Machine Learning – Lecture 9

Support Vector Machines II

07.11.2019

Bastian Leibe
RWTH Aachen
leibe@vision.rwth-aachen.de

Topics of This Lecture

• Recap: Support Vector Machines
  - Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
• Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer’s condition
  - Popular kernels
• Analysis
  - Error function
• 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
    \[ \arg \min_{w,b} \frac{1}{2} \|w\|^2 \]
    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 – Lagrangian Formulation

• Find hyperplane minimizing \( \|w\|^2 \) under the constraints
  \[ t_n (w^T x_n + b) - 1 \geq 0 \quad \forall n \]

• Lagrangian formulation
  - Introduce positive Lagrange multipliers: \( a_n \geq 0 \quad \forall n \)
  - Minimize Lagrangian ("primal form")
    \[ L(w, b, a) = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left( t_n (w^T x_n + b) - 1 \right) \]
  - I.e., find \( w, b, \) and \( a \) such that
    \[ \frac{\partial L}{\partial w} = 0 \Rightarrow \sum_{n=1}^{N} a_n t_n x_n = 0 \]
    \[ \frac{\partial L}{\partial b} = 0 \Rightarrow w = \sum_{n=1}^{N} a_n t_n x_n \]

Recap: SVM – Primal Formulation

• Lagrangian primal form
  \[ L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left( t_n (w^T x_n + b) - 1 \right) \]
  \[ = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left( t_n y(x_n) - 1 \right) \]
  • The solution of \( L_p \) needs to fulfill the KKT conditions
    - Necessary and sufficient conditions
      \[ a_n \geq 0 \quad \forall n \]
      \[ t_n y(x_n) - 1 \geq 0 \]
      \[ f(x) \geq 0 \]
      \[ \lambda f(x) = 0 \]

Course Outline

• Fundamentals
  - Bayes Decision Theory
  - Probability Density Estimation
• Classification Approaches
  - Linear Discriminants
  - Support Vector Machines
  - Ensemble Methods & Boosting
  - Randomized Trees, Forests & Ferns
• Deep Learning
  - Foundations
  - Convolutional Neural Networks
  - Recurrent Neural Networks
Recap: SVM for Non-Separable Data

- 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.
  - $\xi_n = y_n - y(x_n)$ for all other points.
- We do not have to set the slack variables ourselves!
  $\Rightarrow$ They are jointly optimized together with $w$.

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

Recap: SVM – New Dual Formulation

- New SVM Dual: Maximize
  $$L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_na_mt_n(x_n^T x_m)$$
  under the conditions
  $$\begin{align*}
  0 \cdot a_n, C \\
  \sum_{n=1}^{N} a_n = 0
  \end{align*}$$

- This is again a quadratic programming problem
  $\Rightarrow$ Solve as before...

Topics of This Lecture

- Recap: Support Vector Machines
  - Recap: Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
- Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer’s condition
  - Popular kernels
- Analysis
  - Error function
- Applications

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:

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:
Nonlinear SVM

- General idea
  - Nonlinear transformation $\phi$ of the data points $x_i$:
    $$ x \in \mathbb{R}^D \rightarrow \phi : \mathbb{R}^D \rightarrow \mathcal{H} $$
  - Hyperplane in higher-dim. space $\mathcal{H}$ (linear classifier in $\mathcal{H}$)
    $$ w^T \phi(x) + b = 0 $$
  - Nonlinear classifier in $\mathbb{R}^D$.

What Could This Look Like?

- Example:
  - Mapping to polynomial space, $x, y \in \mathbb{R}^2$.
    $$ \phi(x) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix} $$

Problem with High-dim. Basis Functions

- Problem
  - In order to apply the SVM, we need to evaluate the function
    $$ y(x) = w^T \phi(x) + b $$
  - Using the hyperplane, which is itself defined as
    $$ w = \sum_{n=1}^{N} a_n \phi(x_n) $$

Solution: The Kernel Trick

- Important observation
  - $\phi(x)$ only appears in the form of dot products $\phi(x)^T \phi(y)$:
    $$ y(x) = w^T \phi(x) + b $$
    $$ = \sum_{n=1}^{N} a_n \phi(x_n)^T \phi(x) + b $$
  - Trick: 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:
    $$ y(x) = \sum_{n=1}^{N} a_n k(x_n, x) + b $$
  - 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:
  $$ \phi(x)^T \phi(y) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix} \begin{bmatrix} y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{bmatrix} $$
  $$ = x_1^2 y_1^2 + 2x_1x_2y_1y_2 + x_2^2 y_2^2 $$
  $$ = (x^T y)^2 =: k(x, y) $$
  - 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:

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

(positive definite \(\Rightarrow\) all eigenvalues are \(> 0\))

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
  
  \[
  k(x, y) = \tanh(\alpha x^T y + \beta)
  \]
  e.g. Sigmoid

(and many, many more…)

Example: Bag of Visual Words Representation

- General framework in visual recognition
  - Create a codebook (vocabulary) of prototypical image features
  - Represent images as histograms over codebook activations
  - Compare two images by any histogram kernel, e.g. \(\chi^2\) kernel

\[
k_{\chi^2}(h, h’) = \exp\left(-\frac{1}{2} \sum\frac{(h_i - h’_i)^2}{h_i + h’_i}\right)
\]

Nonlinear SVM – Dual Formulation

- SVM Dual: Maximize

\[
L_d(a) = \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j t_i t_j k(x_i, x_j)
\]

under the conditions

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

Classify new data points using

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

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](http://www.kernel-machines.org)
Summary: SVMs

- **Limitations**
  - How to select the right kernel?
    - Best practice guidelines are available for many applications.
  - 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.
  - Training for very large datasets (millions of data points)
    - Stochastic gradient descent and other approximations can be used.

Topics of This Lecture

- Recap: Support Vector Machines
  - Recap: Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
- Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer’s condition
  - Popular kernels
- Analysis
  - Error function
  - Applications

Recap: Error Functions

$t_n \in \{-1, 1\}$

Ideal misclassification error

$f_n \in \{-1, 1\}$

Squared error

Penalizes “too correct” data points!

Squares error used in Least-Squares Classification

- Very popular, leads to closed-form solutions.
- However, sensitive to outliers due to squared penalty.
- Penalizes “too correct” data points
  - Generally does not lead to good classifiers.

Error Functions (Loss Functions)

- “Hinge error” used in SVMs
  - Zero error for points outside the margin ($z_n > 1$) ⇒ sparsity
  - Linear penalty for misclassified points ($z_n < 1$) ⇒ robustness
  - Not differentiable around $z_n = 1$ ⇒ Harder to optimize directly.
SVM – Discussion

- SVM optimization function
  \[ \min_{\mathbf{w} \in \mathbb{R}^D} \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{n=1}^{N} \left[ 1 - t_n y_n (\mathbf{w}^T \mathbf{x}_n) \right]_+ \]
  - L_2 regularizer
  - Hinge loss

- Hinge loss enforces sparsity
  - Only a subset of training data points actually influences the decision boundary.
  - This is different from sparsity obtained through the regularizer! There, only a subset of input dimensions are used.
  - Unconstrained optimization, but non-differentiable function.
  - Solve, e.g., by subgradient descent
  - Currently most efficient: stochastic gradient descent

Topics of This Lecture

- Recap: Support Vector Machines
  - Recap: Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
- Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - The Kernel trick
  - Mercer’s condition
  - Popular kernels
- Analysis
  - Error function
- 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:

<table>
<thead>
<tr>
<th>Class</th>
<th>Bays/Rocchio</th>
<th>Chris-KNN</th>
<th>SVM (poly)</th>
<th>SVM (rbf)</th>
</tr>
</thead>
<tbody>
<tr>
<td>barn</td>
<td>95.9</td>
<td>96.1</td>
<td>97.3</td>
<td>98.5</td>
</tr>
<tr>
<td>cat</td>
<td>91.5</td>
<td>92.8</td>
<td>93.3</td>
<td>93.4</td>
</tr>
<tr>
<td>money</td>
<td>62.9</td>
<td>67.5</td>
<td>78.2</td>
<td>76.9</td>
</tr>
<tr>
<td>grain</td>
<td>72.5</td>
<td>79.5</td>
<td>82.2</td>
<td>81.9</td>
</tr>
<tr>
<td>crude</td>
<td>81.0</td>
<td>81.3</td>
<td>82.7</td>
<td>82.6</td>
</tr>
<tr>
<td>trade</td>
<td>50.0</td>
<td>77.1</td>
<td>77.4</td>
<td>77.7</td>
</tr>
<tr>
<td>interest</td>
<td>58.0</td>
<td>72.5</td>
<td>74.0</td>
<td>74.9</td>
</tr>
<tr>
<td>ship</td>
<td>78.7</td>
<td>83.1</td>
<td>89.9</td>
<td>90.5</td>
</tr>
<tr>
<td>wheat</td>
<td>60.5</td>
<td>79.4</td>
<td>76.6</td>
<td>76.3</td>
</tr>
<tr>
<td>corn</td>
<td>47.3</td>
<td>62.2</td>
<td>77.9</td>
<td>80.6</td>
</tr>
<tr>
<td>microavg</td>
<td>72.0</td>
<td>79.0</td>
<td>82.3</td>
<td>82.2</td>
</tr>
</tbody>
</table>

This was one of the first applications of SVMs

- T. Joachims (1997)

Example Application: OCR

- Handwritten digit recognition
  - US Postal Service Database
  - Standard benchmark task for many learning algorithms
Historical Importance

- **USPS benchmark**
  - 2.5% error: human performance

- **Different learning algorithms**
  - 16.2% error: Decision tree (C4.5)
  - 5.9% error: (best) 2-layer Neural Network
  - 5.1% error: LeNet 1 – (massively hand-tuned) 5-layer network

- **Different SVMs**
  - 4.0% error: Polynomial kernel \( p=3 \), 274 support vectors
  - 4.1% error: Gaussian kernel \( \gamma=0.3 \), 291 support vectors

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>( \approx 3,000 )</td>
<td>274</td>
<td>4.7</td>
</tr>
<tr>
<td>3</td>
<td>( \approx 1 \times 10^9 )</td>
<td>321</td>
<td>4.2</td>
</tr>
<tr>
<td>4</td>
<td>( \approx 1 \times 10^{12} )</td>
<td>374</td>
<td>4.3</td>
</tr>
<tr>
<td>5</td>
<td>( \approx 1 \times 10^{14} )</td>
<td>377</td>
<td>4.5</td>
</tr>
<tr>
<td>6</td>
<td>( \approx 1 \times 10^{16} )</td>
<td>422</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Example Application: Object 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.

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 Schoelkopf & Smola, 2002, pp. 221)

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