These are the slides of the lecture

Machine Learning
Summer term 2017
University of Applied Sciences Rosenheim

Based on the slides of the lecture *Pattern Recognition* taught at the FAU Erlangen-Nuremberg, courtesy of D. Hahn, J. Hornegger, S. Steidl and E. Nöth.

Rosenheim, May 9, 2018
Prof. Dr.-Ing. Korbinian Riedhammer

Machine Learning (ML)

Summer Term 2017

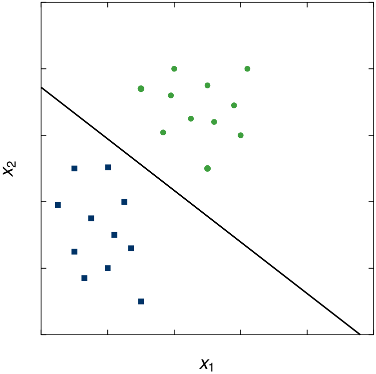
Riedhammer
Fakultät für Informatik
Hochschule Rosenheim

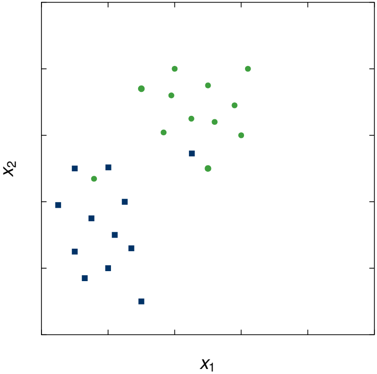
Motivation

- Assume two linearly separable classes.
- Computation of linear decision boundary that allows the separation of training data and that generalizes well.
- Vapnik 1996: Optimal separating hyperplane separates two classes and maximizes the distance to the closest point from either class. This results in
  - unique solution for hyperplanes, and
  - (in most cases) better generalization.

Motivation (cont.)

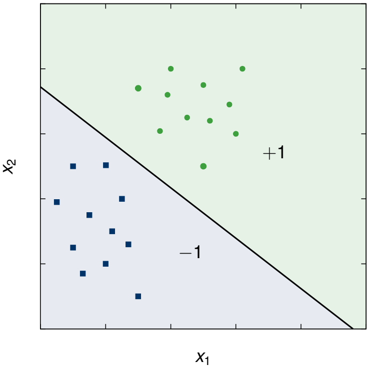
Linearly separable and non-separable classes





Motivation (cont.)

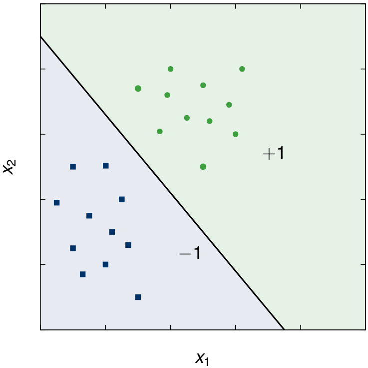
Many, many, many solutions ...



Idea: Average the perceptron solutions.

Motivation (cont.)

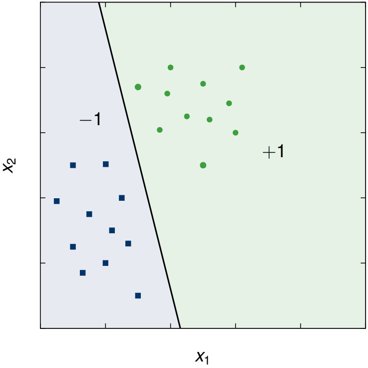
Many, many, many solutions . . .



Idea: Average the perceptron solutions.

Motivation (cont.)

Many, many, many solutions . . .

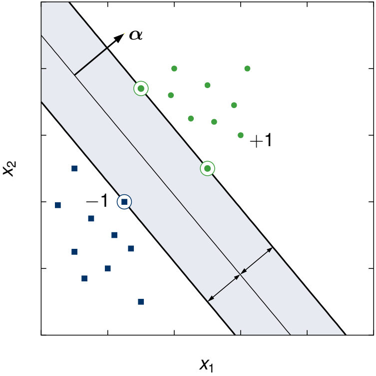


Idea: Average the perceptron solutions.

Motivation (cont.)

We distinguish between:

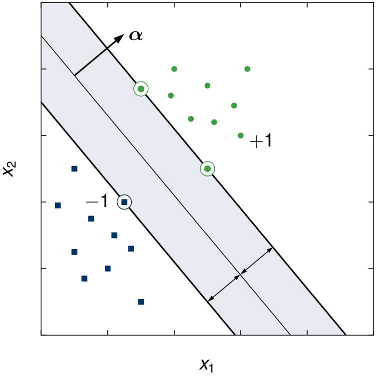
1. Hard margin problem



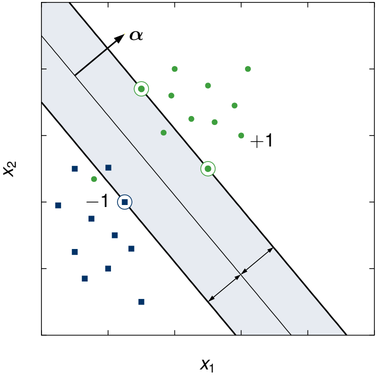
Motivation (cont.)

We distinguish between:

1. Hard margin problem



2. Soft margin problem



Remarks on Linear Algebra

Assume we have an affine function that defines the decision boundary:

Remarks on Linear Algebra

Assume we have an affine function that defines the decision boundary:

$$f(\boldsymbol{x}) = \boldsymbol{\alpha}^T \boldsymbol{x} + \alpha_0$$

Remarks on Linear Algebra

Assume we have an affine function that defines the decision boundary:

$$f(\boldsymbol{x}) \quad = \quad \boldsymbol{\alpha}^T \boldsymbol{x} + \alpha_0$$

- For any point $\boldsymbol{x}$ on the hyperplane, we have $f(\boldsymbol{x}) = 0$.

Remarks on Linear Algebra

Assume we have an affine function that defines the decision boundary:

$$f(\boldsymbol{x}) \quad = \quad \boldsymbol{\alpha}^T \boldsymbol{x} + \alpha_0$$

- For any point $\boldsymbol{x}$ on the hyperplane, we have $f(\boldsymbol{x}) = 0$.
- A necessary condition for two points on the hyperplane is:

$$f(\boldsymbol{x}_1) = f(\boldsymbol{x}_2) \quad \text{and thus} \quad \boldsymbol{\alpha}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0.$$

Remarks on Linear Algebra

Assume we have an affine function that defines the decision boundary:

$$f(\boldsymbol{x}) = \boldsymbol{\alpha}^T \boldsymbol{x} + \alpha_0$$

- For any point $\boldsymbol{x}$ on the hyperplane, we have $f(\boldsymbol{x}) = 0$.
- A necessary condition for two points on the hyperplane is:

$$f(\boldsymbol{x}_1) = f(\boldsymbol{x}_2) \quad \text{and thus} \quad \boldsymbol{\alpha}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0.$$

- The normal vector $\boldsymbol{n}$ of the hyperplane is $\boldsymbol{n} = \boldsymbol{\alpha}/\|\boldsymbol{\alpha}\|_2$.

Remarks on Linear Algebra (cont.)

- The signed distance d of a point $\boldsymbol{x}$ to the hyperplane is:

Remarks on Linear Algebra (cont.)

- The signed distance d of a point $\boldsymbol{x}$ to the hyperplane is:

$$d = \frac{\|\boldsymbol{\alpha}\|_2}{\boldsymbol{\alpha}^T\boldsymbol{\alpha}} \cdot f(\boldsymbol{x}) = \frac{1}{\|\boldsymbol{\alpha}\|_2} \cdot f(\boldsymbol{x}) = \frac{1}{\|\nabla f(\boldsymbol{x})\|_2} \cdot f(\boldsymbol{x})$$

Remarks on Linear Algebra (cont.)

- The signed distance d of a point $\boldsymbol{x}$ to the hyperplane is:

$$d = \frac{\|\boldsymbol{\alpha}\|_2}{\boldsymbol{\alpha}^T\boldsymbol{\alpha}} \cdot f(\boldsymbol{x}) = \frac{1}{\|\boldsymbol{\alpha}\|_2} \cdot f(\boldsymbol{x}) = \frac{1}{\|\nabla f(\boldsymbol{x})\|_2} \cdot f(\boldsymbol{x})$$

- Assume points $\boldsymbol{x}_1, \boldsymbol{x}_2$ on either side of the margin satisfy $f(\boldsymbol{x}_1) = +1$ and $f(\boldsymbol{x}_2) = -1$.

  Thus we have:

Remarks on Linear Algebra (cont.)

- The signed distance d of a point $\boldsymbol{x}$ to the hyperplane is:

$$d = \frac{\|\boldsymbol{\alpha}\|_2}{\boldsymbol{\alpha}^T\boldsymbol{\alpha}} \cdot f(\boldsymbol{x}) = \frac{1}{\|\boldsymbol{\alpha}\|_2} \cdot f(\boldsymbol{x}) = \frac{1}{\|\nabla f(\boldsymbol{x})\|_2} \cdot f(\boldsymbol{x})$$

- Assume points $\boldsymbol{x}_1, \boldsymbol{x}_2$ on either side of the margin satisfy $f(\boldsymbol{x}_1) = +1$ and $f(\boldsymbol{x}_2) = -1$.

  Thus we have:

$$\boldsymbol{\alpha}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 2 \quad \text{and} \quad \frac{\boldsymbol{\alpha}^T}{\|\boldsymbol{\alpha}\|_2}(\boldsymbol{x}_1 - \boldsymbol{x}_2) = \frac{2}{\|\boldsymbol{\alpha}\|_2}$$

Constrained Optimization Problem

Constraints:

- Separation of classes has to be done with margin:

$$\begin{aligned} \boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0 &\leq -1, \quad \text{if} \quad y_i = -1 \\ \boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0 &\geq +1, \quad \text{if} \quad y_i = +1 \end{aligned}$$

Constrained Optimization Problem

Constraints:

- Separation of classes has to be done with margin:

$$\begin{aligned} \boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0 &\leq -1, \quad \text{if} \quad y_i = -1 \\ \boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0 &\geq +1, \quad \text{if} \quad y_i = +1 \end{aligned}$$

- This is equivalent to:

$$y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) \geq 1$$

Constrained Optimization Problem (cont.)

The maximization of the margin corresponds to the following optimization problem with linear constraints:

$$\begin{array}{ll} \text{maximize} & \dfrac{1}{\|\boldsymbol{\alpha}\|_2} \\ \text{subject to} & y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) \geq 1, \quad \text{for all} \quad i \end{array}$$

Constrained Optimization Problem (cont.)

The maximization of the margin corresponds to the following optimization problem with linear constraints:

$$
\begin{aligned}
&\text{maximize} && \frac{1}{\|\boldsymbol{\alpha}\|_2} \\
&\text{subject to} && y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) \geq 1, \quad \text{for all} \quad i
\end{aligned}
$$

Note:

- Linear constraints ensure that all feature vectors have maximum distance to decision boundary.

Constrained Optimization Problem (cont.)

The maximization of the margin corresponds to the following optimization problem with linear constraints:

$$\begin{aligned} &\text{maximize} && \frac{1}{\|\boldsymbol{\alpha}\|_2} \\ &\text{subject to} && y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) \geq 1, \quad \text{for all} \quad i \end{aligned}$$

Note:

- Linear constraints ensure that all feature vectors have maximum distance to decision boundary.
- Basically we compute the distance of the convex hulls of feature sets.

Constrained Optimization Problem (cont.)

The maximization of the margin corresponds to the following optimization problem with linear constraints:

$$
\begin{array}{ll}
\text{maximize} & \dfrac{1}{\|\boldsymbol{\alpha}\|_2} \\
\text{subject to} & y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) \geq 1, \quad \text{for all} \quad i
\end{array}
$$

Note:

- Linear constraints ensure that all feature vectors have maximum distance to decision boundary.
- Basically we compute the distance of the convex hulls of feature sets.
- We need constrained optimization methods to solve the problem.

Constrained Optimization Problem (cont.)

The optimization problem is equivalent to

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\boldsymbol{\alpha}\|_2^2 \\ \text{subject to} & y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) - 1 \geq 0, \quad \text{for all} \quad i \end{array}$$

Constrained Optimization Problem (cont.)

Remarks on the optimization problem:

- Convex optimization problem
- Efficient algorithms for solving the convex optimization problem (interior point method)
- Standard libraries can be used for minimization
- Solution is unique

Non-linearly Separable Classes

If classes are not linearly separable, we have to introduce *slack variables*.

Non-linearly Separable Classes

If classes are not linearly separable, we have to introduce *slack variables*.

Convex optimization problem:

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{\alpha}\|_2^2 + \mu \sum_i \xi_i$$

$$\text{subject to} \quad \forall i : \quad -(y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) - 1 + \xi_i) \leq 0 \,,$$

$$\forall i : \quad -\xi_i \leq 0$$

Comprehensive Questions

- What is the concept of a SVM?
- What is the difference between a hard and soft margin SVM?
- What is the convex optimization problem of the hard margin SVM?
- What is the convex optimization problem of the soft margin SVM?

Hard Margin Problem

The hard margin SVM optimization problem is formulated as:

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{\alpha}\|_2^2$$

$$\text{subject to} \quad \forall i: \quad y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) - 1 \geq 0$$

Soft Margin Problem

The soft margin SVM optimization problem is formulated as:

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{\alpha}\|_2^2 + \mu \sum_i \xi_i$$

$$\begin{aligned} \text{subject to} \quad \forall i : & \quad -(y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) - 1 + \xi_i) \le 0 \,, \\ \forall i : & \quad -\xi_i \le 0 \end{aligned}$$

Lagrangian

The solution of the constrained convex optimization problem requires the Lagrangian:

$$\begin{aligned} L(\boldsymbol{\alpha}, \alpha_0, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \quad &= \quad \frac{1}{2}\|\boldsymbol{\alpha}\|_2^2 + \mu \sum_i \xi_i - \sum_i \mu_i \xi_i \\ &\quad - \sum_i \lambda_i (y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) - 1 + \xi_i) \end{aligned}$$

Lagrangian (cont.)

Partial derivatives I:

$$\frac{\partial L(\boldsymbol{\alpha}, \alpha_0, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu})}{\partial \boldsymbol{\alpha}} = \boldsymbol{\alpha} - \sum_i \lambda_i y_i \boldsymbol{x}_i \stackrel{!}{=} \mathbf{0}.$$

Thus we have:

$$\boldsymbol{\alpha} = \sum_i \lambda_i y_i \boldsymbol{x}_i \ .$$

Lagrangian (cont.)

Partial derivatives II:

$$\frac{\partial L(\boldsymbol{\alpha}, \alpha_0, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu})}{\partial \alpha_0} = -\sum_i \lambda_i y_i \stackrel{!}{=} 0$$

Lagrangian (cont.)

Partial derivatives II:

$$\frac{\partial L(\boldsymbol{\alpha}, \alpha_0, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu})}{\partial \alpha_0} = -\sum_i \lambda_i y_i \stackrel{!}{=} 0$$

Partial derivatives III:

$$\frac{\partial L(\boldsymbol{\alpha}, \alpha_0, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu})}{\partial \xi_i} = \mu - \mu_i - \lambda_i \stackrel{!}{=} 0$$

Lagrange Dual

Let us consider the Lagrange function for the dual problem for the hard margin case:

$$L_D = \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha} - \sum_i \lambda_i(y_i \cdot (\boldsymbol{\alpha}^T\boldsymbol{x}_i + \alpha_0) - 1)$$

Lagrange Dual

Let us consider the Lagrange function for the dual problem for the hard margin case:

$$
\begin{aligned}
L_D &= \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha} - \sum_i \lambda_i (y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) - 1) \\
&= \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha} - \underbrace{(\sum_i \lambda_i y_i \cdot \boldsymbol{x}_i)^T}_{\boldsymbol{\alpha}^T} \boldsymbol{\alpha} - \underbrace{\sum_i \lambda_i y_i}_{=0} \ \alpha_0 + \sum_i \lambda_i
\end{aligned}
$$

Lagrange Dual

Let us consider the Lagrange function for the dual problem for the hard margin case:

$$
\begin{aligned}
L_D &= \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha} - \sum_i \lambda_i(y_i \cdot (\boldsymbol{\alpha}^T\boldsymbol{x}_i + \alpha_0) - 1) \\
&= \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha} - \underbrace{(\sum_i \lambda_i y_i \cdot \boldsymbol{x}_i)^T}_{\boldsymbol{\alpha}^T}\boldsymbol{\alpha} - \underbrace{\sum_i \lambda_i y_i}_{=0}\ \alpha_0 + \sum_i \lambda_i \\
&= -\frac{1}{2}\sum_i\sum_j \lambda_i\lambda_j y_i y_j \cdot \boldsymbol{x}_i^T\boldsymbol{x}_j + \sum_i \lambda_i
\end{aligned}
$$

The Lagrange Dual Problem

The Lagrange dual problem is given the optimization problem:

$$\text{maximize} \quad -\frac{1}{2}\sum_i \sum_j \lambda_i \lambda_j y_i y_j \cdot \boldsymbol{x}_i^T \boldsymbol{x}_j + \sum_i \lambda_i$$

$$\text{subject to} \quad \boldsymbol{\lambda} \succeq 0$$

Lagrange Dual Problem (cont.)

For strong convex functions the duality gap is zero,
if the KKT conditions are satisfied.

Lagrange Dual Problem (cont.)

For strong convex functions the duality gap is zero,
if the KKT conditions are satisfied.

Especially the complementary slackness condition is interesting for us:

$$\forall i: \quad \lambda_i(y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) - 1) = 0$$

Lagrange Dual Problem (cont.)

For strong convex functions the duality gap is zero,
if the KKT conditions are satisfied.

Especially the complementary slackness condition is interesting for us:

$$\forall i : \quad \lambda_i(y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) - 1) = 0$$

Conclusion:

1. If $\lambda_i > 0$, then $y_i(\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) - 1 = 0$, and thus:

   $$y_i(\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) = 1 \ .$$

   All $\boldsymbol{x}_i$ with $\lambda_i > 0$ are elements of the boundary of the slab.
   These $\boldsymbol{x}_i$'s are called *support vectors*.

Lagrange Dual Problem (cont.)

For strong convex functions the duality gap is zero,
if the KKT conditions are satisfied.

Especially the complementary slackness condition is interesting for us:

$$\forall i : \quad \lambda_i(y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) - 1) = 0$$

Conclusion:

1. If $\lambda_i > 0$, then $y_i(\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) - 1 = 0$, and thus:

   $$y_i(\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0) = 1 \ .$$

   All $\boldsymbol{x}_i$ with $\lambda_i > 0$ are elements of the boundary of the slab.
   These $\boldsymbol{x}_i$'s are called *support vectors*.
2. We have seen that $\boldsymbol{\alpha} = \sum_i \lambda_i y_i \boldsymbol{x}_i$, thus the norm vector of the decision boundary is a linear combination of support vectors.

Dual Representation

The decision function can also be rewritten using the duality:

$$f(\boldsymbol{x}) \quad = \quad \boldsymbol{\alpha}^T \boldsymbol{x} + \alpha_0 \quad = \quad \sum_i \lambda_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + \alpha_0$$

Dual Representation

The decision function can also be rewritten using the duality:

$$f(\boldsymbol{x}) \quad = \quad \boldsymbol{\alpha}^T \boldsymbol{x} + \alpha_0 \quad = \quad \sum_i \lambda_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + \alpha_0$$

Conclusion:

Feature vectors only appear in inner products, both in learning and classification phase.

Motivation

Linear decision boudaries in its current form have serious limitations:

Motivation

Linear decision boudaries in its current form have serious limitations:

- too simple to provide good decision boundaries

Motivation

Linear decision boudaries in its current form have serious limitations:

- too simple to provide good decision boundaries
- non-linearly separable data cannot be classified

Motivation

Linear decision boudaries in its current form have serious limitations:

- too simple to provide good decision boundaries
- non-linearly separable data cannot be classified
- noisy data cause problems

Motivation

Linear decision boudaries in its current form have serious limitations:

- too simple to provide good decision boundaries
- non-linearly separable data cannot be classified
- noisy data cause problems
- formulation deals with vectorial data only

Motivation

Linear decision boudaries in its current form have serious limitations:

- too simple to provide good decision boundaries
- non-linearly separable data cannot be classified
- noisy data cause problems
- formulation deals with vectorial data only

Possible solution:

- Map data into higher dimensional feature space using non-linear feature transform, then use a linear classifier.

Dual Representation

- The decision boundary of a support vector machine can be rewritten in dual form:

$$f(\boldsymbol{x}) = \boldsymbol{\alpha}^T \boldsymbol{x} + \alpha_0 = \sum_i \lambda_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + \alpha_0$$

  where we have used the identity:

$$\boldsymbol{\alpha} = \sum_i \lambda_i y_i \boldsymbol{x}_i \ .$$

Dual Representation

- The decision boundary of a support vector machine can be rewritten in dual form:

$$f(\boldsymbol{x}) = \boldsymbol{\alpha}^T \boldsymbol{x} + \alpha_0 = \sum_i \lambda_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + \alpha_0$$

  where we have used the identity:

$$\boldsymbol{\alpha} = \sum_i \lambda_i y_i \boldsymbol{x}_i \ .$$

- The Lagrange dual problem is given the optimization problem:

$$\begin{array}{ll} \text{maximize} & -\frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j \cdot \boldsymbol{x}_i^T \boldsymbol{x}_j + \sum_i \lambda_i \\ \text{subject to} & \boldsymbol{\lambda} \succeq 0 \end{array}$$

Dual Representation

- The decision boundary of a support vector machine can be rewritten in dual form:

$$f(\boldsymbol{x}) = \boldsymbol{\alpha}^T \boldsymbol{x} + \alpha_0 = \sum_i \lambda_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + \alpha_0$$

  where we have used the identity:

$$\boldsymbol{\alpha} = \sum_i \lambda_i y_i \boldsymbol{x}_i \ .$$

- The Lagrange dual problem is given the optimization problem:

$$\begin{array}{ll} \text{maximize} & -\frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j \cdot \boldsymbol{x}_i^T \boldsymbol{x}_j + \sum_i \lambda_i \\ \text{subject to} & \boldsymbol{\lambda} \succeq 0 \end{array}$$

Conclusion: Feature vectors $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, and $\boldsymbol{x}$ only appear in inner products, both in learning and classification phase.

Inner Product and the Perceptron

The decision boundary that we get for the perceptron can also be written in terms of inner products:

$$F(\boldsymbol{x})$$

Inner Product and the Perceptron

The decision boundary that we get for the perceptron can also be written in terms of inner products:

$$F(\boldsymbol{x}) \quad = \quad \left(\sum_{i \in \mathcal{E}} y_i \cdot \boldsymbol{x}_i \right)^T \boldsymbol{x} + \sum_{i \in \mathcal{E}} y_i$$

Inner Product and the Perceptron

The decision boundary that we get for the perceptron can also be written in terms of inner products:

$$\begin{aligned} F(\boldsymbol{x}) &= \left(\sum_{i\in\mathcal{E}} y_i \cdot \boldsymbol{x}_i\right)^T \boldsymbol{x} + \sum_{i\in\mathcal{E}} y_i \\ &= \sum_{i\in\mathcal{E}} y_i \cdot \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + \sum_{i\in\mathcal{E}} y_i \end{aligned}$$

Feature Transforms

We select a feature transform $\phi : \mathbb{R}^d \to \mathbb{R}^D$ such that the resulting features $\phi(\boldsymbol{x}_i)$, $i = 1, 2, \ldots, m$ are linearly separable.

Feature Transforms

We select a feature transform $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$ such that the resulting features $\phi(\boldsymbol{x}_i)$, $i = 1, 2, \ldots, m$ are linearly separable.

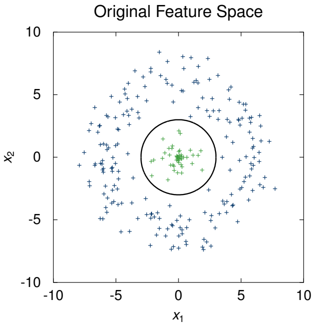


Fig.: Application of feature transform $\phi(\boldsymbol{x}_i) = \left(x_1^2, x_2^2\right)^T$.

Feature Transforms

We select a feature transform $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$ such that the resulting features $\phi(\boldsymbol{x}_i)$, $i = 1, 2, \ldots, m$ are linearly separable.

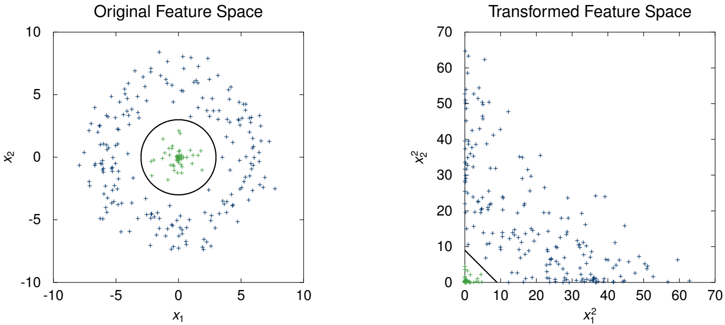


Fig.: Application of feature transform $\phi(\boldsymbol{x}_i) = \left(x_1^2, x_2^2\right)^T$.

Feature Transforms (cont.)

Example

Assume the decision boundary is given by the quadratic function

$$f(\boldsymbol{x}) = a_0 + a_1 x_1^2 + a_2 x_2^2 + a_3 x_1 x_2 + a_4 x_1 + a_5 x_2.$$

Obviously this is not a linear decision boundary.

Feature Transforms (cont.)

Example

Assume the decision boundary is given by the quadratic function

$$f(\boldsymbol{x}) = a_0 + a_1 x_1^2 + a_2 x_2^2 + a_3 x_1 x_2 + a_4 x_1 + a_5 x_2.$$

Obviously this is not a linear decision boundary.

By the following mapping, we get features that have a linear decision boundary:

$$\phi(\boldsymbol{x}) = \begin{pmatrix} 1 \\ x_1^2 \\ x_2^2 \\ x_1 x_2 \\ x_1 \\ x_2 \end{pmatrix}$$

Feature Transforms (cont.)

Consider distances in transformed feature space:

$$\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\|_2^2$$

Feature Transforms (cont.)

Consider distances in transformed feature space:

$$\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\|_2^2 \quad = \quad \langle \phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')), (\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\rangle$$

Feature Transforms (cont.)

Consider distances in transformed feature space:

$$
\begin{aligned}
\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\|_2^2 &= \langle \phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')), (\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\rangle \\
&= \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x})\rangle - 2\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}')\rangle + \langle \phi(\boldsymbol{x}'), \phi(\boldsymbol{x}')\rangle
\end{aligned}
$$

Feature Transforms (cont.)

Consider distances in transformed feature space:

$$\begin{aligned}\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\|_2^2 &= \langle\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')), (\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\rangle \\ &= \langle\phi(\boldsymbol{x}), \phi(\boldsymbol{x})\rangle - 2\langle\phi(\boldsymbol{x}), \phi(\boldsymbol{x}')\rangle + \langle\phi(\boldsymbol{x}'), \phi(\boldsymbol{x}')\rangle\end{aligned}$$

Conclusion: Distances can be computed by just evaluating inner products.

Feature Transforms (cont.)

These feature transforms can be easily incorporated into SVMs:

- Decision boundary:

$$f(\boldsymbol{x}) = \sum_i \lambda_i y_i \cdot \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle + \alpha_0$$

Feature Transforms (cont.)

These feature transforms can be easily incorporated into SVMs:

- Decision boundary:

$$f(\boldsymbol{x}) = \sum_i \lambda_i y_i \cdot \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle + \alpha_0$$

- The Lagrange dual problem is given the optimization problem:

$$\text{maximize} \qquad -\frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j \cdot \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle + \sum_i \lambda_i$$

$$\text{subject to} \qquad \boldsymbol{\lambda} \succeq 0$$

Kernel Functions

Definition

A *kernel function* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric function that maps a pair of features to a real number. For a kernel function the following property holds:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$$

for any feature mapping ϕ.

Kernel Functions

Definition

A *kernel function* $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a symmetric function that maps a pair of features to a real number. For a kernel function the following property holds:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$$

for any feature mapping ϕ.

Note:

Usually the evaluation of the kernel function is much easier than the computation of transformed features followed by the inner product.

Kernel Functions (cont.)

Definition

For a given set of feature vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m$, we define the *kernel matrix*

$$\boldsymbol{K} \quad = \quad [K_{i,j}]_{i,j=1,2,\ldots,m} \, , \quad \text{where} \quad K_{i,j} = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle .$$

Kernel Functions (cont.)

Definition

For a given set of feature vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m$, we define the *kernel matrix*

$$\boldsymbol{K} \quad = \quad [K_{i,j}]_{i,j=1,2,\ldots,m} \, , \quad \text{where} \quad K_{i,j} = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle .$$

Note:

The entries of the matrix are similarity measures for transformed feature pairs.

Kernel Functions (cont.)

Lemma

The kernel matrix is positive semidefinite.

Kernel Functions (cont.)

Lemma

The kernel matrix is positive semidefinite.

Proof: We need to show $\forall \boldsymbol{x} : \ \boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} \geq 0$:

Kernel Functions (cont.)

Lemma

The kernel matrix is positive semidefinite.

Proof: We need to show $\forall \boldsymbol{x} : \ \boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} \geq 0$:

$$\boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x}$$

Kernel Functions (cont.)

Lemma

The kernel matrix is positive semidefinite.

Proof: We need to show $\forall \boldsymbol{x} : \ \boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} \geq 0$:

$$\boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} = \sum_{i,j=1}^{m} x_i x_j K_{i,j}$$

Kernel Functions (cont.)

Lemma

The kernel matrix is positive semidefinite.

Proof: We need to show $\forall \boldsymbol{x} : \ \boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} \geq 0$:

$$\boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} = \sum_{i,j=1}^{m} x_i x_j K_{i,j} = \sum_{i,j=1}^{m} x_i x_j \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$$

Kernel Functions (cont.)

Lemma

The kernel matrix is positive semidefinite.

Proof: We need to show $\forall \boldsymbol{x} : \ \boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} \geq 0$:

$$
\begin{aligned}
\boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} &= \sum_{i,j=1}^{m} x_i x_j K_{i,j} = \sum_{i,j=1}^{m} x_i x_j \left\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \right\rangle \\
&= \sum_{i,j=1}^{m} \left\langle x_i \phi(\boldsymbol{x}_i), x_j \phi(\boldsymbol{x}_j) \right\rangle
\end{aligned}
$$

Kernel Functions (cont.)

Lemma

The kernel matrix is positive semidefinite.

Proof: We need to show $\forall \boldsymbol{x} : \ \boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} \geq 0$:

$$
\begin{aligned}
\boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} &= \sum_{i,j=1}^{m} x_i x_j K_{i,j} = \sum_{i,j=1}^{m} x_i x_j \left\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \right\rangle \\
&= \sum_{i,j=1}^{m} \left\langle x_i \phi(\boldsymbol{x}_i), x_j \phi(\boldsymbol{x}_j) \right\rangle \\
&= \left\langle \sum_{i=1}^{m} x_i \phi(\boldsymbol{x}_i), \sum_{j=1}^{m} x_j \phi(\boldsymbol{x}_j) \right\rangle
\end{aligned}
$$

Kernel Functions (cont.)

Lemma

The kernel matrix is positive semidefinite.

Proof: We need to show $\forall \boldsymbol{x} : \ \boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} \geq 0$:

$$\begin{aligned}
\boldsymbol{x}^T \boldsymbol{K} \boldsymbol{x} &= \sum_{i,j=1}^{m} x_i x_j K_{i,j} \quad = \quad \sum_{i,j=1}^{m} x_i x_j \left\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \right\rangle \\
&= \sum_{i,j=1}^{m} \left\langle x_i \phi(\boldsymbol{x}_i), x_j \phi(\boldsymbol{x}_j) \right\rangle \\
&= \left\langle \sum_{i=1}^{m} x_i \phi(\boldsymbol{x}_i), \sum_{j=1}^{m} x_j \phi(\boldsymbol{x}_j) \right\rangle \quad = \quad \left\| \sum_{i=1}^{m} x_i \phi(\boldsymbol{x}_i) \right\|_2^2 \quad \geq \quad 0
\end{aligned}$$

Kernel Functions (cont.)

Typical kernel functions:

Kernel Functions (cont.)

Typical kernel functions:

- Linear: $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$

Kernel Functions (cont.)

Typical kernel functions:

- Linear: $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$
- Polynomial: $k(\boldsymbol{x}, \boldsymbol{x}') = (\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1)^d$

Kernel Functions (cont.)

Typical kernel functions:

- Linear: $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$
- Polynomial: $k(\boldsymbol{x}, \boldsymbol{x}') = (\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1)^d$
- Laplacian radial basis function: $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_1}{\sigma^2}}$

Kernel Functions (cont.)

Typical kernel functions:

- Linear: $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$
- Polynomial: $k(\boldsymbol{x}, \boldsymbol{x}') = (\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1)^d$
- Laplacian radial basis function: $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_1}{\sigma^2}}$
- Gaussian radial basis function: $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{\sigma^2}}$

Kernel Functions (cont.)

Typical kernel functions:

- Linear: $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$
- Polynomial: $k(\boldsymbol{x}, \boldsymbol{x}') = (\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1)^d$
- Laplacian radial basis function: $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_1}{\sigma^2}}$
- Gaussian radial basis function: $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{\sigma^2}}$
- Sigmoid kernel: $k(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\alpha \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \beta)$

Kernel Functions (cont.)

Typical kernel functions:

- Linear: $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$
- Polynomial: $k(\boldsymbol{x}, \boldsymbol{x}') = (\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1)^d$
- Laplacian radial basis function: $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_1}{\sigma^2}}$
- Gaussian radial basis function: $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{\sigma^2}}$
- Sigmoid kernel: $k(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\alpha \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \beta)$

Question:

Can we compute for any kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ a feature mapping ϕ such that the kernel function can be written as an inner product?

Kernel Functions (cont.)

Theorem (Mercer's Theorem)

For any symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *that is square integrable on its domain and which satisfies*

$$\int_{\mathcal{X}\times\mathcal{X}} f(\boldsymbol{x})\, f(\boldsymbol{x}')\, k(\boldsymbol{x}, \boldsymbol{x}')\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{x}' \geq 0$$

for all square integrable functions f*, there exist transforms* $\phi_i : \mathcal{X} \to \mathbb{R}$ *and* $\lambda_i \geq 0$ *such that:*

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_i \lambda_i\, \phi_i(\boldsymbol{x})\, \phi_i(\boldsymbol{x}')$$

for all $\boldsymbol{x}$ *and* $\boldsymbol{x}'$.

Kernel Functions (cont.)

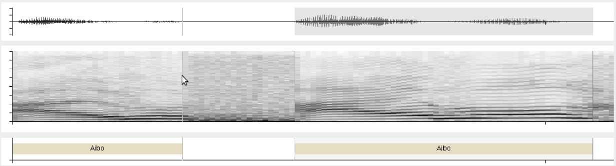
The Kernel Trick

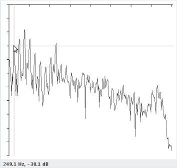
In *any* algorithm that is formulated in terms of a positive semidefinite kernel k, we can derive an alternative algorithm by replacing the kernel function k by another positive semidefinite kernel k'.

Kernels for Feature Sequences

Example (String Kernels)

- In speech recognition we do not have feature vectors but sequences of feature vectors.
- In order to use kernel methods we need a kernel for time series.





Kernels for Feature Sequences (cont.)

Example (String Kernels (cont.))

- Feature vectors are considered in $\mathbb{R}^d = \mathcal{X}$.
- Sequences of feature vectors are elements of $\mathcal{X}^*$.
- Problem: How to define a kernel over the sequence space $\mathcal{X}^*$?

Implications:

- PCA on feature sequences – COOL!
- SVM for feature sequences – EVEN COOLER!

Kernels for Feature Sequences (cont.)

Example (String Kernels (cont.))

Comparison of sequences via *dynamic time warping* (DTW):

Given the feature sequences ($p, q \in \{1, 2, \dots\}$):

$$\begin{aligned} \langle \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_p \rangle &\in \mathcal{X}^* \\ \langle \boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_q \rangle &\in \mathcal{X}^* \end{aligned}$$

Kernels for Feature Sequences (cont.)

Example (String Kernels (cont.))

- Distance is computed by DTW:

$$D(\langle \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p \rangle, \langle \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_q \rangle) = \frac{1}{p} \sum_{k=1}^{p} \| \boldsymbol{x}_{v(k)} - \boldsymbol{y}_{w(k)} \|_2$$

where v, w define the mapping of indices to indices.

Kernels for Feature Sequences (cont.)

Example (String Kernels (cont.))

- Distance is computed by DTW:

$$D(\langle \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p \rangle, \langle \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_q \rangle) = \frac{1}{p} \sum_{k=1}^{p} \| \boldsymbol{x}_{v(k)} - \boldsymbol{y}_{w(k)} \|_2$$

where v, w define the mapping of indices to indices.

- The DTW kernel can be defined as:

$$k(\boldsymbol{x}, \boldsymbol{y}) = e^{-D(\langle \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p \rangle, \langle \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_q \rangle)}$$

Kernels for Higher Order Features

- Nominal data
- Feature vectors where dimensions have semantic meaning

Kernels for Higher Order Features

- Nominal data
- Feature vectors where dimensions have semantic meaning

Example (Feature sequence for Speaker ID)

- Compute sequence of MFCC features for audio recording
- Estimate Gaussian mixture model (GMM) for speaker (using EM or adaptation)
- Use Bhattacharyya distance to compare two speakers

$$d_B(\omega_i, \omega_j) = \frac{1}{4}(\mathbf{m}_i - \mathbf{m}_j)^T \left[\frac{\Sigma_i + \Sigma_j}{2}\right]^{-1} (\mathbf{m}_i - \mathbf{m}_j) + log \left[\frac{\left|\frac{\Sigma_i + \Sigma_j}{2}\right|}{(|\Sigma_i| \; |\Sigma_j|)^{1/2}}\right]$$

Lessons Learned

- Limitations of linear decision boundaries
- Non-linear feature transforms
- Kernel function and kernel matrix
- Kernel trick
- Probabilities and kernels

Further Readings

- Bernhard Sch"olkopf, Alexander J. Smola: Learning with Kernels, The MIT Press, Cambridge, 2003.
- Vladimir N. Vapnik: The Nature of Statistical Learning Theory, Information Science and Statistics, Springer, Heidelberg, 2000.

Comprehensive Questions

- What are the properties of kernel functions?
- What is the kernel matrix?
- What is the kernel trick?
- How can we use kernels for string comparison?