DM825

Introduction to Machine Learning

Lecture 9

Support Vector Machines

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Overview

Support Vector Machines:

1. Functional and Geometric Margins
2. Optimal Margin Classifier
3. Lagrange Duality
4. Karush Kuhn Tucker Conditions
5. Solving the Optimal Margin
6. Kernels
7. Soft margins
8. SMO Algorithm

In This Lecture

1. Kernels

2. Soft margins

3. SMO Algorithm

Resume

$$\max_{\vec{\alpha}} \quad W(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y^i y^j \alpha_i \alpha_j \langle \vec{x}^i, \vec{x}^j \rangle$$

$$\text{s.t.} \quad \alpha_i \geq 0 \qquad \forall i = 1 \ldots m$$

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

$$\vec{\theta} = \sum_{i=1}^{m} \alpha_i y^i \vec{x}^i \qquad \forall i = 1 \ldots m$$

$$y_i(\vec{\theta}^T \vec{x}^i + \theta_0) \geq 1 \qquad \forall i = 1 \ldots m$$

$$\alpha_i [y_i(\vec{\theta}^T \vec{x}^i + \theta_0) - 1] = 0 \qquad \forall i = 1 \ldots m$$

Prediction:

$$h(\vec{\theta}, \vec{x}) = \text{sign} \left(\sum_{i=1}^{m} \alpha_i y^i \langle \vec{x}^i, \vec{x} \rangle + \theta_0 \right)$$

Introduction

We saw:

1. $h(\vec{\theta}, \vec{x})$ fitted $\vec{\theta}$ on training data then discarded training data
2. k-NN training data kept during the prediction phase. Memory based method. (fast to train, slower to predict)
3. locally weighted linear regression

$$\vec{\theta} = \text{argmin} \sum_i w_i (y^i - \vec{\theta}^T \vec{x}^i)^2, \quad w^i = \exp\left(-\frac{(\vec{x}^i - \vec{x})^T (\vec{x}^i - \vec{x})}{2\tau^2}\right)$$

(linear parametric method where predictions are based on a linear combination of kernel functions evaluated at training data)

Outline

1. Kernels

2. Soft margins

3. SMO Algorithm

Kernels

$x_1, \ldots, x_D$ inputs
if we want all polynomial terms up to degree 2:

$$\vec{\phi}(\vec{x}) = \begin{bmatrix} x_1^2 & x_2^2 & \ldots & x_D^2 & x_1x_2 & x_1x_3 & \ldots & x_{D-1}x_D \end{bmatrix}^T$$

$\binom{D}{2} = O(D^2)$ terms
For $D = 3$

$$\vec{\phi}(\vec{x}) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_3 \\ x_1^2 \\ x_2^2 \\ x_3^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \sqrt{2}x_2x_3 \end{bmatrix}$$

In SVM we need $\langle \vec{\phi}(\vec{x}^i)^T \cdot \vec{\phi}(\vec{x}^j) \rangle \Longrightarrow O(D^2)$ for m^2 times

$$\vec{\phi}(\vec{x})^T \vec{\phi}(\vec{z}) = 1 + 2\sum_{i=1}^{d} x_i z_i + \sum_{i=1}^{d} x_i^2 z_i^2 + 2\sum_{i=1}^{m} x_i x_j z_i z_j$$

someone recognized that this is the same as $(1 + \vec{x}^T \cdot \vec{z})^2$ which can be computed in $O(D)$.

$$k(\vec{x}, \vec{z}) = (1 + \vec{x}^T \cdot \vec{z})^s \qquad \text{kernel}$$

we may restrict to compute Kernel matrix

Kernels

For models with fixed non linear feature space:

Definition (Kernel)

$k(\vec{x}, \vec{x}') = \vec{\phi}(\vec{x})^T \cdot \vec{\phi}(\vec{x}')$

It follows that $k(\vec{x}, \vec{x}') = k(\vec{x}', \vec{x})$

Kernel Trick

If we have an algorithm in which the input vector $\vec{x}$ enters only in form of scalar products, then we can replace the scalar product with some choice of kernel.

- This is our case with SVM: thanks to dual formulation, both training and prediction can be done via scalar product.
- No need to define features

Constructing Kernels

It must be $k(\vec{x}, \vec{x}') = \vec{x}^T \cdot \vec{x}'$ (scalar product)

1. define some basis functions $\vec{\phi}(\vec{x})$:

$$k(\vec{x}, \vec{x}') = \vec{\phi}(\vec{x})^T \vec{\phi}(\vec{x}') = \sum_{i=1}^{D} \phi_i(\vec{x}) \phi_i(\vec{x}')$$

2. define kernel directly provided it is some scalar product in some feature space (maybe infinite)

$$k(\vec{x}, \vec{x}') = (1 + \vec{x}^T \cdot \vec{x}')^2$$

Constructing Kernels

Following approach 2:

Theorem (Mercer's Kernel)

Necessary and sufficient condition for $k(\cdot)$ to be a valid kernel is that the Gram matrix $\mathbf{k}$, whose elements are $k(\vec{x}^i, \vec{x}^j)$, is positive semidefinite ($\forall x \in \mathbb{R}^n, \vec{x}^T\mathbf{k}\vec{x} \geq 0$) for all choices of the set $\{\vec{x}^i\}$.

Proof:
Symmetry: $k_{ij} = k(\vec{x}^i, \vec{x}^j) = \vec{\phi}(\vec{x}^i)^T\vec{\phi}(\vec{x}^j) = \vec{\phi}(\vec{x}^j)^T\vec{\phi}(\vec{x}^i) = k_{ji}$

$$\begin{aligned}
z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\
&= \sum_i \sum_j z_i \phi(x^{(i)})^T \phi(x^{(j)}) z_j \\
&= \sum_i \sum_j z_i \sum_k \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\
&= \sum_k \sum_i \sum_j z_i \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\
&= \sum_k \left(\sum_i z_i \phi_k(x^{(i)}) \right)^2 \\
&\geq 0.
\end{aligned}$$

Constructing Kernels

One easy way to construct kernels is by recombining building blocks.

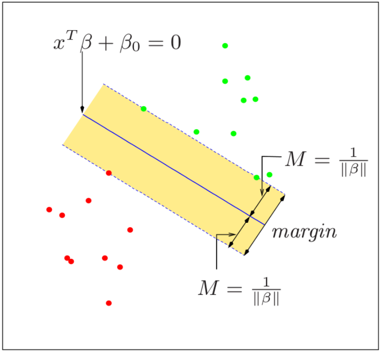
Known building blocks:

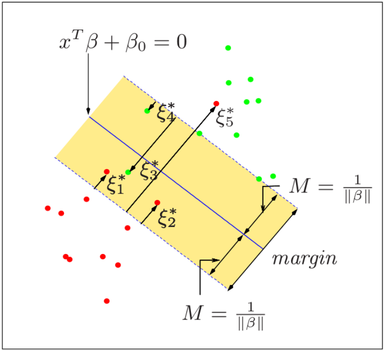
Linear:	$k(\vec{x}, \vec{x}') = \vec{x}^T \vec{x}$
Polynomials:	$k(\vec{x}, \vec{x}') = (\vec{x}^T \vec{x} + c)^s$
radial basis:	$k(\vec{x}, \vec{x}') = \exp(- \parallel \vec{x} - \vec{x}' \parallel^2 / 2\sigma^2)$ (has infinite dimensionality)
sigmoid func.:	$k(\vec{x}, \vec{x}') = \tanh(k \vec{x}^T \vec{x} - \sigma)$

Outline

Soft margins

What if data are not separable?





Soft margins

We allow some points to be on the wrong side and introduce slack variables $\vec{\xi} = (\xi_1 \ldots, \xi_m)$ in the formulation:
geometric margin becomes:

- $y^i(\vec{\theta}^T \vec{x}^i + \theta_0) > 0$ if predicted correct
- $y^i(\vec{\theta}^T \vec{x}^i + \theta_0) > -\xi_i$ for the points mispredicted

In the formulation we modify
$y^i(\vec{\theta}^T \vec{x}^i + \theta_0) > \gamma$ into
$y^i(\vec{\theta}^T \vec{x}^i + \theta_0) > \gamma(1 - \xi_i)$ and include a regularization term to minimize:

$$\text{(OPT)} : \min_{\vec{\theta}, \theta_0} \ \frac{1}{2} \parallel \vec{\theta} \parallel^2 + C \sum_{i=1}^{m} \xi_i$$

$$\begin{array}{lll} \alpha_i : & 1 - \xi_i \leq y^i(\vec{\theta}^T \vec{x}^i + \theta_0) & \forall i = 1, \ldots, m \\ \mu_i : & \xi_i \geq 1 & \forall i = 1, \ldots, m \end{array}$$

still convex optimization

$$\mathcal{L}(\vec{\theta}, \theta_0, \vec{\alpha}, \vec{\mu}) = \frac{1}{2} \parallel \vec{\theta} \parallel^2 + C \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i \left[y^i (\vec{\theta}^T \vec{x}^i + \theta_0) - (1 - \xi_i) \right] - \sum_{i=1}^{m} \mu_i \xi_i$$

fixed $\vec{\alpha}, \vec{\mu}$ we have the primal $\mathcal{L}_P(\vec{\theta}, \theta_0, \vec{\xi})$ which we minimize in $\vec{\theta}, \theta_0, \vec{\xi}$:

$$\nabla_{\vec{\theta}} \mathcal{L}_P = 0 \Longrightarrow \vec{\theta} = \sum_{i=1}^{m} \alpha_i y^i x^i$$

$$\frac{\partial \mathcal{L}_P}{\partial \theta_0} = 0 \Longrightarrow 0 = \sum_{i=1}^{m} \alpha_i y^i$$

$$\frac{\partial \mathcal{L}_P}{\partial \xi_i} = 0 \Longrightarrow \alpha_i = C - \mu_i \quad \forall i$$

Lagrange dual:

$$\mathcal{L}_D = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \vec{x}_i^T \vec{x}_j$$

$$\max \mathcal{L}_D = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \vec{x}_i^T \vec{x}_j \tag{1}$$

$$0 \leq \alpha_i \leq C \tag{2}$$

$$\sum_{i=1}^{m} \alpha_i y^i = 0 \tag{3}$$

$$\alpha_i [y_i(\vec{x}_i^T \vec{\theta} + \theta_0) - (1 - \xi_i)] = 0 \tag{4}$$

$$\mu_i \xi_i = 0 \tag{5}$$

$$y_i(\vec{x}_i^T \vec{\theta} + \theta_0) - (1 - \xi_i) \geq 0 \tag{6}$$

$$\mu_i \geq 0, \quad \xi_i \geq 0 \tag{7}$$

for (5) + $\frac{\partial \mathcal{L}_P}{\partial \xi_i} = 0$ support vectors are:

- the points that lie on the edge of the margin ($\xi_i = 0$) and hence $\Longrightarrow 0 < \alpha_i < C$
- the misclassified points $\xi_i > 0$ that have $\alpha_i = C$

The margin points can be used to solve (4) for θ_0

Outline

1. Kernels

2. Soft margins

3. SMO Algorithm

Coordinate ascent

$$\max_{\vec{\alpha}} W(\alpha_1, \alpha_2, \ldots, \alpha_m)$$

repeat
  for i=1,...,m **do**
    $\alpha_i := \arg\max_{\hat{\alpha}_i} W(\alpha_1, \ldots, \alpha_{i-1} \hat{\alpha}_i, \alpha_{i+1}, \ldots, \alpha_m)$
until till convergence ;

Sequential Minimal Optimization

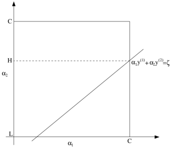
$$\max_{\vec{\alpha}} W(\alpha_1, \alpha_2, \ldots, \alpha_m)$$

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

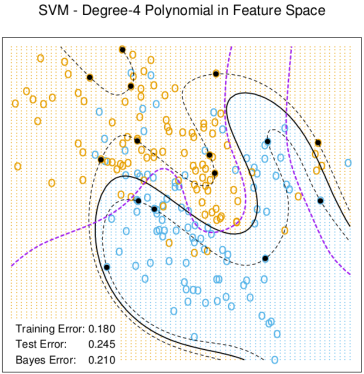
Fix and change two αs at a time.

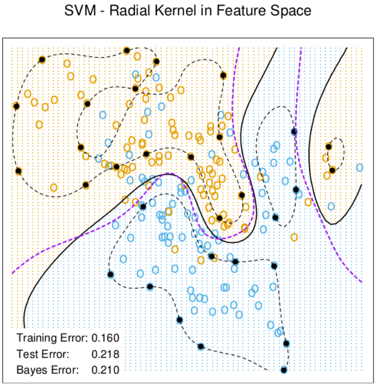
repeat
  select α_i and α_j by some heuristic;
  hold all α_l, $l \neq i, j$ fixed and optimize $W(\vec{\alpha})$ in α_i, α_j
until till convergence ;

$\alpha_1 y^1 + \alpha_2 y^2 = -\sum_{i=3}^{m} \alpha_i y^i = \text{const} \Longrightarrow \alpha_1 = \frac{C - \alpha_2 y^2}{y^1}$



Example


SVM - Degree-4 Polynomial in Feature Space
Training Error: 0.180
Test Error: 0.245
Bayes Error: 0.210


SVM - Radial Kernel in Feature Space
Training Error: 0.160
Test Error: 0.218
Bayes Error: 0.210

SVM for K-Classes

1. train K SVM each SVM classifies one class from all the others.

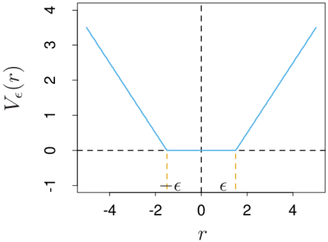
2. choose the indication of the SVM that makes the strongest prediction: where the basis vector input point is furthest into positive region

SVM for regression

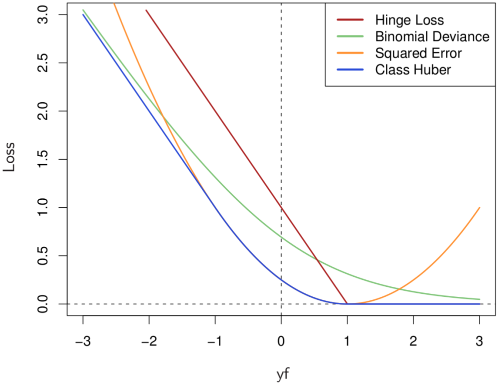
With a quantitative response we try to fit as much as possible within the margin change, hence we change the objective function in (OPT3) into:

$$\min \sum_{i=1}^{m} V(y^i - f(x^i)) + \frac{\lambda}{2} \parallel \vec{\theta} \parallel^2$$

$$V_\epsilon = \begin{cases} 0 & if|r| < \epsilon \\ |r| - \epsilon & \text{otherwise} \end{cases}$$



SVM as Regularized Function



Loss Function	$L[y, f(x)]$	Minimizing Function
Binomial Deviance	$\log[1 + e^{-yf(x)}]$	$f(x) = \log \frac{\Pr(Y = +1\|x)}{\Pr(Y = -1\|x)}$
SVM Hinge Loss	$[1 - yf(x)]_+$	$f(x) = \text{sign}[\Pr(Y = +1\|x) - \frac{1}{2}]$
Squared Error	$[y - f(x)]^2 = [1 - yf(x)]^2$	$f(x) = 2\Pr(Y = +1\|x) - 1$