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

Topics of This Lecture

- 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: 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
    $$\frac{\partial L}{\partial w} = 0 \Rightarrow \sum_{n=1}^{N} a_n t_n x_n = 0 \quad \frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} a_n t_n = 0$$
Recap: SVM – Primal Formulation

• Lagrangian primal form

\[ L_\nu = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \alpha_n \{ t_n(w^T x_n + b) - 1 \} \]

\[ = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \alpha_n \{ t_n y(x_n) - 1 \} \]

• The solution of \( L_\nu \) needs to fulfill the KKT conditions
  - Necessary and sufficient conditions
    - \( \alpha_n \geq 0 \)
    - \( t_n y(x_n) - 1 \geq 0 \)
    - \( f(x) \geq 0 \)
    - \( \alpha_n \{ t_n y(x_n) - 1 \} = 0 \)

KKT: \( f(x) = 0 \)

SVM – Support Vectors

• The training points for which \( \alpha_n > 0 \) are called “support vectors”.

• Graphical interpretation:
  - The support vectors are the points on the margin.
  - They define the margin and thus the hyperplane.

⇒ Robustness to “too correct” points!

SVM – Solution (Part 1)

• Solution for the hyperplane
  - Computed as a linear combination of the training examples

\[ w = \sum_{n=1}^{N} \alpha_n t_n x_n \]

• Because of the KKT conditions, the following must also hold

\[ \alpha_n \left( t_n (w^T x_n + b) - 1 \right) = 0 \]

⇒ Only some of the data points actually influence the decision boundary!

SVM – Discussion (Part 1)

• Linear SVM
  - Linear classifier
  - SVMs have a “guaranteed” generalization capability.
  - Formulation as convex optimization problem.
  ⇒ Globally optimal solution!

• Primal form formulation
  - Solution to quadratic prog. problem in \( M \) variables is in \( \mathcal{O}(M^3) \).
  - Here: \( D \) variables ⇒ \( \mathcal{O}(D^3) \).
  - Problem: scaling with high-dim. data (“curse of dimensionality”)

SVM – Solution (Part 2)

• Solution for the hyperplane
  - To define the decision boundary, we still need to know \( b \).
  - Observation: any support vector \( x_n \) satisfies

\[ t_n y(x_n) = \sum_{m \in S} a_m t_m x_m^T x_n + b \]

⇒ Using \( t_n^2 = 1 \) we can derive:

\[ b = t_n - \sum_{m \in \mathcal{S}} a_m t_m x_m^T x_n \]

• In practice, it is more robust to average over all support vectors:

\[ b = \frac{1}{|\mathcal{S}|} \sum_{n \in \mathcal{S}} \left( t_n - \sum_{m \in \mathcal{S}} a_m t_m x_m^T x_n \right) \]

SVM – Dual Formulation

• Improving the scaling behavior: rewrite \( L_\nu \) in a dual form

\[ L_\nu = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \alpha_n \{ t_n(w^T x_n + b) - 1 \} \]

\[ = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \alpha_n t_n w^T x_n - b \sum_{n=1}^{N} \alpha_n + \sum_{n=1}^{N} \alpha_n \]

⇒ Using the constraint \( \sum_{n=1}^{N} \alpha_n t_n = 0 \) we obtain

\[ \frac{\partial L_\nu}{\partial \alpha_n} = 0 \]

\[ L_\nu = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \alpha_n t_n w^T x_n + \sum_{n=1}^{N} \alpha_n \]
Dual Formulation

\[ L_p = \frac{1}{2} \| w \|^2 - \sum_{n=1}^{N} a_n t_n w^T x_n + \sum_{n=1}^{N} a_n \]

- Using the constraint \( w = \sum_{n=1}^{N} a_n t_n x_n \), we obtain

\[ \frac{\partial L_p}{\partial w} = 0 \]

\[ L_p = \frac{1}{2} \| w \|^2 - \sum_{n=1}^{N} a_n t_n + \sum_{n=1}^{N} a_n \]

\[ = \frac{1}{2} \| w \|^2 - \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (x_n^T x_m) + \sum_{n=1}^{N} a_n \]

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

1. \( a_n \geq 0 \) \( \forall n \)
2. \( \sum_{n=1}^{N} a_n t_n = 0 \)

- The hyperplane is given by the \( N_d \) support vectors:

\[ w = \sum_{n=1}^{N_d} a_n t_n x_n \]

**SVM – Discussion (Part 2)**

- Dual form formulation
  - In going to the dual, we now have a problem in \( N \) variables \( (a_n) \).
  - Isn’t this worse??? We penalize large training sets!

- However...
  1. SVMs have sparse solutions: \( a_n \neq 0 \) only for support vectors!
  2. We have avoided the dependency on the dimensionality.

- However…
  1. SVMs have sparse solutions: \( a_n \neq 0 \) only for support vectors!
  2. We have avoided the dependency on the dimensionality.

**SVM – Non-Separable Data**

- Non-separable data
  - I.e. the following inequalities cannot be satisfied for all data points

\[ w^T x_n + b \geq +1 \quad \text{for } t_n = +1 \]
\[ w^T x_n + b \leq -1 \quad \text{for } t_n = -1 \]

- Instead use

\[ w^T x_n + b \geq +1 - \xi_n \quad \text{for } t_n = +1 \]
\[ w^T x_n + b \leq -1 + \xi_n \quad \text{for } t_n = -1 \]

with "slack variables"

\[ \xi_n \geq 0 \quad \forall n \]
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.
  - \( \xi_n = |y_n - y(x_n)| \) for all other points (linear penalty).
- We do not have to set the slack variables ourselves! They are jointly optimized together with \( w \).

SVM – Non-Separable Data

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

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

  This is all that changed!

  • This is again a quadratic programming problem
  \( \Rightarrow \) 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.
  \( \Rightarrow \) The slack points with \( \xi_n > 0 \) are now also support vectors!
- Compute \( b \) by averaging over all \( N_M \) points with \( 0 < a_n < C \):
  \[ b = \frac{1}{N_M} \sum_{m \in M} \left( t_n - \sum_{m \in M} a_n t_m x_m^T x_n \right) \]

Interpretation of Support Vectors

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

• 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

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:

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:

What Could This Look Like?

• Example:
  - Mapping to polynomial space, $x, y \in \mathbb{R}^2$:

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

    Motivation: Easier to separate data in higher-dimensional space.
    But wait – isn’t there a big problem?
    ⇒ How should we evaluate the decision function?
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
    \[ w = \sum_{n=1}^{N} a_n \phi(x_n) \]
  - What happens if we try this for a million-dimensional feature space \( \phi(x) \)?
    - Oh-oh...

Solution: The Kernel Trick

- Important observation
  - \( \phi(x) \) only appears in the form of dot products \( \phi(x)^T \phi(y) \):
    \[ y(x) = w^T \phi(x) + b = \sum_{n=1}^{N} a_n \phi(x_n)^T \phi(x) + b \]

- 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:
    \[ y(x) = \sum_{n=1}^{N} a_n \phi(x_n) T \phi(x) + b \]
  - The kernel function implicitly maps the data to the higher-dimensional space (without having to compute \( \phi(x) \) explicitly)!

SVMs with Kernels

- Using kernels
  - Applying the kernel trick is easy. Just replace every dot product by a kernel function...
    - ...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 = 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
  \[ k(x, y) = \tanh(x^T y + \delta) \]
  e.g. Sigmoid

(…and many, many more…)

Slide credit: Raymond Mooney
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

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

Nonlinear SVM – Dual Formulation

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

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

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/)

- 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.
    - There are techniques to reduce the effective SV set.
  - 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
  - Lagrangian (primal) formulation
  - Dual formulation
  - Soft-margin classification
- Nonlinear Support Vector Machines
  - Nonlinear basis functions
  - Mercer’s condition
  - Popular kernels
- Analysis
  - Error function
- Applications
SVM – Analysis

- Traditional soft-margin formulation
  \[
  \min_{w \in \mathbb{R}^p, \xi_n \in \mathbb{R}^+} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n \quad \text{“Maximize the margin”}
  \]
  subject to the constraints
  \[
  t_n y_n(x_n) \geq 1 - \xi_n \quad \text{“Most points should be on the correct side of the margin”}
  \]
- Different way of looking at it
  - We can reformulate the constraints into the objective function.
    \[
    \min_{w \in \mathbb{R}^p} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} [1 - t_n y_n(x_n)]^+
    \]
  - \(L_2\) regularizer
  - “Hinge loss”
  where \(\lceil x \rceil := \max(0,x)\).

Recap: Error Functions

- \(t_n \in \{-1, 1\}\)
- \(E(z_n)\) Ideal misclassification error
  - Squared 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
      \(\Rightarrow\) Generally does not lead to good classifiers.
  - Hinge error
    - Zero error for points outside the margin \((z_n > 1)\) \(\Rightarrow\) sparsity
    - Linear penalty for misclassified points \((z_n < 1)\) \(\Rightarrow\) robustness
    - Not differentiable around \(z_n = 1\) \(\Rightarrow\) Cannot be optimized directly.

SVM – Discussion

- SVM optimization function
  \[
  \min_{w \in \mathbb{R}^p} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} [1 - t_n y_n(x_n)]^+
  \]
  - \(L_2\) regularizer
  - Hinge loss
  \(\Rightarrow\) sparsity
  \(\Rightarrow\) unconstrained optimization
  \(\Rightarrow\) subgradient descent
  \(\Rightarrow\) stochastic gradient descent

Recap: Error Functions

- \(t_n \in \{-1, 1\}\)
- \(E(z_n)\) Ideal misclassification error
  - Not differentiable!

Error Functions (Loss Functions)

- “Hinge error” used in SVMs
  - Zero error for points outside the margin \((z_n > 1)\) \(\Rightarrow\) sparsity
  - Linear penalty for misclassified points \((z_n < 1)\) \(\Rightarrow\) robustness
  - Not differentiable around \(z_n = 1\) \(\Rightarrow\) Cannot be optimized directly.

Topics of This Lecture

- 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)

Example Application: Text Classification

- Results:

<table>
<thead>
<tr>
<th></th>
<th>Bayes</th>
<th>Rocchio</th>
<th>C4.5</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SVM (poly)</td>
<td>SVM (RBF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>error</td>
<td>degree</td>
<td>width</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>g=0.6</td>
<td>0.08</td>
<td>0.12</td>
</tr>
<tr>
<td>train</td>
<td>95.9</td>
<td>95.8</td>
<td>95.9</td>
<td>95.9</td>
</tr>
<tr>
<td>test</td>
<td>95.3</td>
<td>95.5</td>
<td>95.8</td>
<td>95.7</td>
</tr>
<tr>
<td>money</td>
<td>91.0</td>
<td>92.1</td>
<td>92.0</td>
<td>92.2</td>
</tr>
<tr>
<td>trade</td>
<td>81.0</td>
<td>81.5</td>
<td>86.7</td>
<td>86.8</td>
</tr>
<tr>
<td>crude</td>
<td>50.9</td>
<td>69.5</td>
<td>76.6</td>
<td>77.1</td>
</tr>
<tr>
<td>internet</td>
<td>78.7</td>
<td>83.1</td>
<td>80.9</td>
<td>79.2</td>
</tr>
<tr>
<td>wheat</td>
<td>80.0</td>
<td>79.4</td>
<td>76.5</td>
<td>86.4</td>
</tr>
<tr>
<td>mean</td>
<td>73.0</td>
<td>79.0</td>
<td>79.4</td>
<td>82.3</td>
</tr>
<tr>
<td>micro</td>
<td>73.0</td>
<td>79.0</td>
<td>79.4</td>
<td>82.3</td>
</tr>
<tr>
<td></td>
<td>84.2</td>
<td>85.1</td>
<td>86.7</td>
<td>85.9</td>
</tr>
<tr>
<td></td>
<td>88.0</td>
<td>88.4</td>
<td>88.6</td>
<td>86.4</td>
</tr>
<tr>
<td></td>
<td>86.4</td>
<td>86.5</td>
<td>86.3</td>
<td>86.2</td>
</tr>
</tbody>
</table>

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 (n=0.3, 291 support vectors)
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

N. Dalal, B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

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:

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006

B. Schölkopf, A. Smola
Learning with Kernels
MIT Press, 2002
http://www.learning-with-kernels.org/