Talk Announcement

Volker Blanz (University of Siegen) 11.05., 14:00h, 6317

3D Gesichtsanimation mit einem Morphable Model


Der Vortrag präsentiert die Grundlagen von Morphable Models, die Animation von Mund- und Augenbewegungen sowie die Simulation des Wachstums von Kindergesichtern.

Announcements

- Exercise 2 available on L2P
  - Risk
  - VC dimension
  - Linear classifiers
  - Fisher’s Linear Discriminant
  - SVMs
  ⇒ Submit your results until next Tuesday evening.

Course Outline

- Fundamentals (2 weeks)
  - Bayes Decision Theory
  - Probability Density Estimation
- Discriminative Approaches (5 weeks)
  - Linear Discriminant Functions
  - Statistical Learning Theory
  - Support Vector Machines
  - Boosting, Decision Trees
- Generative Models (5 weeks)
  - Bayesian Networks
  - Markov Random Fields
- Regression Problems (2 weeks)
  - Gaussian Processes

Recap: Generalization and Overfitting

- Goal: predict class labels of new observations
  - Train classification model on limited training set.
  - The further we optimize the model parameters, the more the training error will decrease.
  - However, at some point the test error will go up again.
  ⇒ Overfitting to the training set!

Recap: Risk

- Empirical risk
  - Measured on the training/validation set
    \[ R_{\text{emp}}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i; \alpha)) \]
- Actual risk = Expected risk
  - expectation of the error on all data.
    \[ R(\alpha) = \int L(y, f(x; \alpha)) dP_{X,Y}(x, y) \]
    \[ P_{X,Y}(x, y) \] is the probability distribution of \((x, y)\).
    It is fixed, but typically unknown.
    ⇒ In general, we can’t compute the actual risk directly!
Recap: Statistical Learning Theory

- Idea
  - Compute an upper bound on the actual risk based on the empirical risk
    \[ R(\alpha) \leq R_{\text{emp}}(\alpha) + \epsilon(N, p^*, h) \]
  - where
    \( N \): number of training examples
    \( p^* \): probability that the bound is correct
    \( h \): capacity of the learning machine ("VC-dimension")

Recap: VC Dimension

- Vapnik-Chervonenkis dimension
  - Measure for the capacity of a learning machine.

- Formal definition:
  - If a given set of \( f \) points can be labeled in all possible \( 2^f \) ways, and for each labeling, a member of the set \( \{f(\alpha)\} \) can be found which correctly assigns those labels, we say that the set of points is shattered by the set of functions.
  - The VC dimension for the set of functions \( \{f(\alpha)\} \) is defined as the maximum number of training points that can be shattered by \( \{f(\alpha)\} \).

Recap: Upper Bound on the Risk

- Important result (Vapnik 1979, 1995)
  - With probability \((1-\eta)\), the following bound holds
    \[ R(\alpha) \leq R_{\text{emp}}(\alpha) + \epsilon(N, p^*, h) \]
  - This bound is independent of \( P_{X,Y}(x,y) \! \!\! \)!
  - If we know \( h \) (the VC dimension), we can easily compute the risk bound
    \[ R(\alpha) \leq R_{\text{emp}}(\alpha) + \epsilon(N, p^*, h) \]

Recap: Structural Risk Minimization

- How can we implement Structural Risk Minimization?
  - \( R(\alpha) \leq R_{\text{emp}}(\alpha) + \epsilon(N, p^*, h) \)

- Classic approach
  - Keep \( \epsilon(N, p^*, h) \) constant and minimize \( R_{\text{emp}}(\alpha) \).

- Support Vector Machines (SVMs)
  - Keep \( R_{\text{emp}}(\alpha) \) constant and minimize \( \epsilon(N, p^*, h) \).
  - In fact: \( R_{\text{emp}}(\alpha) = 0 \) for separable data.
  - Control \( \epsilon(N, p^*, h) \) by adapting the VC dimension (controlling the “capacity” of the classifier).

Topics of This Lecture

- Linear Support Vector Machines (Recap)
  - Lagrangian (primal) formulation
  - Dual formulation
  - Discussion
- Linearly non-separable case
  - Soft-margin classification
  - Updated formulation
- Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer’s condition
  - Popular kernels
- Applications

Recap: Support Vector Machine (SVM)

- Basic idea
  - The SVM tries to find a classifier which maximizes the margin between pos. and neg. data points.
  - Up to now: consider linear classifiers
    \[ w^T x + b = 0 \]

- Formulation as a convex optimization problem
  - Find the hyperplane satisfying
    \[ \arg \min_{w^T} \frac{1}{2} \|w\|^2 \]
  - under the constraints
    \[ t_n (w^T x_n + b) \geq 1 \quad \forall n \]
  - based on training data points \( x_n \) and target values \( t_n \in \{-1, 1\} \).
Recap: SVM - Primal Formulation

- Lagrangian primal form
  \[ L_p = \frac{1}{2} ||w||^2 - \sum_{n=1}^{N} a_n \{ t_n (w^T x_n + b) - 1 \} \]
  \[ = \frac{1}{2} ||w||^2 - \sum_{n=1}^{N} a_n \{ t_n y_n - 1 \} \]
- The solution of \( L_p \) needs to fulfill the KKT conditions
  - Necessary and sufficient conditions
    \[ a_n \geq 0 \]
    \[ t_n y_n - 1 \geq 0 \]
    \[ a_n \{ t_n y_n - 1 \} = 0 \]
  - Graphical interpretation:
    - The support vectors are the points on the margin.
    - They define the margin and thus the hyperplane.
  - All other data points can be discarded!

Recap: SVM - Solution

- Solution for the hyperplane
  - Computed as a linear combination of the training examples
  \[ w = \sum_{n=1}^{N} a_n t_n x_n \]
  - Sparse solution: \( a_n \neq 0 \) only for some points, the support vectors
  - Only the SVs actually influence the decision boundary!
  - Compute \( b \) by averaging over all support vectors:
  \[ b = \frac{1}{N_S} \sum_{n \in S} \left( t_n - \sum_{m \in S} a_m t_m x_n^T x_m \right) \]

Recap: SVM - Dual Formulation

- Maximize
  \[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m x_n^T x_m \]
  under the conditions
  \[ a_n \geq 0 \quad \forall n \]
  \[ \sum_{n=1}^{N} a_n t_n = 0 \]
- Comparison
  - \( L_d \) is equivalent to the primal form \( L_p \), but only depends on \( a_n \).
  - \( L_d \) scales with \( O(D) \).
  - \( L_d \) scales with \( O(N^2) \) - in practice between \( O(N) \) and \( O(N^2) \).

Topics of This Lecture

- Linear Support Vector Machines (Recap)
  - Lagrangian (primal) formulation
  - Dual formulation
  - Discussion
- Linearly non-separable case
  - Soft-margin classification
  - Updated formulation
- Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The kernel trick
  - Mercer’s condition
  - Popular kernels
- Applications

So Far...

- Only looked at linearly separable case...
  - Current problem formulation has no solution if the data are not linearly separable!
  - Need to introduce some tolerance to outlier data points.
SVM - Non-Separable Data

- Non-separable data
  - I.e. the following inequalities cannot be satisfied for all data points
    \[ w^T x_n + b \geq 1 \quad \text{for} \quad t_n = +1 \]
    \[ w^T x_n + b \leq -1 \quad \text{for} \quad t_n = -1 \]
  - Instead use
    \[ w^T x_n + b \geq 1 - \xi_n \quad \text{for} \quad t_n = +1 \]
    \[ w^T x_n + b \leq -1 + \xi_n \quad \text{for} \quad t_n = -1 \]
  - with “slack variables” \( \xi_n \geq 0 \quad \forall n \)

- Separable data
  - Minimize
    \[ \frac{1}{2} \|w\|^2 \]
  - Non-separable data
    - Minimize
      \[ \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n \]

  Tradeoff parameter!

SVM - Soft-Margin Classification

- Slack variables
  - One slack variable \( \xi_n \geq 0 \) for each training data point.
- Interpretation
  - \( \xi_n = 0 \) for points that are on the correct side of the margin.
  - \( \xi_n = 1 - y_i(x_i) \) for all other points.
  - We do not have to set the slack variables ourselves!
    ⇒ They are jointly optimized together with \( w \).

Point on decision boundary: \( \xi_n = 1 \)
Misclassified point: \( \xi_n > 1 \)

SVM - New Primal Formulation

- New SVM Primal: Optimize
  \[ L_p = \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} a_n (t_n y_i(x_i) - 1) - \sum_{n=1}^{N} \mu_n \xi_n \]
  \[ \text{Constraint} \quad t_n y_i(x_i) \geq 1 - \xi_n \]
  \[ \xi_n \geq 0 \]
- KKT conditions
  \[ a_n \geq 0 \]
  \[ \mu_n \geq 0 \]
  \[ \xi_n \geq 0 \]
  \[ f(x) \geq 0 \]
  \[ a_n (t_n y_i(x_i) - 1 + \xi_n) = 0 \]
  \[ \mu_n \xi_n = 0 \]
  \[ A f(x) = 0 \]

SVM - New Dual Formulation

- New SVM Dual: Maximize
  \[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_n^T x_m) \]
  under the conditions
  \[ 0 \leq a_n \leq C \]
  \[ \sum_{n=1}^{N} a_n t_n = 0 \]

  This is all that changed!

- This is again a quadratic programming problem
  ⇒ Solve as before... (more on that later)

SVM - New Solution

- Solution for the hyperplane
  - Computed as a linear combination of the training examples
    \[ w = \sum_{n=1}^{N} a_n t_n x_n \]
  - Again sparse solution: \( a_n = 0 \) for points outside the margin.
  - The slack points with \( \xi_n > 0 \) are now also support vectors!
  - Compute \( b \) by averaging over all \( N_s \) points with \( 0 < a_n < C \):
    \[ b = \frac{1}{N_s} \sum_{n \in S} \left( t_n - \sum_{m \in M} a_m t_m x_n^T x_m \right) \]
Interpretation of Support Vectors

- Those are the hard examples!
  - We can visualize them, e.g., for face detection

Topics of This Lecture

- Linear Support Vector Machines (Recap)
  - Lagrangian (primal) formulation
  - Dual formulation
  - Discussion
- Linearly non-separable case
  - Soft-margin classification
  - Updated formulation
- Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer’s condition
  - Popular kernels
  - Applications

So Far...

- Only looked at linearly separable case...
  - Current problem formulation has no solution if the data are not linearly separable!
  - Need to introduce some tolerance to outlier data points
  - Slack variables.
- Only looked at linear decision boundaries...
  - This is not sufficient for many applications.
  - Want to generalize the ideas to non-linear boundaries.

Nonlinear SVM

- Linear SVMs
  - Datasets that are linearly separable with some noise work well:
  - But what are we going to do if the dataset is just too hard?
- How about... mapping data to a higher-dimensional space:

Another Example

- Non-separable by a hyperplane in 2D

Another Example

- Separable by a surface in 3D
Nonlinear SVM - Feature Spaces

- General idea: The original input space can be mapped to some higher-dimensional feature space where the training set is separable:

\[ \phi: \mathbb{R}^D \rightarrow \phi(\mathbb{R}^D) \]

\[ \phi(x) = \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} \]

\[ \phi(x) \] only appears in the form of dot products \( \phi(x)^T \phi(y) \):

\[ y(x) = w^T \phi(x) + b = \sum_{n=1}^{N} a_n t_n \phi(x_n)^T \phi(x) + b \]

Define a so-called kernel function \( k(x,y) = \phi(x)^T \phi(y) \).

Now, in place of the dot product, use the kernel instead:

\[ y(x) = \sum_{n=1}^{N} a_n t_n k(x_n, x) + b \]

The kernel function implicitly maps the data to the higher-dimensional space (without having to compute \( \phi(x) \) explicitly)!

Nonlinear SVM

- General idea

  - Nonlinear transformation \( \phi \) of the data points \( x \):

  \[ x \in \mathbb{R}^D \quad \phi: \mathbb{R}^D \rightarrow \mathcal{H} \]

  - Hyperplane in higher-dim. space \( \mathcal{H} \) (linear classifier in \( \mathcal{H} \))

  \[ w^T \phi(x) + b = 0 \]

  \( \Rightarrow \) Nonlinear classifier in \( \mathbb{R} \).

Problem with High-dim. Basis Functions

- Problem

  - Motivation: Easier to separate data in higher-dimensional space.

  - But wait - isn’t there a big problem?

  - How should we evaluate the decision function?

  - Oh-oh...

Solution: The Kernel Trick

- Important observation

  - \( \phi(x) \) only appears in the form of dot products \( \phi(x)^T \phi(y) \):

  \[ y(x) = w^T \phi(x) + b = \sum_{n=1}^{N} a_n t_n \phi(x_n)^T \phi(x) + b \]

- Define a so-called kernel function \( k(x,y) = \phi(x)^T \phi(y) \).

- Now, in place of the dot product, use the kernel instead:

  \[ y(x) = \sum_{n=1}^{N} a_n t_n k(x_n, x) + b \]

- The kernel function implicitly maps the data to the higher-dimensional space (without having to compute \( \phi(x) \) explicitly)!

Back to Our Previous Example...

- 2nd degree polynomial kernel:

  \[ \phi(x)^T \phi(y) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix} \begin{bmatrix} \sqrt{2}y_1 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{bmatrix} = x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 = (x^T y)^2 = k(x,y) \]

  - Whenever we evaluate the kernel function \( k(x,y) = (x^T y)^2 \), we implicitly compute the dot product in the higher-dimensional feature space.
SVMs with Kernels

- Using kernels
  - Applying the kernel trick is easy. Just replace every dot product by a kernel function...
  - \[ x^T y \rightarrow k(x, y) \]
  - ...and we're done.
  - Instead of the raw input space, we're now working in a higher-dimensional (potentially infinite dimensional) space, where the data is more easily separable.

  “Sounds like magic...”

- Wait - does this always work?
  - The kernel needs to define an implicit mapping to a higher-dimensional feature space \( \phi(x) \).
  - When is this the case?

Which Functions are Valid Kernels?

- Mercer’s theorem (modernized version):
  - Every positive definite symmetric function is a kernel.
- Positive definite symmetric functions correspond to a positive definite symmetric Gram matrix:

  \[
  K = \begin{bmatrix}
  k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_N) \\
  k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_N) \\
  \vdots & \vdots & \ddots & \vdots \\
  k(x_N, x_1) & k(x_N, x_2) & \cdots & k(x_N, x_N) 
  \end{bmatrix}
  \]

Kernels Fulfilling Mercer's Condition

- Polynomial kernel
  \[ k(x, y) = (x^T y + 1)^p \]
- Radial Basis Function kernel
  \[ k(x, y) = \exp\left( -\frac{(x - y)^2}{2\sigma^2} \right) \]
  e.g. Gaussian
- Hyperbolic tangent kernel
  \[ k(x, y) = \tanh(k^T x + \delta) \]
  e.g. Sigmoid

  (and many, many more...)

Nonlinear SVM - Dual Formulation

- SVM Dual: Maximize
  \[ L_D(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} a_n a_m t_m t_n k(x_m, x_n) \]

  under the conditions
  \[ 0 \leq a_n \leq C \]

  \[ \sum_{n=1}^{N} a_n t_n = 0 \]

- Classify new data points using
  \[ y(x) = \sum_{n=1}^{N} a_n t_n k(x_n, x) + b \]

VC Dimension for Polynomial Kernel

- Polynomial kernel of degree \( p \):
  \[ k(x, y) = (x^T y)^p \]
  - Dimensionality of \( \mathcal{H} \):
    \[ D + p - 1 \]
    \[ p \]
    \[ \dim(\mathcal{H}) = 183.181.376 \]
  - Example:
    \[ D = 16 \times 16 = 256 \]
    \[ p = 4 \]
    \[ \dim(\mathcal{H}) = 183.181.376 \]
  - The hyperplane in \( \mathcal{H} \) then has VC-dimension
    \[ \dim(\mathcal{H}) + 1 \]

VC Dimension for Gaussian RBF Kernel

- Radial Basis Function:
  \[ k(x, y) = \exp\left( -\frac{(x - y)^2}{2\sigma^2} \right) \]
  - In this case, \( \mathcal{H} \) is infinite dimensional!
  - \[ \exp(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \ldots \]
  - Since only the kernel function is used by the SVM, this is no problem.
  - The hyperplane in \( \mathcal{H} \) then has VC-dimension
    \[ \dim(\mathcal{H}) + 1 = \infty \]
VC Dimension for Gaussian RBF Kernel

- Intuitively
  - If we make the radius of the RBF kernel sufficiently small, then each data point can be associated with its own kernel.
  - However, this also means that we can get finite VC-dimension if we set a lower limit to the RBF radius.

Example: RBF Kernels

- Decision boundary on toy problem

But... but... but...

- Don’t we risk overfitting with those enormously high-dimensional feature spaces?
  - No matter what the basis functions are, there are really only up to \( N \) parameters: \( \alpha_1, \alpha_2, \ldots, \alpha_N \) and most of them are usually set to zero by the maximum margin criterion.
  - The data effectively lives in a low-dimensional subspace of \( \mathbb{H} \).

- What about the VC dimension? I thought low VC-dim was good (in the sense of the risk bound)?
  - Yes, but the maximum margin classifier “magically” solves this.
  - Reason (Vapnik): by maximizing the margin, we can reduce the VC-dimension.
  - Empirically, SVMs have very good generalization performance.

Theoretical Justification for Maximum Margins

- Vapnik has proved the following:
  - The class of optimal linear separators has VC dimension \( h \) bounded from above as
    \[
    h \leq \min \left\lceil \frac{D^2}{\sigma^2} m_0 \right\rceil + 1
    \]
  - where \( \sigma \) is the margin, \( D \) is the diameter of the smallest sphere that can enclose all of the training examples, and \( m_0 \) is the dimensionality.
  - Intuitively, this implies that regardless of dimensionality \( m_0 \) we can minimize the VC dimension by maximizing the margin \( \sigma \).
  - Thus, complexity of the classifier is kept small regardless of dimensionality.

Summary: SVMs

- Properties
  - Empirically, SVMs work very, very well.
  - SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.
  - SVMs can be applied to complex data types beyond feature vectors (e.g., graphs, sequences, relational data) by designing kernel functions for such data.
  - SVM techniques have been applied to a variety of other tasks – e.g. SV Regression, One-class SVMs, ...
  - The kernel trick has been used for a wide variety of applications. It can be applied wherever dot products are in use – e.g., Kernel PCA, kernel FLD, ...
  - Good overview, software, and tutorials available on http://www.kernel-machines.org/
Summary: SVMs

- Limitations
  - How to select the right kernel?
    - Still something of a black art...
  - How to select the kernel parameters?
    - (Massive) cross-validation.
    - Usually, several parameters are optimized together in a grid search.
  - Solving the quadratic programming problem
    - Standard QP solvers do not perform too well on SVM task.
    - Dedicated methods have been developed for this, e.g. SMO.
  - Speed of evaluation
    - Evaluating \( y(x) \) scales linearly in the number of SVs.
    - Too expensive if we have a large number of support vectors.
    - ⇒ There are techniques to reduce the effective SV set.
  - Training for very large datasets (millions of data points)
    - Still problematic...

- How to select the right kernel?
  - Still something of a black art...

- How to select the kernel parameters?
  - (Massive) cross-validation.
  - Usually, several parameters are optimized together in a grid search.
  - Standard QP solvers do not perform too well on SVM task.
  - Dedicated methods have been developed for this, e.g. SMO.
  - Speed of evaluation
    - Evaluating \( y(x) \) scales linearly in the number of SVs.
    - Too expensive if we have a large number of support vectors.
    - ⇒ There are techniques to reduce the effective SV set.
  - Training for very large datasets (millions of data points)
    - Still problematic...

Topics of This Lecture

- Linear Support Vector Machines (Recap)
  - Lagrangian (primal) formulation
  - Dual formulation
  - Discussion

- Linearly non-separable case
  - Soft-margin classification
  - Updated formulation

- Nonlinear Support Vector Machines
  - Nonlinear basis functions
    - The kernel trick
  - Averon’s condition
  - Popular kernels

- Applications

Example Application: Text Classification

- Problem:
  - Classify a document in a number of categories

- Representation:
  - “Bag-of-words” approach
  - Histogram of word counts (on learned dictionary)
  - Very high-dimensional feature space (~10,000 dimensions)
  - Few irrelevant features

- This was one of the first applications of SVMs
  - T. Joachims (1997)

- Results:

Example Application: OCR

- Handwritten digit recognition
  - US Postal Service Database
  - Standard benchmark task for many learning algorithms
Example Application: OCR

- Results
  - Almost no overfitting with higher-degree kernels.

<table>
<thead>
<tr>
<th>degree of polynomial</th>
<th>dimensionality of feature space</th>
<th>support vectors</th>
<th>raw error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256</td>
<td>282</td>
<td>8.9</td>
</tr>
<tr>
<td>2</td>
<td>≈ 3,000</td>
<td>217</td>
<td>4.7</td>
</tr>
<tr>
<td>3</td>
<td>≈ 1 x 10^6</td>
<td>274</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>≈ 1 x 10^9</td>
<td>321</td>
<td>4.2</td>
</tr>
<tr>
<td>5</td>
<td>≈ 1 x 10^12</td>
<td>374</td>
<td>4.3</td>
</tr>
<tr>
<td>6</td>
<td>≈ 1 x 10^14</td>
<td>377</td>
<td>4.5</td>
</tr>
<tr>
<td>7</td>
<td>≈ 1 x 10^16</td>
<td>422</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Example Application: Pedestrian Detection

- Sliding-window approach
  - E.g. histogram representation (HOG)
    - Map each grid cell in the input window to a histogram of gradient orientations.
    - Train a linear SVM using training set of pedestrian vs. non-pedestrian windows.

Many Other Applications

- Lots of other applications in all fields of technology
  - OCR
  - Text classification
  - Computer vision
  - ...
  - High-energy physics
  - Monitoring of household appliances
  - Protein secondary structure prediction
  - Design on decision feedback equalizers (DFE) in telephony

(Detailed references in Schölkopf & Smola, 2002, pp. 221)

You Can Try It At Home...

- Lots of SVM software available, e.g.
  - svmlight (http://svmlight.joachims.org/)
    - Command-line based interface
    - Source code available (in C)
    - Interfaces to Python, MATLAB, Perl, Java, DLL,...
  - libsvm (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
    - Library for inclusion with own code
    - C++ and Java sources
    - Interfaces to Python, R, MATLAB, Perl, Ruby, Weka, C++,NET,...
  - Both include fast training and evaluation algorithms, support for multi-class SVMs, automated training and cross-validation, ...
  - Easy to apply to your own problems!

References and Further Reading

- More information on SVMs can be found in Chapter 7.1 of Bishop’s book. You can also look at Schölkopf & Smola (some chapters available online).
  - A more in-depth introduction to SVMs is available in the following tutorial:
    - B. Schölkopf, A. Smola Learning with Kernels MIT Press, 2002
    - S. Boyd, S. Vandenbeuc, and A. C. Parker, Convex Optimization: A Tutorial
      - Available online: http://www.convexoptimization.com/