Machine Learning - Lecture 7
Statistical Learning Theory
23.05.2016

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de
leibe@vision.rwth-aachen.de

Course Outline
- Fundamentals (2 weeks)
  - Bayes Decision Theory
  - Probability Density Estimation
- Discriminative Approaches (5 weeks)
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Randomized Trees, Forests & Ferns
- Generative Models (4 weeks)
  - Bayesian Networks
  - Markov Random Fields

Topics of This Lecture
- Recap: Generalized Linear Discriminants
- Logistic Regression
  - Probabilistic discriminative models
  - Logistic sigmoid (logit function)
  - Cross-entropy error
  - Gradient descent
  - Iteratively Reweighted Least Squares
- Note on error functions
- Statistical Learning Theory
  - Generalization and overfitting
  - Empirical and actual risk
  - VC dimension
  - Empirical Risk Minimization
  - Structural Risk Minimization

Recap: Linear Discriminant Functions
- Basic idea
  - Directly encode decision boundary
  - Minimize misclassification probability directly.
- Linear discriminant functions
  \[ y(x) = w^T x + w_0 \]
  - weight vector
  - "bias" (= threshold)
  - If a data set can be perfectly classified by a linear discriminant, then we call it linearly separable.

Recap: Extension to Nonlinear Basis Fcts.
- Generalization
  - Transform vector \( x \) with \( M \) nonlinear basis functions \( \phi_j(x) \):
  \[ y_k(x) = \sum_{j=1}^{M} w_{kj} \phi_j(x) + w_{k0} \]
- Advantages
  - Transformation allows non-linear decision boundaries.
  - By choosing the right \( \phi_j \), every continuous function can (in principle) be approximated with arbitrary accuracy.
- Disadvantage
  - The error function can in general no longer be minimized in closed form.
  \[ \Rightarrow \] Minimization with Gradient Descent

Recap: Basis Functions
- Generally, we consider models of the following form
  \[ y_k(x) = \sum_{j=0}^{M} w_{kj} \phi_j(x) = w^T \phi(x) \]
  - where \( \phi_j(x) \) are known as basis functions.
  - In the simplest case, we use linear basis functions: \( \phi_j(x) = x_j \).
- Other popular basis functions
  - Polynomial
  - Gaussian
  - Sigmoid
**Iterative minimization**

- Start with an initial guess for the parameter values \(w_0\).
- Move towards a (local) minimum by following the gradient.

**Basic strategies**

- "Batch learning" \(w_{k,j}^{(r+1)} = w_{k,j}^{(r)} - \eta \frac{\partial E(w)}{\partial w_{k,j}} |_{w(w)}\)
- "Sequential updating" \(w_{k,j}^{(r+1)} = w_{k,j}^{(r)} - \eta \frac{\partial E_n(w)}{\partial w_{k,j}} |_{w(w)}\)

Where \(E(w) = \sum_{n=1}^{N} E_n(w)\)

**Logistic sigmoid (logit function)**

\[ y(x_n; w) = \frac{1}{1 + e^{-w^T x_n}} \]

**Sequential updating leads to delta rule (LMS rule)**

\[ w_{k,j}^{(r+1)} = w_{k,j}^{(r)} - \eta \delta_{kn} \phi_j(x_n) \]

Where \( \delta_{kn} = y_k(x_n; w) - t_{kn} \)

→ Simply feed back the input data point, weighted by the classification error.

**Gradient descent**

- Cases with differentiable, non-linear activation function

\[ y_k(x) = g(a_k) = g \left( \sum_{j=0}^{M} w_{k,j} \phi_j(x_n) \right) \]

**Gradient descent (again with quadratic error function)**

\[ \frac{\partial E_n(w)}{\partial w_{k,j}} = \frac{\partial g(a_k)}{\partial w_{k,j}} (y_k(x_n; w) - t_{kn}) \phi_j(x_n) \]

\[ w_{k,j}^{(r+1)} = w_{k,j}^{(r)} - \frac{\partial E_n(w)}{\partial w_{k,j}} |_{w(w)} \]

\[ \delta_{kn} = \frac{\partial g(a_k)}{\partial w_{k,j}} (y_k(x_n; w) - t_{kn}) \]

**Recap: Gradient Descent**

- Example: Quadratic error function

\[ E(w) = \sum_{n=1}^{N} (y(x_n; w) - t_{kn})^2 \]

**Recap: Classification as Dim. Reduction**

- Consider linear classification as a projection onto a lower-dim. space.

\[ y = w^T x \]

→ Learning problem: Try to find the projection vector \( w \) that maximizes class separation.

**Recap: Fisher’s Linear Discriminant Analysis**

- Maximize distance between classes
- Minimize distance within a class
- Criterion: \( J(w) = \frac{w^T S_B w}{w^T S_W w} \)

\( S_B \) = between-class scatter matrix
\( S_W \) = within-class scatter matrix

- The optimal solution for \( w \) can be obtained as:

\[ w = S_B^{-1} (m_2 - m_1) \]

- Classification function:

\[ y(x) = w^T x + w_0 \]

\[ y(x) \leq 0 \quad \text{Class 1} \]

\[ y(x) > 0 \quad \text{Class 2} \]

\[ w_0 = -w^T m \]

**Topics of This Lecture**

- Recap: Generalized Linear Discriminants
- Logistic Regression
  - Probabilistic discriminative models
  - Logistic sigmoid (logit function)
  - Cross-entropy error
  - Gradient descent
  - Iteratively Reweighted Least Squares
- Note on error functions
- Statistical Learning Theory
  - Generalization and overfitting
  - Empirical and actual risk
  - VC dimension
  - Empirical Risk Minimization
  - Structural Risk Minimization
We have seen that we can write
\[ p(C_1|x) = \sigma(a) = \frac{1}{1 + \exp(-a)} \]

We can obtain the familiar probabilistic model by setting
\[ a = \ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)} \]

Or we can use generalized linear discriminant models
\[ a = w^T x \text{ or } a = w^T \phi(x) \]

This model is called logistic regression.

Why should we do this? What advantage does such a model have compared to modeling the probabilities?
\[ p(C_1|\phi) = y(\phi) = \sigma(w^T \phi) \]

with
\[ p(C_2|\phi) = 1 - p(C_1|\phi) \]

Any ideas?

For large M, logistic regression has clear advantages!

Comparison

Let's look at the number of parameters...

- Assume we have an \( M \)-dimensional feature space \( \phi \).
- And assume we represent \( p(\phi|C_1) \) and \( p(\phi|C_2) \) by Gaussians.
- How many parameters do we need?
  - For the means: \( 2M \)
  - For the covariances: \( M(M+1)/2 \)
  - Together with the class priors, this gives \( M(M+5)/2+1 \) parameters!
- How many parameters do we need for logistic regression?
  - Just the values of \( w \) \( \Rightarrow \) \( M \) parameters.

Logistic Sigmoid

Properties
- Definition:
  \[ \sigma(a) = \frac{1}{1 + \exp(-a)} \]
- Inverse:
  \[ a = \ln \left( \frac{\sigma}{1 - \sigma} \right) \]
- Symmetry property:
  \[ \sigma(-a) = 1 - \sigma(a) \]
- Derivative:
  \[ \frac{d\sigma}{da} = \sigma(1 - \sigma) \]

Logistic Regression

Let's consider a data set \( \{\phi_n, t_n\} \) with \( n = 1, \ldots, N \), where \( \phi_n = \phi(x_n) \) and \( t_n \in \{0, 1\} \), \( t = (t_1, \ldots, t_N)^T \).

With \( y_n = p(C_1|\phi_n) \), we can write the likelihood as
\[ p(t|w) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1-t_n} \]

Define the error function as the negative log-likelihood
\[ E(w) = -\ln p(t|w) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln (1 - y_n)\} \]

This is the so-called cross-entropy error function.

Gradient of the Error Function

Error function
\[ y_n = \sigma(w^T \phi_n) \]
\[ \frac{dy_n}{dw} = y_n(1-y_n) \phi_n \]
\[ E(w) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln (1 - y_n)\} \]

Gradient
\[ \nabla E(w) = -\sum_{n=1}^{N} \left\{ t_n \frac{\partial y_n}{\partial y_n} + (1 - t_n) \frac{\partial y_n}{\partial (1 - y_n)} \right\} \phi_n \]
\[ = -\sum_{n=1}^{N} \left\{ t_n \frac{1 - y_n}{y_n} \phi_n + (1 - t_n) \frac{y_n}{1 - y_n} \phi_n \right\} \]
\[ = \sum_{n=1}^{N} \left\{ (t_n - y_n) \phi_n - y_n + y_n \phi_n \right\} \]
\[ = \sum_{n=1}^{N} (y_n - t_n) \phi_n \]
Gradient of the Error Function

- Gradient for logistic regression
  \[ \nabla E(w) = \sum_{n=1}^{N} (y_n - t_n) \phi_n \]

- Does this look familiar to you?
- This is the same result as for the Delta (=LMS) rule
  \[ w_{k+1} = w_k - \eta (y_k(x_n; w) - t_k) \phi_j(x_n) \]
- We can use this to derive a sequential estimation algorithm.
  However, this will be quite slow.

Newton-Raphson for Least-Squares Estimation

- Let’s first apply Newton-Raphson to the least-squares error function:
  \[ E(w) = \frac{1}{2} \sum_{n=1}^{N} (w^T \phi_n - t_n)^2 \]
  \[ \nabla E(w) = \sum_{n=1}^{N} (w^T \phi_n - t_n) \phi_n = \Phi^T \Phi w - \Phi^T t \]
  \[ H = \nabla^2 E(w) = \sum_{n=1}^{N} \phi_n \phi_n^T = \Phi^T \Phi \]
  where \( \Phi = \begin{bmatrix} \phi_1^T \\ \vdots \\ \phi_N^T \end{bmatrix} \)
- Resulting update scheme:
  \[ w^{(r+1)} = w^{(r)} - (\Phi^T \Phi)^{-1} \Phi^T (y - t) \]
  Closed-form solution!

Iteratively Reweighted Least Squares

- Update equations
  \[ w^{(r+1)} = w^{(r)} - (\Phi^T R \Phi)^{-1} \Phi^T (y - t) \]
  \[ = (\Phi^T R \Phi)^{-1} (\Phi^T R w^{(r)} - \Phi^T t) \]
  
  with \( z = \Phi w^{(r)} - R^{-1}(y - t) \)
- Again very similar form (normal equations)
  - But now with non-constant weighting matrix \( R \) (depends on \( w \)).
  - Need to apply normal equations iteratively.
  \( \Rightarrow \text{Iteratively Reweighted Least-Squares (IRLS)} \)

A More Efficient Iterative Method...

- Second-order Newton-Raphson gradient descent scheme
  \[ w^{(r+1)} = w^{(r)} - H^{-1} \nabla E(w) \]
  where \( H = \nabla^2 E(w) \) is the Hessian matrix, i.e. the matrix of second derivatives.

- Properties
  - Local quadratic approximation to the log-likelihood.
  - Faster convergence.

Newton-Raphson for Logistic Regression

- Now, let’s try Newton-Raphson on the cross-entropy error function:
  \[ E(w) = -\sum_{n=1}^{N} \{ t_n \ln y_n + (1 - t_n) \ln (1 - y_n) \} \]
  \[ \frac{\partial E}{\partial w} = \sum_{n=1}^{N} (y_n - t_n) \phi_n = \Phi^T (y - t) \]
  \[ H = \nabla^2 E(w) = \sum_{n=1}^{N} y_n (1 - y_n) \phi_n \phi_n^T = \Phi^T R \Phi \]
  where \( R \) is an \( N \times N \) diagonal matrix with \( R_{nn} = y_n (1 - y_n) \).
  \( \Rightarrow \) The Hessian is no longer constant, but depends on \( w \) through the weighting matrix \( R \).

Summary: Logistic Regression

- Properties
  - Directly represent posterior distribution \( p(\phi | \mathcal{D}) \)
  - Requires fewer parameters than modeling the likelihood + prior.
  - Very often used in statistics.
  - It can be shown that the cross-entropy error function is concave.
    Optimization leads to unique minimum
  - But no closed-form solution exists
  - Iterative optimization (IRLS)
  - Both online and batch optimizations exist
  - There is a multi-class version described in (Bishop Ch.4.3.4).

- Caveat
  - Logistic regression tends to systematically overestimate odds ratios when the sample size is less than -500.
Topics of This Lecture

- Recap: Generalized Linear Discriminants
- Logistic Regression
  - Probabilistic discriminative models
  - Logistic sigmoid (logit function)
  - Cross-entropy error
  - Gradient descent
  - Iteratively Reweighted Least Squares
- Note on error functions
- Statistical Learning Theory
  - Generalization and overfitting
  - Empirical and actual risk
  - VC dimension
  - Empirical Risk Minimization
  - Structural Risk Minimization

A Note on Error Functions

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

- Ideal misclassification error
  - Not differentiable!
  - We cannot minimize it by gradient descent.

- Cross-Entropy Error
  - Minimizer of this error is given by posterior class probabilities.
  - Concave error function, unique minimum exists.
  - Robust to outliers, error increases only roughly linearly
  - But no closed-form solution, requires iterative estimation.

A Note on Error Functions

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

- 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
  - Generally does not lead to good classifiers.

- Cross-Entropy Error
  - Minimizer of this error is given by posterior class probabilities.
  - Concave error function, unique minimum exists.
  - Robust to outliers, error increases only roughly linearly
  - But no closed-form solution, requires iterative estimation.

- Squared error with sigmoid activation function (tanh)
  - Fixes the problems with outliers and "too correct" data points.
  - But: zero gradient for confidently misclassified data points.
  - Will give better performance than original squared error, but still does not fix all problems.
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.
  \( \Rightarrow \) Overfitting to the training set!

Difficulty: how should the risk be estimated?
However, they will most likely result in different predictions on novel test data.
\( \Rightarrow \) Different generalization performance

Overfitting is often a problem with linearly separable data
- Which of the many possible decision boundaries is correct?
- All of them have zero error on the training set...
- However, they will most likely result in different predictions on novel test data.
  \( \Rightarrow \) Different generalization performance

How to select the classifier with the best generalization performance?

A Broader View on Statistical Learning

- Formal treatment: Statistical Learning Theory
- Supervised learning
  - Environment: assumed stationary.
    - i.e. the data \( x \) have an unknown but fixed probability density \( p_X(x) \)
  - Teacher: specifies for each data point \( x \) the desired classification \( y \) (where \( y \) may be subject to noise).
    \[ y = g(x, \nu) \] with noise \( \nu \)
  - Learning machine: represented by class of functions, which produce for each \( x \) an output \( y \):
    \[ y = f(x; \alpha) \] with parameters \( \alpha \)

Statistical Learning Theory

- Supervised learning (from the learning machine’s view)
  - Selection of a specific function \( f(x; \alpha) \)
  - Given: training examples \( \{(x_i, y_i)\}_{i=1}^N \)
  - Goal: the desired response \( y \) shall be approximated optimally.

Measuring the optimality
- Loss function
  \[ L(y, f(x; \alpha)) \]
  - Example: quadratic loss
    \[ L(y, f(x; \alpha)) = (y - f(x; \alpha))^2 \]

Risk

- Measuring the “optimality”
  - Measure the optimality by the risk (= expected loss).
  - Difficulty: how should the risk be estimated?

Practical way
- Empirical risk (measured on the training/validation set)
  \[ R_{\text{emp}}(\alpha) = \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i; \alpha)) \]
  - Example: quadratic loss function
    \[ R_{\text{emp}}(\alpha) = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i; \alpha))^2 \]

Risk

However, what we’re really interested in is
- Actual risk (= Expected risk)
  \[ R(\alpha) = \int L(y, f(x; \alpha)) dP_{X,Y}(x, y) \]
  - \( P_{X,Y}(x, y) \) is the probability distribution of \((x,y)\).
  - \( P_{X,Y}(x, y) \) is fixed, but typically unknown.
  \( \Rightarrow \) In general, we can’t compute the actual risk directly!
- The expected risk is the expectation of the error on all data.
  - I.e., it is the expected value of the generalization error.
**Summary: Risk**

- Actual risk
  - Advantage: measure for the generalization ability
  - Disadvantage: in general, we don’t know $P_{X,Y}(x, y)$
- Empirical risk
  - Disadvantage: no direct measure of the generalization ability
  - Advantage: does not depend on $P_{X,Y}(x, y)$
  - We typically know learning algorithms which minimize the empirical risk.

**Statistical Learning Theory**

- Idea
  - Compute an upper bound on the actual risk based on the empirical risk
    $$ R(\alpha) \cdot R_{emp}(\alpha) + \epsilon(\alpha, p, h) $$
  - where
    - $N$: number of training examples
    - $p$: probability that the bound is correct
    - $h$: capacity of the learning machine ("VC-dimension")
- Side note:
  - (This idea of specifying a bound that only holds with a certain probability is explored in a branch of learning theory called “Probably Approximately Correct” or PAC Learning).

**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 $\{f(\alpha)\}$.

**VC Dimension**

- Example
  - The VC dimension of all oriented lines in $\mathbb{R}^2$ is 3.
    1. Shattering 3 points with an oriented line:
      ![Shattering 3 points](image)
    2. More difficult to show: it is not possible to shatter 4 points (XOR).
- More general: the VC dimension of all hyperplanes in $\mathbb{H}^n$ is $n+1$.

**VC Dimension**

- Interpretation as a two-player game
  - Opponent’s turn: He says a number $N$.
  - Our turn: We specify a set of $N$ points $\{x_1, ..., x_N\}$.
  - Opponent’s turn: He gives us a labeling $\{x_1, ..., x_N\} \in \{0, 1\}^N$
  - Our turn: We specify a function $f(\alpha)$ which correctly classifies all $N$ points.

**VC Dimension**

- Intuitive feeling (unfortunately wrong)
  - The VC dimension has a direct connection with the number of parameters.
  - Counterexample
    $$ f(x; \alpha) = g(\sin(\alpha x)) $$
    $$ g(x) = \begin{cases} 
      1, & x > 0 \\
      -1, & x \leq 0 
    \end{cases} $$
  - Just a single parameter $\alpha$.
  - Infinite VC dimension
    - Proof: Choose $x_i = 10^{-i}$, $i = 1, \ldots, \ell$
    - $\alpha = \pi \left( 1 + \sum_{i=1}^{\ell} \frac{(1 - y_i)10^i}{2} \right)$
Upper Bound on the Risk

- Important result (Vapnik 1979, 1995)
  - With probability $(1-\eta)$, the following bound holds
  $$ R(\alpha) \cdot R_{\text{emp}}(\alpha) + \sqrt{\frac{h(\log(2N/h) + 1) - \log(\eta/4)}{N}} $$
  - This bound is independent of $P_{\mathcal{X},\mathcal{Y}}(x,y)$
  - Typically, we cannot compute the left-hand side (the actual risk)
  - If we know $h$ (the VC dimension), we can however easily compute the risk bound
    $$ R(\alpha) \cdot R_{\text{emp}}(\alpha) + \epsilon(N, p^*, h) $$

Structural Risk Minimization

- How can we implement this?
  $$ R(\alpha) \cdot R_{\text{emp}}(\alpha) + \epsilon(N, p^*, h) $$

- 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).

References and Further Reading

- More information on SVMs can be found in Chapter 7.1 of Bishop’s book.
- Additional information about Statistical Learning Theory and a more in-depth introduction to SVMs are available in the following tutorial: