Lecture 11: Kernel Methods (Part II)

SVM revisited

Machine Learning, Summer Term 2019

Michael Tangermann Frank Hutter Marius Lindauer

University of Freiburg

Lecture Overview

1. Motivation: Shortcomings of Linear Models
2. Optimization Problem of the Optimal Hyperplane Classifier
3. Support Vector Machine (SVM)
4. A Bit More On Kernels

Lecture Overview

1. Motivation: Shortcomings of Linear Models
2. Optimization Problem of the Optimal Hyperplane Classifier
3. Support Vector Machine (SVM)
4. A Bit More On Kernels

Shortcomings of Linear Models (I)

- Linear models (e.g. LDA for classification, OLS regression) make strong **assumptions about data distributions**.
- Linear models require data points (objects) to be vectors from **Euclidian vector space** in order to calculate distances, similarities etc.
- Linear models intrinsically can not deal with **non-linear problems**

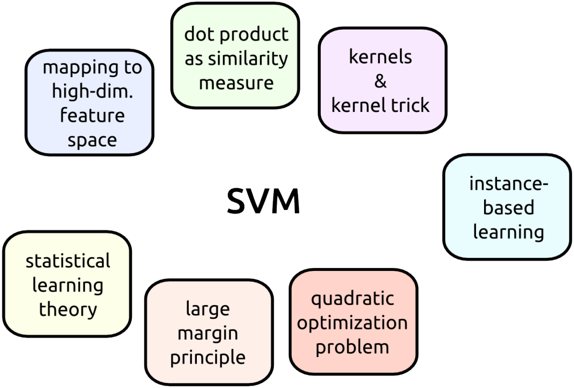
Shortcomings of Linear Models (II)

Using domain knowledge, non-linear feature pre-processing can add helpful extra dimensions, leading to (potentially very) high dimensionality.

This is problematic:

- Large dimensionality but limited data $\rightarrow$ **numerical and computational burden**.
- **Overfitting** may become a problem!
  (how could we control for it?)

SVM Mitigates Shortcomings of Linear Methods



Lecture Overview

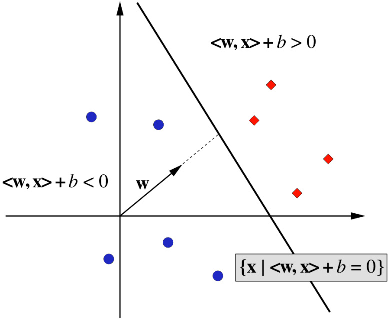
Reminder: Hyperplane Classifiers (Linearly Separable Case)

Let's consider the function class of hyperplanes in a dot product space $\mathcal{H}$:

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$$

where $\mathbf{w}, \mathbf{x} \in \mathcal{H}, b \in \mathbb{R}$, and the decision function is

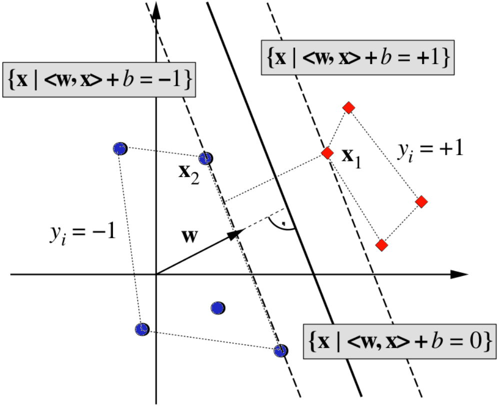
$$f(\mathbf{x}) = sgn\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right)$$



Reminder: Find the **Optimal** Hyperplane Classifier

(For linearly separable problem:) Vapnik et al. proposed to use the unique hyperplane with the maximum margin of separation between any training point $\mathbf{x}_i$ and the hyperplane:

$$\underset{\mathbf{w}\in\mathcal{H},\, b\in\mathbb{R}}{\text{argmax}}\ min\{\|\mathbf{x}-\mathbf{x}_i\| \mid \mathbf{x}\in\mathcal{H},\ \langle\mathbf{w},\mathbf{x}\rangle + b = 0,\ i=1,\ldots,m\}$$



Reminder: The Constrained Optimization Problem

Vapnik's formulation of the optimal hyperplane classifier (separable case) led to the following constrained optimization problem:

$$\underset{\mathbf{w}\in\mathcal{H},\, b\in\mathbb{R}}{\text{argmin}} \; \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

subject to $$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \text{for all} \quad i = 1, \ldots, m.$$

- Function τ is called the **objective function**
- Constraints are **inequality constraints**, which ensure, that the labels will be correct
- The "≥ 1" on the right hand side of the constraints effectively fix the scaling of $\mathbf{w}$.

Solving the Constrained Optimization Problem

$$\underset{\mathbf{w}\in\mathcal{H},\, b\in\mathbb{R}}{\text{argmin}}\ \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

subject to $\quad y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1 \quad \text{for all} \quad i = 1, \ldots, m.$

Constrained optimization problems can be solved by introducing Lagrange multipliers $\alpha_i \in \mathbb{R}$, $\alpha_i \geq 0$, and by **minimizing** the Lagrangian:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i\ (\langle \mathbf{x}_i, \mathbf{w}\rangle + b) - 1)$$

Brief Excursion: Lagrangian Multipliers

Lagrange multipliers are a concept used in optimization.

- For a very nice explanation of Lagrange multipliers, please see `https://en.wikipedia.org/wiki/Lagrange_multiplier`
- The simplest case (with equality constraints) and the calculations for example 1 are rather intuitive.
- If you like to follow a career in machine learning, then it may be worth while taking a full course on optimization.
  ($\rightarrow$ Prof. Diehl, online lecture from winter semester 2015)

The Lagrangian of the Constrained Optimization Problem

$$\text{minimize } L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i \left(\langle \mathbf{x}_i, \mathbf{w} \rangle + b\right) - 1)$$

Observe:

- Constraints have been included into the second part of the Lagrangian
- What do we need to maximize/minimize in order to minimize L?

The Lagrangian of the Constrained Optimization Problem

$$\text{minimize } L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i (\langle \mathbf{x}_i, \mathbf{w} \rangle + b) - 1)$$

Observe:

- Constraints have been included into the second part of the Lagrangian
- What do we need to maximize/minimize in order to minimize L?

Answer:

- Minimize the Lagrangian L with respect to the primal variables $\mathbf{w}$ and b
- Maximize L with respect to the dual variables α_i.

(Effectively, this finds a solution in a saddle point.)

The Lagrangian of the Constrained Optimization Problem

$$\text{minimize } L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i (\langle \mathbf{x}_i, \mathbf{w} \rangle + b) - 1)$$

Observe:

- Constraints have been included into the second part of the Lagrangian
- What do we need to maximize/minimize in order to minimize L?

Answer:

- Minimize the Lagrangian L with respect to the primal variables $\mathbf{w}$ and b
- Maximize L with respect to the dual variables α_i.

(Effectively, this finds a solution in a saddle point.)

How would you go ahead with this problem? 

Solving the Constrained Optimization Problem

$$\text{minimize } L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i (\langle \mathbf{x}_i, \mathbf{w} \rangle + b) - 1)$$

Minimize the Lagrangian L with respect to the primal variables $\mathbf{w}$ and b
$\rightarrow$ the partial derivatives of L with respect to the primal variables must vanish. Thus set:

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \quad \text{and} \quad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0$$

This leads to:

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

Observe: The solution vector $\mathbf{w}$ has an expansion in terms of a subset of the training patterns.

From Primal to Dual Problem

Eliminate the primal variables $\mathbf{w}$ and b from the optimization problem by substituting

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

into the Lagrangian.

In most practical applications of SVMs, this resulting dual optimization problem is preferred for solving:

$$\operatorname*{argmax}_{\boldsymbol{\alpha} \in \mathbb{R}^m} \; W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

subject to $\quad \alpha_i \geq 0 \quad$ for all $i = 1, \ldots, m \quad$ and $\quad \sum_{i=1}^{m} \alpha_i y_i = 0.$

For a discussion of solvers for quadratic programs, see contribution by John Platt in the paper by Hearst et al., IEEE Intelligent Systems, 1995.

Advantage of a Dual Representation

Making use of $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$ once more, the hyperplane decision function for a novel data point $\mathbf{x}$ can be written as:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b\right)$$

Observe:

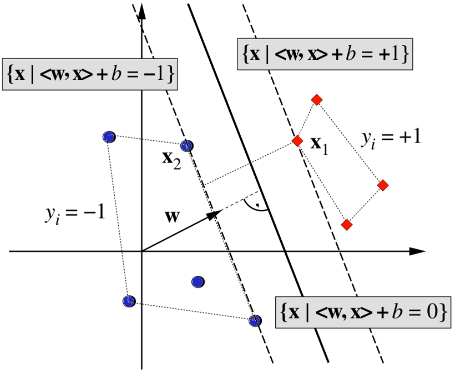
- In dual formulation, the solution can be expressed completely in terms of dot products of training data points ("**instance based learning**"), cp. to e.g. k-Nearest Neighbour algorithm.
- In practical problems, only a subset of the Lagrange multipliers α_i are active (i.e. non-zero) and influence the decision hyperplane; corresponding training data points $\mathbf{x}_i$ are called **support vectors**.
- Other training data points have $\alpha_i = 0$. They do not define the shape of the hyperplane (see mechanical analogy from the beginning of the lecture).

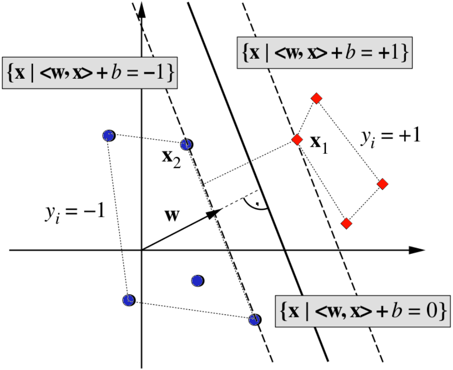
Support Vectors are on the Margin

By the Karush-Kuhn-Tucker (KKT) conditions (see "Learning with Kernels", Chap. 6)

$$\alpha_i[y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) - 1] = 0 \quad \text{for all} \quad i = 1, ..., m$$

the SVs of a separable training data set lie on the margin:



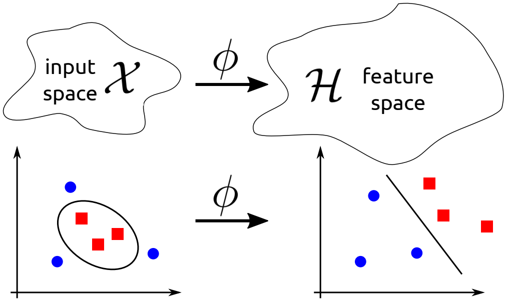


 Practical aspect of SVs: It is interesting, which ratio of the training patterns end up as SVs, in order to estimate the classification performance on unseen data. **Why?**
(Hint: Drawing + consider a LOO cross-validation.)

Lecture Overview

Reminder Support Vector Machine

- SVMs implement the large margin principle and thus share the optimization problem of optimal hyperplane classifiers.
- They make use of nonlinear mappings:



- ... but don't compute the dot product in feature space explicitly! (kernel trick)

Going Non-Linear: Decision Function for SVM

Decision function for SVM including the kernel trick:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \langle \phi(x), \phi(x_i) \rangle + b\right) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i k(x, x_i) + b\right)$$

Going Non-Linear: Decision Function for SVM

Decision function for SVM including the kernel trick:

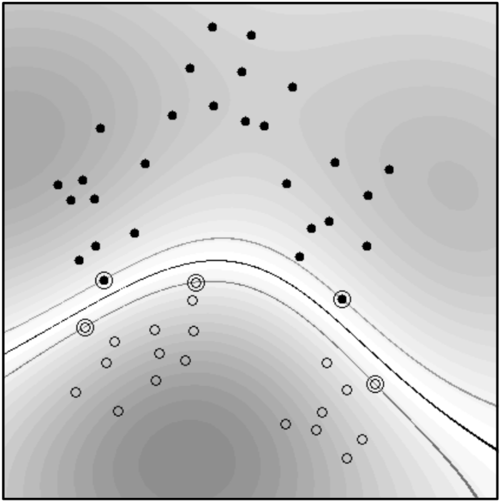
$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \langle \phi(x), \phi(x_i) \rangle + b\right) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i k(x, x_i) + b\right)$$

This decision function for the **non-linear** SVM can again be derived by optimizing the α_i in a quadratic program. (Blue color indicates the changes compared to the linear hyperplane quadratic program!):

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{maximize}} \; W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to $\quad \alpha_i \geq 0 \quad$ for all $i = 1, \ldots, m \quad$ and $\quad \sum_{i=1}^{m} \alpha_i y_i = 0$

Example of SVM Classifier with Radial Basis Function Kernel



Lecture Overview

Recap: Examples of Kernels

Assumption: Input space already has a vectorial representation. (ϕ is identity.)

Polynomial kernel (hyperparameter: d):

$$k(x, x') = \left\langle x, x' \right\rangle^d$$

Gaussian radial basis function kernels (parameter $\sigma > 0$):

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

Sigmoid kernel (parameter $k > 0$ and parameter $\Theta < 0$):

$$k(x, x') = \tanh\left(k \left\langle x, x' \right\rangle + \Theta\right)$$

RBF Kernel Revisited

The RBF kernel is often mentioned as an example, how the SVM can *implicitly* project data to an infinite-dimensional feature space.

Intuition:

- RBF (Gaussian) kernel

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

  is a simple modification of this kernel:

$$k'(x, x') = \exp\left(-\frac{\langle x, x'\rangle}{\sigma^2}\right)$$

- Use power series expansion:

$$k'(x, x') = \sum_{n=0}^{+\infty} \frac{\langle x, x'\rangle^n}{\sigma^2 n!}$$

- Does this numerator look familiar to you?

RBF Kernel Revisited

$$k'(x, x') = \sum_{n=0}^{+\infty} \frac{\langle x, x' \rangle^n}{\sigma^2 n!}$$

- Numerator contains a polynomial kernel of degree n.
- $\rightarrow$ RBF kernel can be seen as a combination of all polynomial kernels of degree $n \geq 0$.
- Please note: RBF kernel maps to this infinite-dimensional feature space only *implicitly*.

Kernel Trick Revisited (I)

Why can we calculate something **implicitly** in a high-dimensional feature space?

Intuition with an example:
Given we have a mapping ϕ, mapping from $\mathbb{R}^3$ to $\mathbb{R}^9$:
$\phi(x_1, x_2, x_3) = (x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3,)$

-  $\langle \mathbf{x}, \mathbf{x}' \rangle$
  How many operations are necessary to perform the mapping to $\mathbb{R}^9$ and to calculate the dot product?

Kernel Trick Revisited (I)

Why can we calculate something **implicitly** in a high-dimensional feature space?

Intuition with an example:
Given we have a mapping ϕ, mapping from $\mathbb{R}^3$ to $\mathbb{R}^9$:
$\phi(x_1, x_2, x_3) = (x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3,)$

-  $\langle \mathbf{x}, \mathbf{x}' \rangle$
  How many operations are necessary to perform the mapping to $\mathbb{R}^9$ and to calculate the dot product?
- 9+9 multiplications for the mapping, 9 multiplications and 8 additions for the dot product $\rightarrow$ **35** operations.

Kernel Trick Revisited (II)

Same situation, $x, x' \in \mathbb{R}^3$

- Now consider using the polynomial kernel function of degree 2 instead of explicit mapping using ϕ:
  $k(x, x') = \langle x, x' \rangle^2$
-  How many operations are necessary now?

Kernel Trick Revisited (II)

Same situation, $x, x' \in \mathbb{R}^3$

- Now consider using the polynomial kernel function of degree 2 instead of explicit mapping using ϕ:
  $k(x, x') = \langle x, x' \rangle^2$
-  How many operations are necessary now?
- 3 multiplications and two additions for the dot product, one multiplication for the squaring $\rightarrow$ **6** operations.

This shows, how kernels help to save computation time, **if dot products ONLY** are required in the high-dimensional space (but nothing else).

Wrap-Up: Summary by Learning Goals

Having heard this lecture and working on the SVM assignment, you will be able to:

- formulate the optimization problems for
  - optimal large-margin hyperplane classifiers
  - SVM
  - soft-margin SVM
- obtain the Lagrange formulations
- explain how to get from the primal to the dual formulations
- Give examples, why the kernel trick is useful.