Announcements

- Exercise 2 available on L2P
  - Linear classifiers
    - Least-squares classification
  - Risk
  - VC dimension
  - SVMs
    - Solve the Quadratic Programming problem in Matlab
    - Application: USPS digit classification
    - Bonus points available!
  ⇒ Submit your results until evening of 23.05.

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
    \[ N: \text{number of training examples} \]
    \[ p^*: \text{probability that the bound is correct} \]
    \[ h: \text{capacity of the learning machine ("VC-dimension")} \]
Recap: VC Dimension

- Vapnik-Chervonenkis dimension
  - Measure for the capacity of a learning machine.
- Formal definition:
  - If a given set of \( \ell \) points can be labeled in all possible \( 2^\ell \) 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: Structural Risk Minimization

- How can we implement Structural Risk Minimization?
  \[ R(\alpha) \cdot R_{\text{emp}}(\alpha) + \epsilon(N, p', h) \]
- Classic approach
  - Keep \( \epsilon(N, p', h) \) constant and minimize \( R_{\text{emp}}(\alpha) \).
  - \( \epsilon(N, p', h) \) can be kept constant by controlling the model parameters.
- 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).

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 \( \mathbf{w}^T \mathbf{x} + b = 0 \)
- Formulation as a convex optimization problem
  - Find the hyperplane satisfying
    \[ \arg\min_{\mathbf{w}, b} \frac{1}{2} ||\mathbf{w}||^2 \]
    under the constraints
    \[ t_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1 \quad \forall n \]
    based on training data points \( \mathbf{x}_n \) and target values \( t_n \in \{-1, 1\} \).

Recap: Upper Bound on the Risk

- Important result (Vapnik 1979, 1995)
  - With probability \((1-\eta)\), the following bound holds
    \[ R(\alpha) \cdot R_{\text{emp}}(\alpha) + \sqrt{\frac{h \log(2N/h) + 1}{N} - \log(\eta/4)} \]
  - This bound is independent of \( P_{X,Y}(x, y) \).
  - If we know \( h \) (the VC dimension), we can easily compute the risk bound
    \[ R(\alpha) \cdot R_{\text{emp}}(\alpha) + \epsilon(N, p', h) \]

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
  - Mercer’s condition
  - Popular kernels
- Applications

Recap: SVM - Primal Formulation

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

- Solution for the k
  
  \[ w = \sum_{n=1}^{N} a_n t_n X_n \]
  
  - Sparse solution: \( a_n \neq 0 \) only for some points, the support vectors
  
  \[ \Rightarrow \text{Only the SVs actually influence the decision boundary!} \]
  
  - Compute \( b \) by averaging over all support vectors:
  
  \[ b = \frac{1}{N_S} \sum_{n \in S} \left( t_n - \sum_{m \in S} a_m t_m X_m^T X_n \right) \]

Recap: SVM - Support Vectors

- The training points for which \( a_n > 0 \) are called “support vectors”.
  
  - Graphical interpretation:
    - The support vectors are the points on the margin.
    - They define the margin and thus the hyperplane.
    
  \[ \Rightarrow \text{All other data points can be discarded!} \]

Recap: SVM - Discussion (Part 1)

- Linear SVM
  
  - Linear classifier
  
  - Approximative implementation of the SRM principle.
  
  - In case of separable data, the SVM produces an empirical risk of zero with minimal value of the VC confidence (i.e. a classifier minimizing the upper bound on the actual risk).
  
  - SVMs thus have a “guaranteed” generalization capability.
  
  - Formulation as convex optimization problem.
  
  \[ \Rightarrow \text{Globally optimal solution!} \]

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

Recap: SVM - Dual Formulation

- Improving the scaling behavior: rewrite \( L_p \) in a dual form
  
  \[ L_p = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left( t_n (w^T X_n + b) - 1 \right) \]
  
  \[ = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n t_n X_n^T X_n - b \sum_{n=1}^{N} a_n t_n + \sum_{n=1}^{N} a_n \]

  \[ \Rightarrow \text{Using the constraint } \sum_{n=1}^{N} a_n t_n = 0 , \text{ we obtain} \]
  
  \[ \frac{\partial L_p}{\partial b} = 0 \]

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

SVM - Dual Formulation

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

- Applying \( \frac{1}{2} \|w\|^2 = \frac{1}{2} w^T w \) and again using \( w = \sum_{n=1}^{N} a_n t_n X_n \)

\[ \frac{1}{2} w^T w = \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (X_n^T X_m) \]

- Inserting this, we get the Wolfe dual

\[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (X_n^T X_m) \]
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
  \[ \begin{align*}
  a_n &\geq 0 \quad \forall n \\
  \sum_{n=1}^{N} a_n t_n &= 0 \\
  \end{align*} \]

- The hyperplane is given by the \( N \) support vectors:
  \[ w = \sum_{n=1}^{N} \alpha_n t_n x_n \]

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! 
     - This makes it possible to construct efficient algorithms 
     - e.g. Sequential Minimal Optimization (SMO) 
     - Effective runtime between \( O(N) \) and \( O(N^2) \).
  2. We have avoided the dependency on the dimensionality. 
     - This makes it possible to work with infinite-dimensional feature spaces by using suitable basis functions \( \phi(x) \).
     - We’ll see that in a few minutes...

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
  \[ 
  w^T x_n + b \geq +1 \quad \text{for } t_n = +1 \\
  w^T x_n + b \leq -1 \quad \text{for } t_n = -1 
  \]
  - Instead use
  \[ 
  \begin{align*}
  w^T x_n + b &\geq +1 - \xi_n \quad \text{for } t_n = +1 \\
  w^T x_n + b &\leq -1 + \xi_n \quad \text{for } t_n = -1 
  \end{align*} \]
  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.
  - \( \xi_n = ||x_n - y(x_n)|| \) for all other points (linear penalty).

- We do not have to set the slack variables ourselves! 
  - They are jointly optimized together with \( w \).
**SVM - New Primal Formulation**

- New SVM Primal: Optimize

\[ L_p = \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} a_n (t_n y(x_n) - 1 + \xi_n) - \sum_{n=1}^{N} \mu_n \xi_n \]

**KKT conditions**

- \( a_n \geq 0 \quad \mu_n \geq 0 \)
- \( t_n y(x_n) - 1 + \xi_n \geq 0 \quad \xi_n \geq 0 \)
- \( a_n (t_n y(x_n) - 1 + \xi_n) = 0 \quad \mu_n \xi_n = 0 \)

**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

- \( 0 \cdot a_n \leq C \)
- \( \sum_{n=1}^{N} a_n t_n = 0 \)

- This is again a quadratic programming problem

\( \Rightarrow \) Solve as before... (more on that later)

**SVM - New Solution**

- Solution for the hyperplane

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

- 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 \( N \) points with \( 0 < a_n < C \):

\[ b = \frac{1}{NM} \sum_{n \in M} t_n - \sum_{m \in M} a_m t_m (x_m^T x_n) \]

**Interpretation of Support Vectors**

- Those are the hard examples!

\( \Rightarrow \) We can visualize them, e.g. for face detection

**Nonlinear SVM**

- Linear SVMs

\( \Rightarrow \) Datasets that are linearly separable with some noise work well:

- But what are we going to do if the dataset is just too hard?

\( \Rightarrow \) How about... mapping data to a higher-dimensional space:

**So Far...**

- Only looked at linearly separable case...

\( \Rightarrow \) Current problem formulation has no solution if the data are not linearly separable!

\( \Rightarrow \) Need to introduce some tolerance to outlier data points.

\( \Rightarrow \) Slack variables.

- Only looked at linear decision boundaries...

\( \Rightarrow \) This is not sufficient for many applications.

\( \Rightarrow \) Want to generalize the ideas to non-linear boundaries.
Another Example

- Non-separable by a hyperplane in 2D

Slide credit: Bill Freeman

Another Example

- Separable by a surface in 3D

Slide credit: Bill Freeman

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:

\[ \phi: \mathbb{R}^2 \rightarrow \mathcal{H} \]

Slide credit: Raymond Mooney

Nonlinear SVM

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

Slide credit: Bernt Schiele

What Could This Look Like?

- Example:
  - Mapping to polynomial space, \( x, y \in \mathbb{R}^2 \):
    \[ \phi(x) = \left[ \frac{x^2}{\sqrt{2}}, \frac{x^2}{2}, x_1 x_2 \right] \]

Motivation: Easier to separate data in higher-dimensional space.
But wait - isn't there a big problem?
How should we evaluate the decision function?

Slide credit: C. Burges, 1998

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
    \[ w = \sum_{n=1}^{N} \alpha_n t_n \phi(x_n) \]
  - \( \Rightarrow \) What happens if we try this for a million-dimensional feature space \( \phi(x) \)?
    - Oh-oh...

Slide credit: B. Leibe
Important observation

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

- 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:

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

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

Solution: The Kernel Trick

Back to Our Previous Example...

- 2nd degree polynomial kernel:

\[ \phi(x)^T \phi(y) = \left[ \begin{array}{c} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{array} \right] \left[ \begin{array}{c} y_1^2 \\ \sqrt{2} y_1 y_2 \\ y_2^2 \end{array} \right] \]
\[ = x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2 \]
\[ = (x^T y)^2 = k(x,y) \]

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.

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...)

Nonlinear SVM - Dual Formulation

- SVM Dual: Maximize

\[ L_d(a) = \sum_{n=1}^{N} a_n - \frac{1}{\gamma} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m) \]

under the conditions

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

- Classify new data points using

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

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

B. Leibe

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!
  - Since only the kernel function is used by the SVM, this is no problem.
  - The hyperplane in $\mathcal{H}$ then has VC-dimension
    $$ \dim(H) + 1 = \infty $$

B. Leibe

Example: RBF Kernels

- Decision boundary on toy problem

RBF Kernel width ($\sigma$)

B. Leibe

Image source: B. Schoelkopf, A. Smola, 2002

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: $a_1, a_2, \ldots, a_N$ and most of them are usually set to zero by the maximum margin criterion.
  - The data effectively lives in a low-dimensional subspace of $H$.

- 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.

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

B. Leibe

Theoretical Justification for Maximum Margins

- Vapnik has proven the following:
  - The class of optimal linear separators has VC dimension $h$ bounded from above as
    $$ h \leq \min \left\{ \frac{D^2}{\rho^2}, m_0 \right\} + 1 $$
    where $\rho$ 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 $\rho$.
  - Thus, complexity of the classifier is kept small regardless of dimensionality.

Slide credit: Raymond Mooney
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/

- 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 \( \langle \mathbf{x}, \mathbf{y} \rangle \) 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

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

- This is also how you could implement a simple spam filter...

![Diagram of text classification process]

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)
  - 5.1% 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 ($\sigma=0.3$, 291 support vectors)

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)

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: