support Vectors

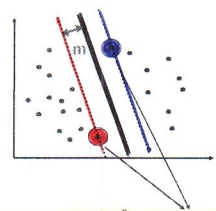
- The resulting discriminant function is:

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

  - Only samples on the margin does contribute (i.e., $\alpha_i > 0$),
  - These samples are the Support Vectors
  - The bias is computed as:

$$b = \frac{1}{|\mathcal{S}|}\sum_{\mathbf{x}_n \in \mathcal{S}} \left(t_n - \sum_{\mathbf{x}_m \in \mathcal{S}} \alpha_m t_m k(\mathbf{x}_n, \mathbf{x}_m)\right)$$

Support Vectors ($\mathcal{S}$)



$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(t_i \left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b\right) - 1\right)$$

0

$y(\underline{x}) = \underline{w}^T \underline{\phi}(\underline{x}) = \sum \alpha_i t_i k(\underline{x}, \underline{x}_n)$ it appears the kernel since we end up with $\underline{\phi}(\cdot)^T \underline{\phi}(\cdot)$

The prediction function depends entirely only on the **KERNEL** (not $\underline{\phi}(\cdot)$). However, didn't we do it to avoid the computations of all the kernels? Yes, but here we got the α's. For most of the points we'll have $\alpha_i = 0$! $\alpha_i \neq 0$ only for support vectors !!

We have to maximize it (z) w.r.t. $\underline{\alpha}$. This means that when $t_i(\underline{w}^T \underline{\phi}(\underline{x}_i) + b) - 1 > 0$ we have:

$z(\underline{\alpha}) =$ something $- \underline{\alpha} \cdot$ some.z

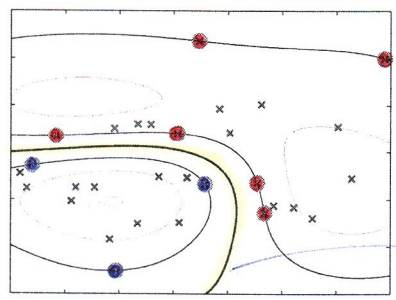
Here "some.z" (something-2) is > 0 (as already said) and so $\nabla z(\underline{\alpha})$ w.r.t. $\underline{\alpha}$ will be: $-$ something 2. This means that in this case, to optimize, $\underline{\alpha}$ must be equal to $\underline{0}$.

$\alpha_i > 0$ i support vector
$\alpha_i = 0$ i not

Example

Note that SVM is linear in the features space, not necessarily in the input space

- An example of SVM discriminant function using Gaussian kernel function

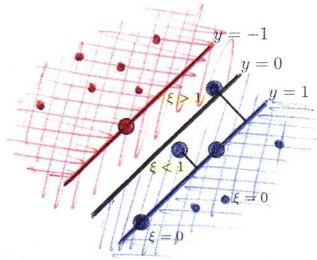


boundary (fully determined by the support vectors, the highlighted points)

Non-separable Problems

Non-Separable Problem

- So far, we assumed that samples are linearly separable in the feature space
- However, this is not always the case (e.g., noisy data)
- How to deal with such problems? We allow "error" (ξ_i) in classification:



There are two possible errors: one is very strong and is when a point is on the wrong side of the decision boundary ($\xi > 1$) and the other is when we have a violation of the margin ($\xi < 1$)

Soft-Margin optimization problem

$$\begin{aligned} \textbf{Minimize} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{n=1}^{N}\xi_n \\ \textbf{Subject to} \quad & t_n(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) + b) \geq 1 - \xi_n, \qquad \text{for all } n \\ & \xi_n \geq 0, \qquad \text{for all } n \end{aligned}$$

- ξ_n are called **slack variables** and represents **penalties** to **margin violation**
- C is a tradeoff parameter between error and margin
  - it allows to adjust the **bias-variance tradeoff**
  - **tuning** is required to find optimal value for C

The bigger is C and the less the violations will be allowed (the bigger C the more important becomes "$C\sum_{n=1}^{N}\xi_n$")

Dual Representation (Box Constraints Problem)

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

- As usual, **support vectors** are the samples for which $\alpha_n > 0$
  - If $\alpha_n < C \Rightarrow \xi_n = 0$, i.e., the sample is on the margin
  - If $\alpha_n = C$ the sample can be inside the margin and be either correctly classified ($\xi_n \leq 1$) or misclassified ($\xi_n > 1$)

In the soft-margin case all the points either on the margin or inside the margin are called support vectors. We can tell if a point is on the margin or not depending on the value of α that corresponds to that point

Alternative formulation: ν-SVM

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

Alternative formulation that allows us to get rid of C. This is good because C is not related to anything in the problem and so, tuning C would mean just make a guess.

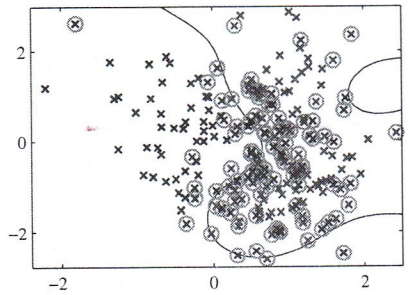
- Where $0 \leq \nu < 1$ is a user parameter that allow to control both the margin errors and the number of support vectors:

fraction of Margin Errors ≤ ν ≤ fraction of SVs

Suppose N = 1000 and ν = 0.1 ! The number of support vectors that we'll find among the 1000 points will be ≥ N·ν = 100. Moreover N·ν will be also the upper bound of the misclassified points.

Example

- An example of ν-SVM discriminant function using Gaussian kernel function



Training SVM in Practice

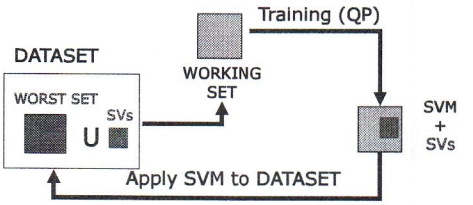
SVM Training

- Just solve the optimization problem to find α_i and b
- ... but it is very expensive: $O(n^3)$ where n is the size of training set
- Faster approaches:
  - Chunking
  - Osuna's methods
  - Sequential Minimal Optimization
- Online learning:
  - Chunking-based methods
  - Incremental methods

we introduced SVM because we wanted something fast for prediction, however SVM is very computationally heavy in training → we can change it

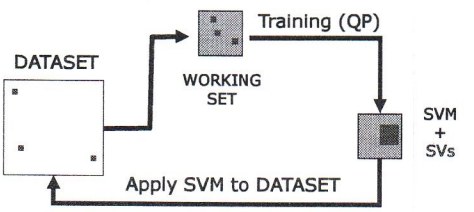
All these methods are based on the idea of considering a subset instead of the whole dataset (we won't go into details)

Chunking



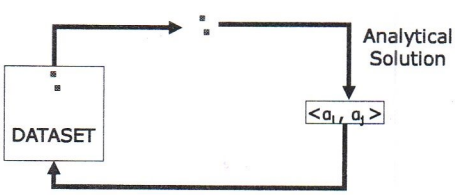
- Solves iteratively a sub-problem (working set)
- Build the working set with current SVs and the M samples with the bigger error (worst set)
- Size of the working set may increase!
- Converges to optimal solution!

Osuna's Method



- Solves iteratively a sub-problem (working set)
- Replace some samples in the working set with missclassified samples in data set
- Size of the working set is fixed!
- Converges to optimal solution!

Sequential Minimal Optimization

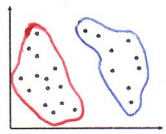


- Works iteratively only on two samples
- Size of the working set is minimal and multipliers are found analytically
- Converges to optimal solution!

Multi-class SVM

Multi-class SVM

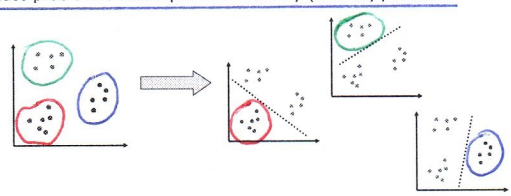
- So far we considered two-classes problems:



- How does SVM deal with multi-class problems?

One-against-all

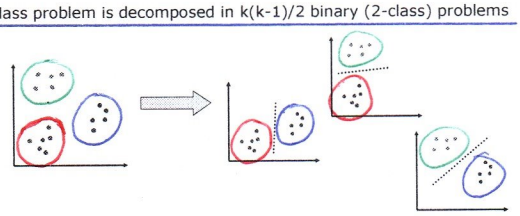
- A k-class problem is decomposed in k binary (2-class) problems



- Training is performed on the entire dataset and involves k SVM classifiers
- Test is performed choosing the class selected with the highest margin among the k SVM classifiers

One-against-one

- A k-class problem is decomposed in k(k-1)/2 binary (2-class) problems

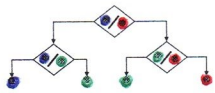


- The k(k-1)/2 SVM classifiers are trained on subsets of the dataset
- Test is performed by applying all the k(k-1)/2 classifiers to the new sample and the most voted label is chosen

DAGSVM

- In DAGSVM, the k-class problem is decomposed in k(k-1)/2 binary (2-class) problems as in one-against-one
- Training is performed as in one-against-one
- But test is performed using a Direct Acyclic Graph to reduce the number of SVM classifiers to apply:

We combine it in a decision tree



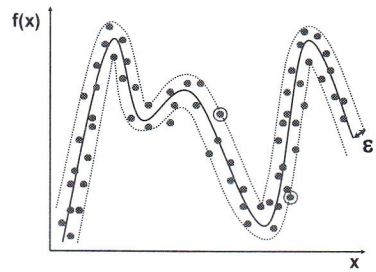
- The test process involves only k-1 binary SVM classifiers instead of k(k-1)/2 as in one-against-one approach

Multi-class SVM: summary

- One-against-all
  - **cheap** in terms of **memory** requirements
  - **expensive training** but **cheap test**
- One-against-one
  - **expensive** in terms of **memory** requirements
  - **expensive test** but **slightly cheap training**
- DAGSVM
  - **expensive** in terms of **memory** requirements
  - **slightly cheap training** and **cheap test**
- **One-against-one** is the **best performing** approach, due to the most effective decomposition
- **DAGSVM** is a **faster approximation** of one-against-one

SVM for Regression

Regression



Regression