Machine Learning, Summer '16

RWTH AACHEN UNIVERSITY

Machine Learning – Lecture 7

Statistical Learning Theory

23.05.2016

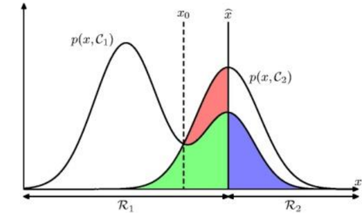
Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de

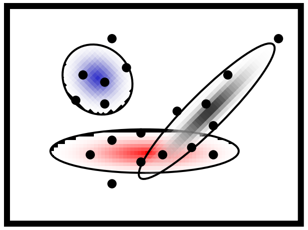
leibe@vision.rwth-aachen.de

Many slides adapted from B. Schiele

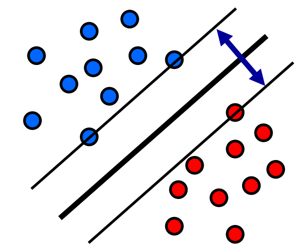
Course Outline

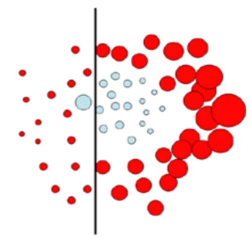
- Fundamentals (2 weeks)
  - Bayes Decision Theory
  - Probability Density Estimation



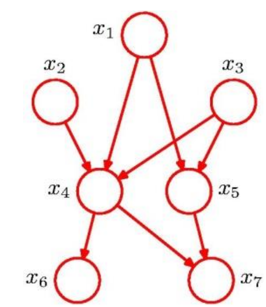


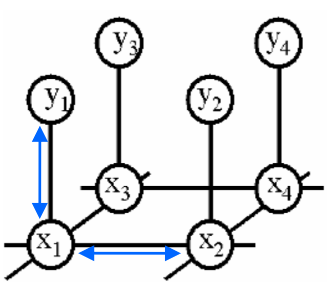
- Discriminative Approaches (5 weeks)
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Randomized Trees, Forests & Ferns





- 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

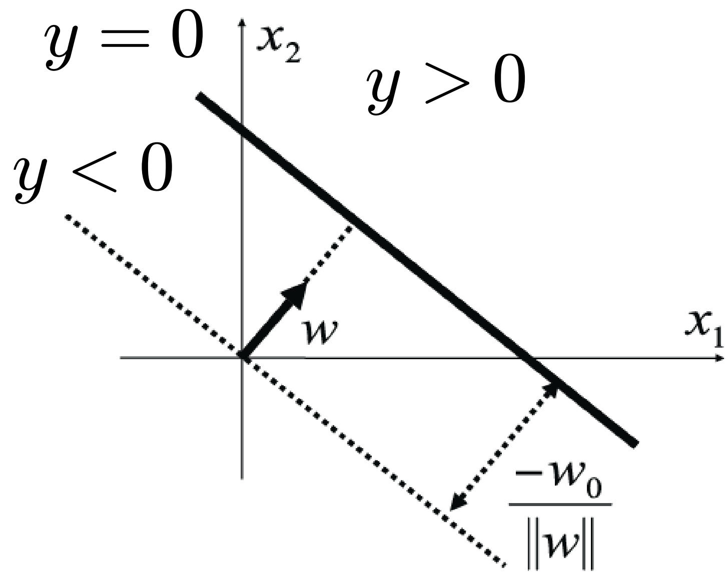
Recap: Linear Discriminant Functions

- **Basic idea**
  - Directly encode decision boundary
  - Minimize misclassification probability directly.

- **Linear discriminant functions**

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

weight vector → $\mathbf{w}$; "bias" (= threshold) → w_0



  - $\mathbf{w}$, w_0 define a hyperplane in $\mathbb{R}^D$.
  - 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 $\mathbf{x}$ with M nonlinear basis functions $\phi_j(\mathbf{x})$:

$$y_k(\mathbf{x}) = \sum_{j=1}^{M} w_{kj}\phi_j(\mathbf{x}) + w_{k0}$$

- **Advantages**
  - Transformation allows non-linear decision boundaries.
  - By choosing the right ϕ_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

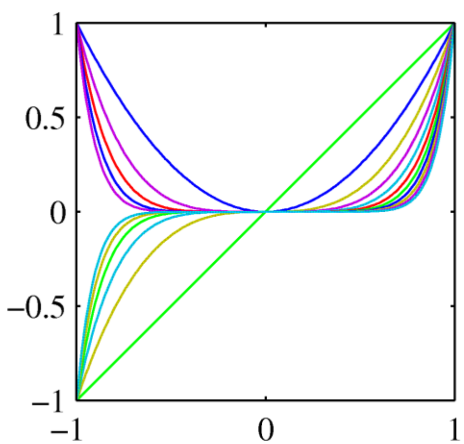
Recap: Basis Functions

- Generally, we consider models of the following form

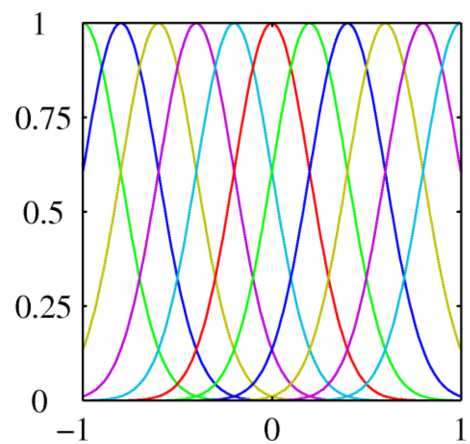
$$y_k(\mathbf{x}) = \sum_{j=0}^{M} w_{kj}\phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

  - where $\phi_j(\mathbf{x})$ are known as *basis functions*.
  - In the simplest case, we use linear basis functions: $\phi_d(\mathbf{x}) = x_d$.

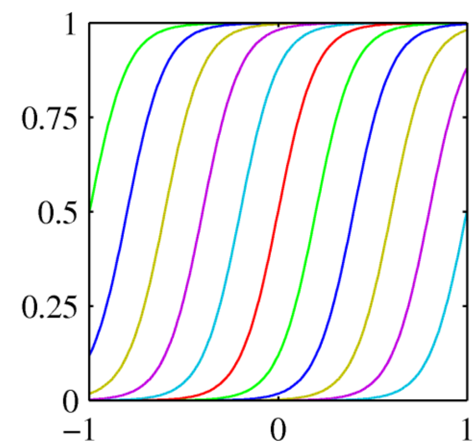
- Other popular basis functions



Polynomial



Gaussian



Sigmoid

Gradient Descent

- **Iterative minimization**
  - Start with an initial guess for the parameter values $w_{kj}^{(0)}$.
  - Move towards a (local) minimum by following the gradient.
- **Basic strategies**
  - "Batch learning"

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \right|_{\mathbf{w}^{(\tau)}}$$

  - "Sequential updating"

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E_n(\mathbf{w})}{\partial w_{kj}} \right|_{\mathbf{w}^{(\tau)}}$$

where

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

Recap: Gradient Descent

- **Example: Quadratic error function**

$$E(\mathbf{w}) = \sum_{n=1}^{N} (y(\mathbf{x}_n; \mathbf{w}) - \mathbf{t}_n)^2$$

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

$$\begin{aligned} w_{kj}^{(\tau+1)} &= w_{kj}^{(\tau)} - \eta \left(y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn}\right) \phi_j(\mathbf{x}_n) \\ &= w_{kj}^{(\tau)} - \eta \delta_{kn} \phi_j(\mathbf{x}_n) \end{aligned}$$

  - **where**

$$\delta_{kn} = y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn}$$

$\Rightarrow$ **Simply feed back the input data point, weighted by the classification error.**

Recap: Gradient Descent

- Cases with differentiable, non-linear activation function

$$y_k(\mathbf{x}) = g(a_k) = g\left(\sum_{j=0}^{M} w_{ki}\phi_j(\mathbf{x}_n)\right)$$

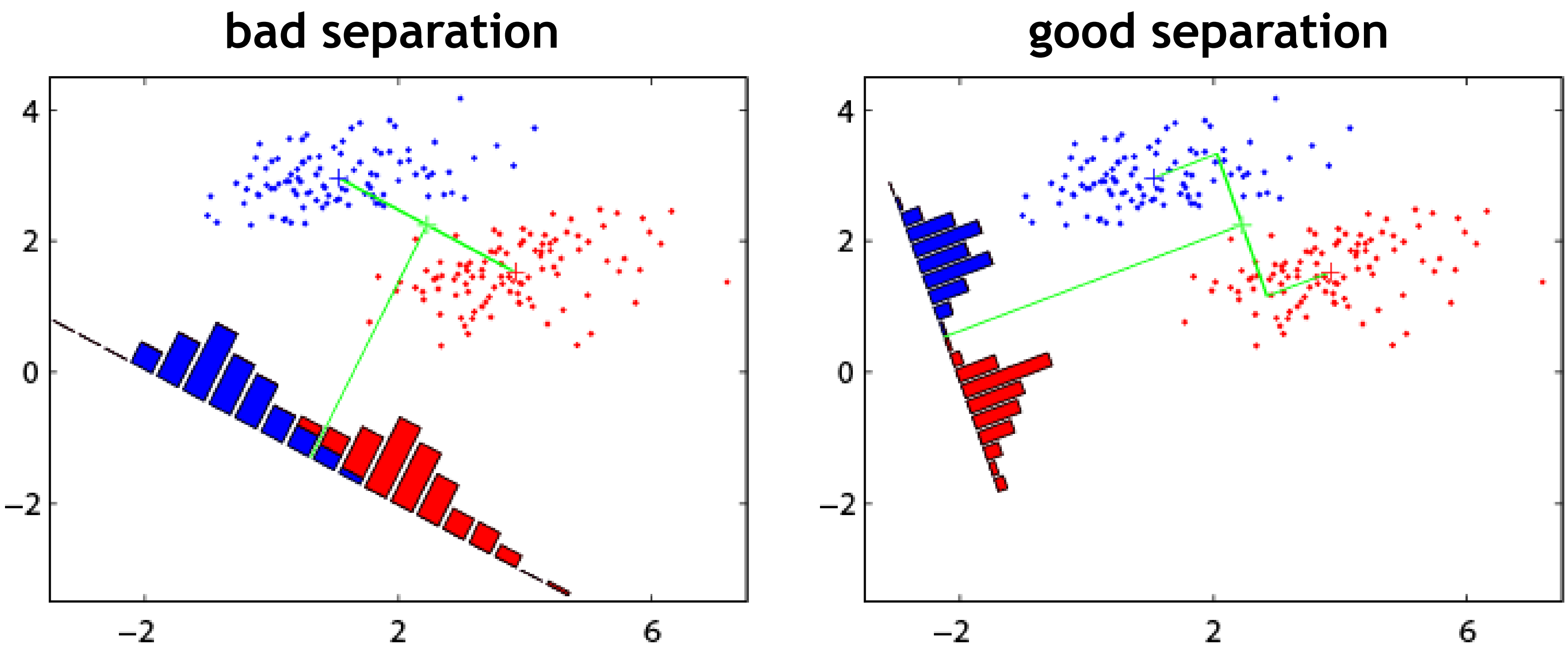
- Gradient descent (again with quadratic error function)

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{kj}} = \frac{\partial g(a_k)}{\partial w_{kj}} \left(y_k(\mathbf{x}_n;\mathbf{w}) - t_{kn}\right)\phi_j(\mathbf{x}_n)$$

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta\delta_{kn}\phi_j(\mathbf{x}_n)$$

$$\delta_{kn} = \frac{\partial g(a_k)}{\partial w_{kj}} \left(y_k(\mathbf{x}_n;\mathbf{w}) - t_{kn}\right)$$

Recap: Classification as Dim. Reduction



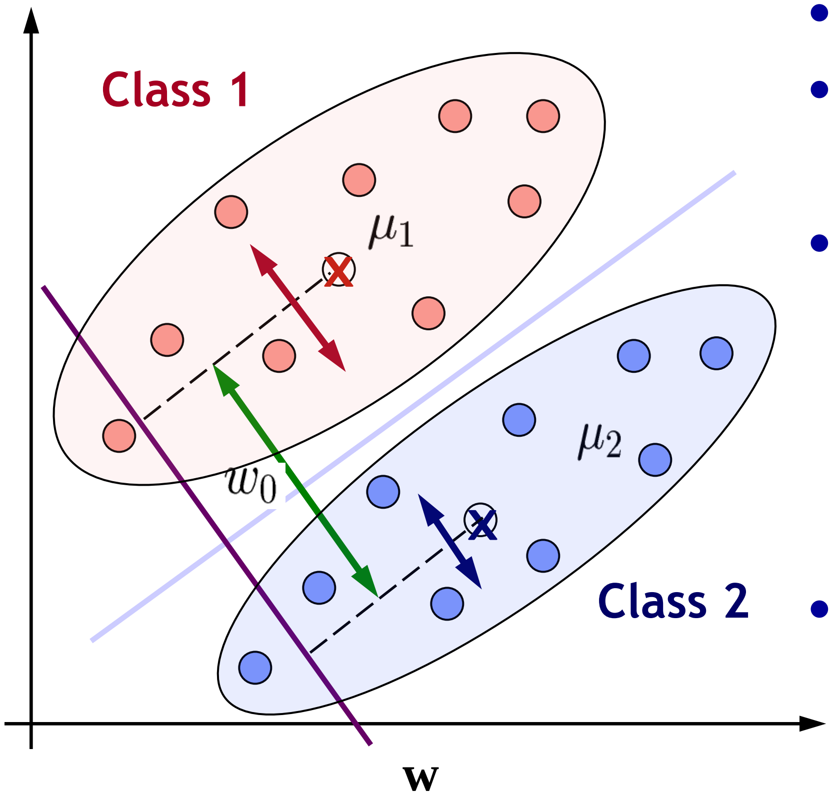
- **Classification as dimensionality reduction**
  - Interpret linear classification as a projection onto a lower-dim. space.

$$y = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

$\Rightarrow$ Learning problem: Try to find the projection vector $\mathbf{w}$ that maximizes class separation.

Image source: C.M. Bishop, 2006

Recap: Fisher's Linear Discriminant Analysis



- Maximize distance between classes
- Minimize distance within a class
- Criterion: $J(\mathbf{w}) = \dfrac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_B\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_W\mathbf{w}}$

  $\mathbf{S}_B$ ... between-class scatter matrix
  $\mathbf{S}_W$ ... within-class scatter matrix

- The optimal solution for $\mathbf{w}$ can be obtained as:

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

- Classification function:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 \underset{\text{Class } 2}{\overset{\text{Class } 1}{\gtrless}} 0$$

where $w_0 = -\mathbf{w}^T\mathbf{m}$

Slide adapted from Ales Leonardis

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

Probabilistic Discriminative Models

- We have seen that we can write

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(a) = \frac{1}{1+\exp(-a)}$$

logistic sigmoid function

- We can obtain the familiar probabilistic model by setting

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

- Or we can use generalized linear discriminant models

$$a = \mathbf{w}^T\mathbf{x}$$

or $$a = \mathbf{w}^T\phi(\mathbf{x})$$

Probabilistic Discriminative Models

- In the following, we will consider models of the form

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathrm{w}^T\phi)$$

with $$p(\mathcal{C}_2|\phi) = 1 - p(\mathcal{C}_1|\phi)$$

- This model is called logistic regression.

- Why should we do this? What advantage does such a model have compared to modeling the probabilities?

$$p(\mathcal{C}_1|\phi) = \frac{p(\phi|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\phi|\mathcal{C}_1)p(\mathcal{C}_1) + p(\phi|\mathcal{C}_2)p(\mathcal{C}_2)}$$

- Any ideas?

Comparison

- Let's look at the number of parameters…
  - Assume we have an M-dimensional feature space ϕ.
  - And assume we represent $p(\phi|\mathcal{C}_k)$ and $p(\mathcal{C}_k)$ 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?

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathrm{w}^T\phi)$$

    - Just the values of $\mathrm{w} \Rightarrow M$ parameters.

$\Rightarrow$ ***For large M, logistic regression has clear advantages!***

Logistic Sigmoid

- **Properties**
  - **Definition:** $$\sigma(a) = \frac{1}{1+\exp(-a)}$$
  - **Inverse:** $$a = \ln\left(\frac{\sigma}{1-\sigma}\right)$$ "logit" function
  - **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 $\phi_n = \phi(\mathbf{x}_n)$ and $t_n \in \{0, 1\}$, $\mathbf{t} = (t_1, \ldots, t_N)^T$.**
- **With $y_n = p(\mathcal{C}_1|\phi_n)$, we can write the likelihood as**

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} \{1 - y_n\}^{1-t_n}$$

- **Define the error function as the negative log-likelihood**

$$\begin{aligned} E(\mathbf{w}) &= -\ln p(\mathbf{t}|\mathbf{w}) \\ &= -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} \end{aligned}$$

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

Gradient of the Error Function

$$y_n = \sigma(\mathbf{w}^T \phi_n)$$
$$\frac{dy_n}{d\mathbf{w}} = y_n(1-y_n)\phi_n$$

- **Error function**

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

- **Gradient**

$$\nabla E(\mathbf{w}) = -\sum_{n=1}^{N} \left\{ t_n \frac{\frac{d}{d\mathbf{w}} y_n}{y_n} + (1-t_n) \frac{\frac{d}{d\mathbf{w}}(1-y_n)}{(1-y_n)} \right\}$$
$$= -\sum_{n=1}^{N} \left\{ t_n \frac{\cancel{y_n}(1-y_n)}{\cancel{y_n}}\phi_n - (1-t_n)\frac{y_n\cancel{(1-y_n)}}{\cancel{(1-y_n)}}\phi_n \right\}$$
$$= -\sum_{n=1}^{N} \{(t_n - \cancel{t_n y_n} - y_n + \cancel{t_n y_n})\phi_n\}$$
$$= \sum_{n=1}^{N} (y_n - t_n)\phi_n$$

Gradient of the Error Function

- Gradient for logistic regression

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\phi_n$$

- Does this look familiar to you?

- This is the same result as for the Delta (=LMS) rule

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta(y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn})\phi_j(\mathbf{x}_n)$$

- We can use this to derive a sequential estimation algorithm.
  - However, this will be quite slow…

A More Efficient Iterative Method…

- Second-order Newton-Raphson gradient descent scheme

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \mathbf{H}^{-1} \nabla E(\mathbf{w})$$

where $\mathbf{H} = \nabla\nabla E(\mathbf{w})$ is the Hessian matrix, i.e. the matrix of second derivatives.

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

Newton-Raphson for Least-Squares Estimation

- Let's first apply Newton-Raphson to the least-squares error function:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^T\phi_n - t_n\right)^2$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}\left(\mathbf{w}^T\phi_n - t_n\right)\phi_n = \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{w} - \mathbf{\Phi}^T\mathbf{t}$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N}\phi_n\phi_n^T = \mathbf{\Phi}^T\mathbf{\Phi}$$

where $\mathbf{\Phi} = \begin{bmatrix} \phi_1^T \\ \vdots \\ \phi_N^T \end{bmatrix}$

- Resulting update scheme:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}(\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{w}^{(\tau)} - \mathbf{\Phi}^T\mathbf{t})$$

$$= (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{t}$$

Closed-form solution!

Newton-Raphson for Logistic Regression

- Now, let's try Newton-Raphson on the cross-entropy error function:

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

$$\frac{dy_n}{d\mathbf{w}} = y_n(1-y_n)\phi_n$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\phi_n = \mathbf{\Phi}^T(\mathbf{y}-\mathbf{t})$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi_n^T = \mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}$$

where $\mathbf{R}$ is an $N \times N$ diagonal matrix with $R_{nn} = y_n(1-y_n)$.

⇒ The Hessian is no longer constant, but depends on $\mathbf{w}$ through the weighting matrix $\mathbf{R}$.

Iteratively Reweighted Least Squares

- Update equations

$$\begin{aligned}\mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - (\boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T(\mathbf{y}-\mathbf{t}) \\ &= (\boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi})^{-1}\left\{\boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi}\mathbf{w}^{(\tau)} - \boldsymbol{\Phi}^T(\mathbf{y}-\mathbf{t})\right\} \\ &= (\boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{R}\mathbf{z}\end{aligned}$$

with $\mathbf{z} = \boldsymbol{\Phi}\mathbf{w}^{(\tau)} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})$

- Again very similar form (normal equations)
  - But now with non-constant weighing matrix $\mathbf{R}$ (depends on $\mathbf{w}$).
  - Need to apply normal equations iteratively.

  ⇒ Iteratively Reweighted Least-Squares (IRLS)

Summary: Logistic Regression

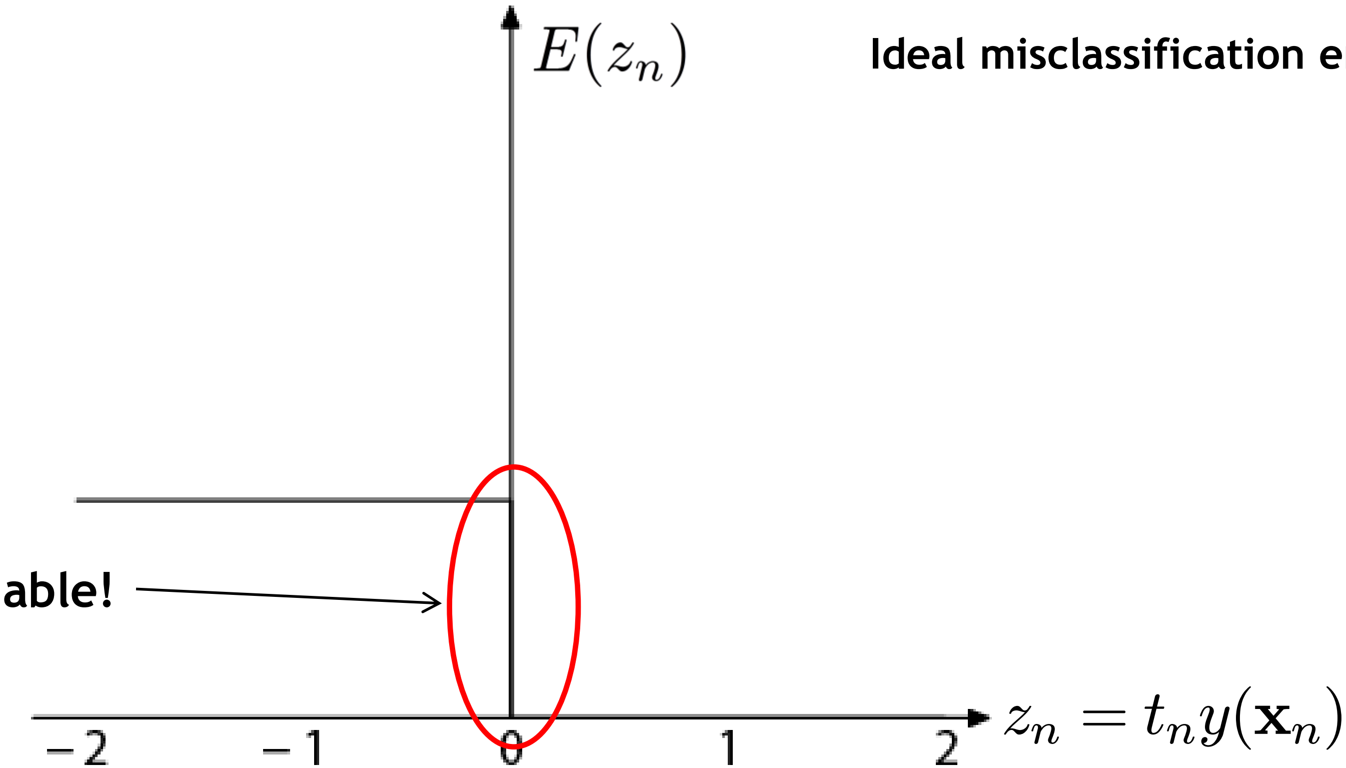
- Properties
  - Directly represent posterior distribution $p(\phi|\mathcal{C}_k)$
  - Requires fewer parameters than modeling the likelihood + prior.
  - Very often used in statistics.
  - It can be shown that the cross-entropy error function is concave
    - Optimization leads to unique minimum
    - 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

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



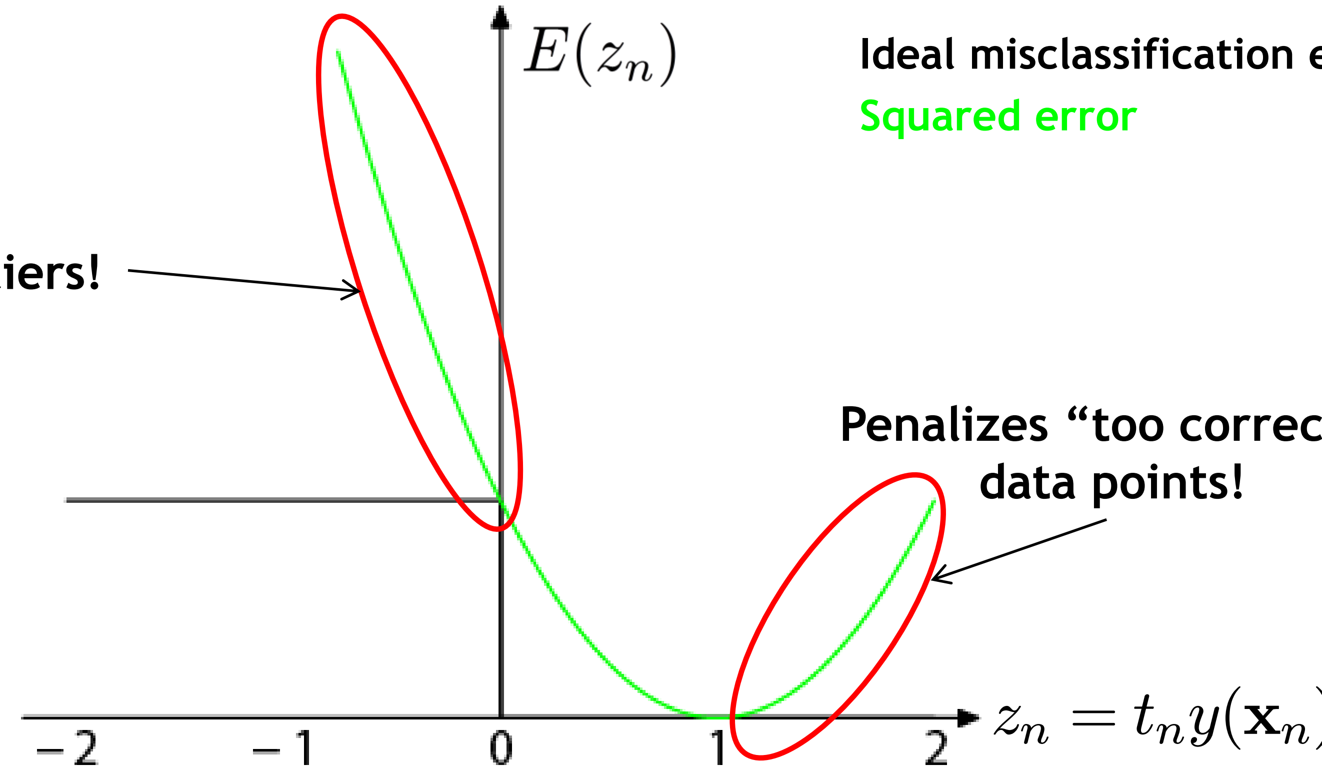
- Ideal misclassification error function (black)
  - This is what we want to approximate,
  - Unfortunately, it is not differentiable.
  - The gradient is zero for misclassified points.

  ⇒ We cannot minimize it by gradient descent.

Image source: Bishop, 2006

A Note on Error Functions

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

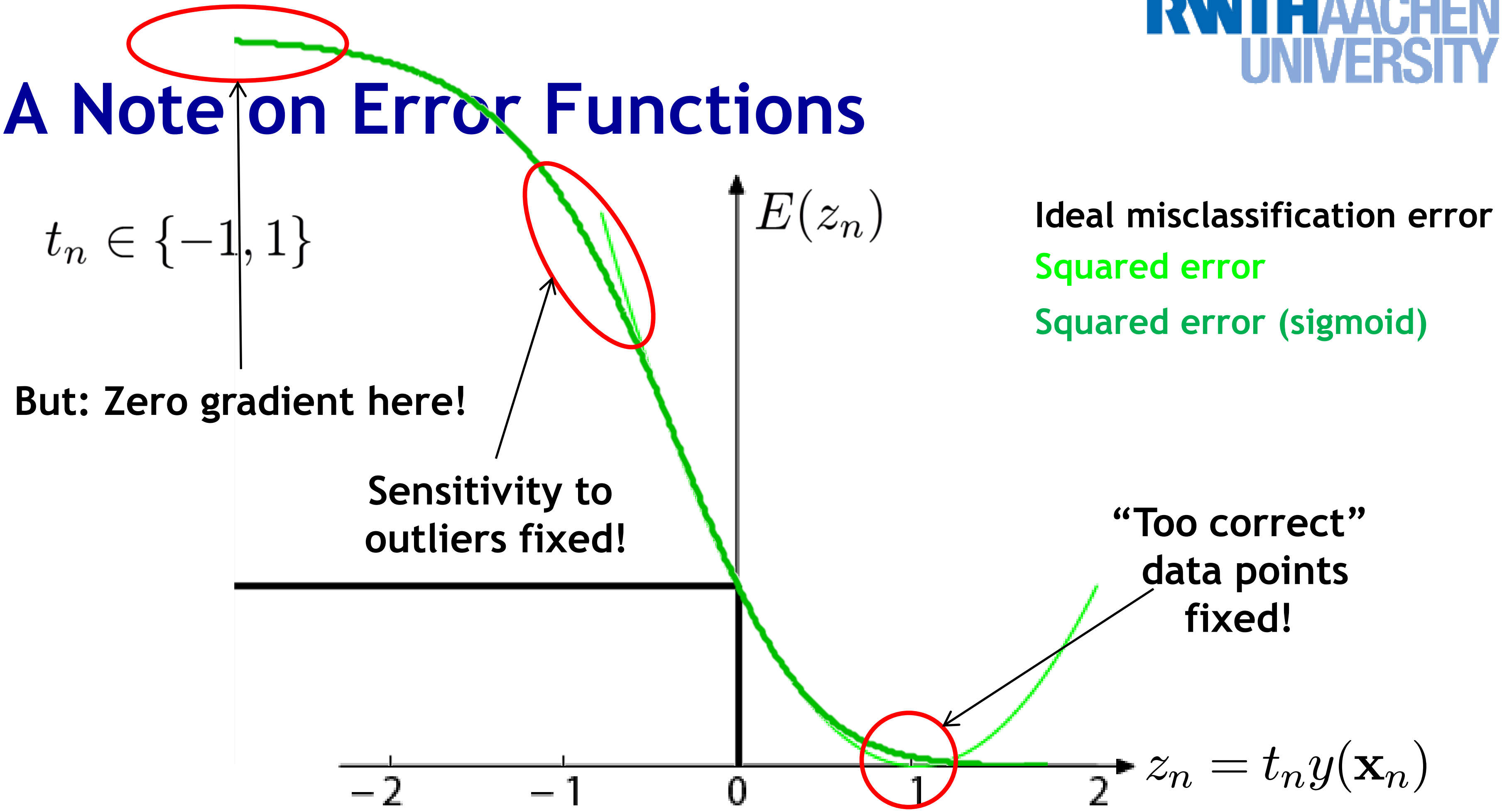


- Squared error used in Least-Squares Classification
  - Very popular, leads to closed-form solutions.
  - However, sensitive to outliers due to squared penalty.
  - Penalizes "too correct" data points

  $\Rightarrow$ Generally does not lead to good classifiers.

Image source: Bishop, 2006

A Note on Error Functions

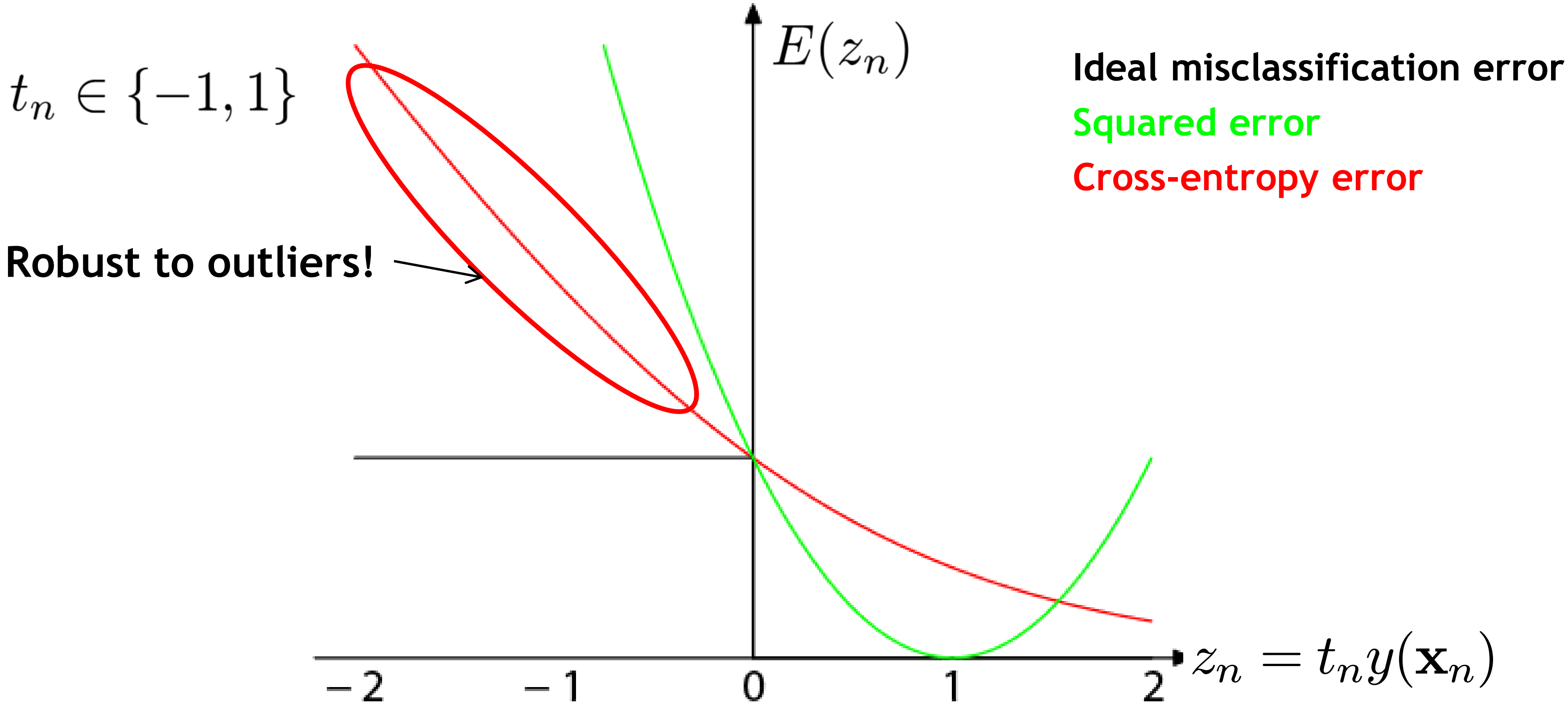


- Squared error with sigmoid activation function (tanh)
  - Fixes the problems with outliers and "too correct" data points.
  - But: zero gradient for confidently misclassified data points.

  $\Rightarrow$ Will give better performance than original squared error, but still does not fix all problems.

Image source: Bishop, 2006

A Note on Error Functions



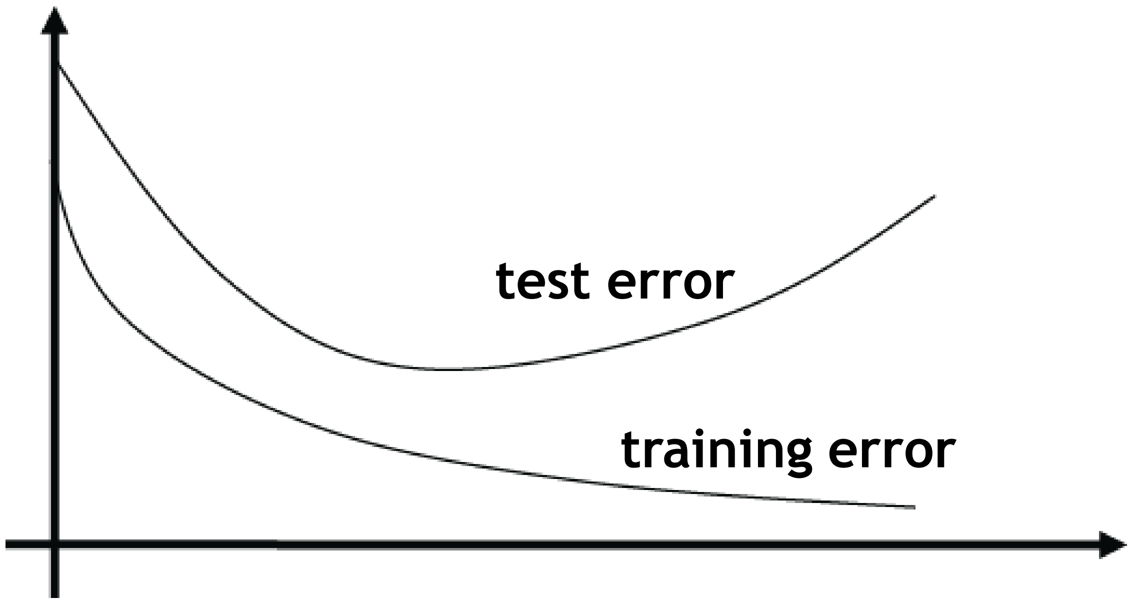
- **Cross-Entropy Error**
  - Minimizer of this error is given by posterior class probabilities.
  - Concave error function, unique minimum exists.
  - Robust to outliers, error increases only roughly linearly
  - But no closed-form solution, requires iterative estimation.

Image source: Bishop, 2006

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

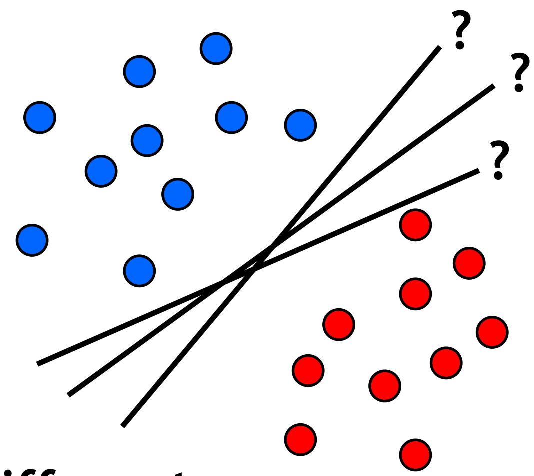
Generalization and Overfitting



- 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!*

Image source: B. Schiele

Example: Linearly Separable Data



- 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 $\mathbf{x}$ have an unknown but fixed probability density**

$$p_X(\mathbf{x})$$

  - **Teacher: specifies for each data point $\mathbf{x}$ the desired classification y (where y may be subject to noise).**

$$y = g(\mathbf{x}, \nu) \qquad \text{with noise } \nu$$

  - **Learning machine: represented by class of functions, which produce for each $\mathbf{x}$ an output y:**

$$y = f(\mathbf{x}; \alpha) \qquad \text{with parameters } \alpha$$

Slide credit: Bernt Schiele

Statistical Learning Theory

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

- Measuring the optimality
  - Loss function

$$L(y, f(\mathbf{x};\alpha))$$

  - Example: quadratic loss

$$L(y, f(\mathbf{x};\alpha)) = (y - f(\mathbf{x};\alpha))^2$$

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)

$$R_{emp}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i; \alpha))$$

  - Example: quadratic loss function

$$R_{emp}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i; \alpha))^2$$

Risk

- However, what we're really interested in is
  - **Actual risk** (= **Expected risk**)

$$R(\alpha) = \int L(y, f(\mathbf{x}; \alpha)) dP_{X,Y}(\mathbf{x}, y)$$

  - $P_{X,Y}(\mathbf{x}, y)$ is the probability distribution of $(\mathbf{x}, y)$.
  - $P_{X,Y}(\mathbf{x}, y)$ is fixed, but typically unknown.

  ⇒ 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 $P_{X,Y}(\mathbf{x}, y)$
- Empirical risk
  - Disadvantage: no direct measure of the generalization ability
  - Advantage: does not depend on $P_{X,Y}(\mathbf{x}, y)$
  - We typically know learning algorithms which minimize the empirical risk.

$\Rightarrow$ Strong interest in connection between both types of risk

Statistical Learning Theory

- **Idea**
  - Compute an upper bound on the actual risk based on the empirical risk

$$R(\alpha) \cdot \quad R_{emp}(\alpha) + \epsilon(N, p^*, h)$$

  - where

    N: number of training examples

    p^*: probability that the bound is correct

    h: capacity of the learning machine ("VC-dimension")

- **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 ℓ points can be labeled in all possible 2^ℓ 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 $\{f(\alpha)\}$.*

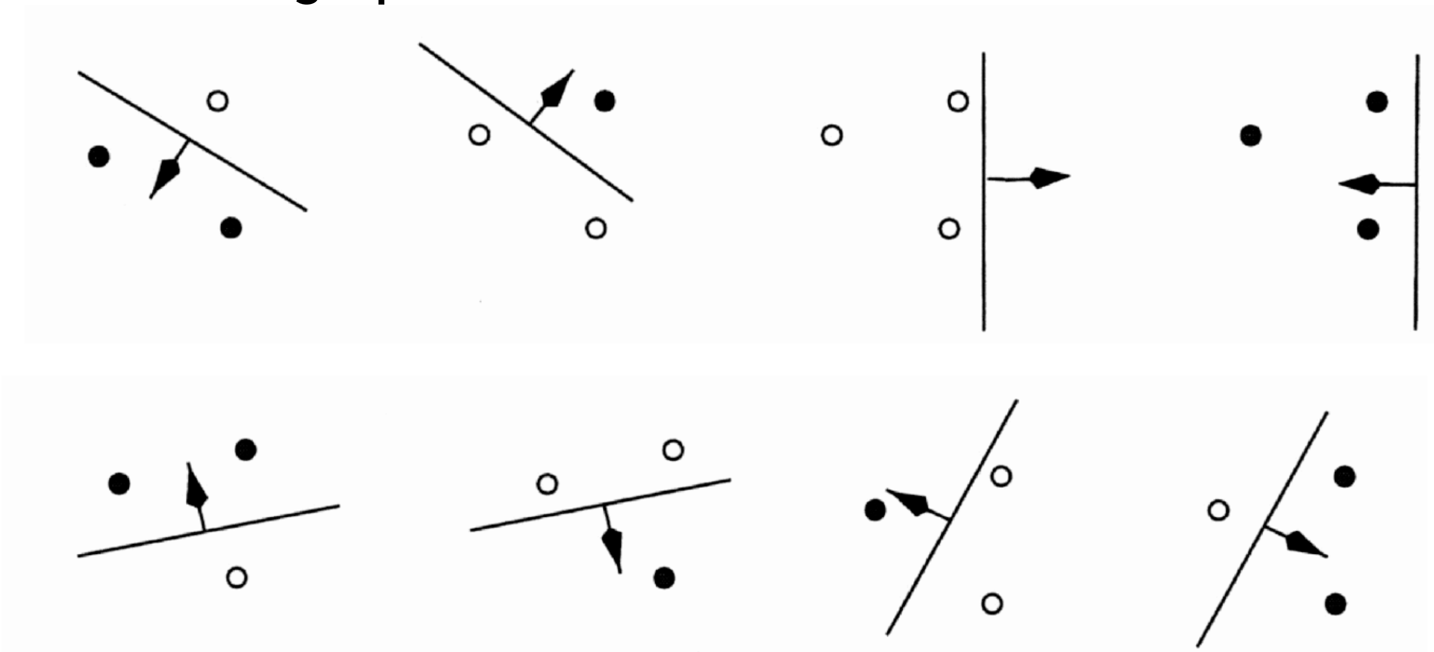
VC Dimension

- Interpretation as a two-player game
  - Opponent's turn: He says a number N.
  - Our turn: We specify a set of N points $\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$.
  - Opponent's turn: He gives us a labeling $\{\mathbf{x}_1,\ldots,\mathbf{x}_N\} \in \{0,1\}^N$
  - Our turn: We specify a function $f(\alpha)$ which correctly classifies all N points.

$\Rightarrow$ If we can do that for all 2^N possible labelings, then the VC dimension is at least N.

VC Dimension

- **Example**
  - The VC dimension of all oriented lines in $\mathbb{R}^2$ is 3.
    1. Shattering 3 points with an oriented line:



    2. More difficult to show: it is not possible to shatter 4 points (XOR)…
  - More general: the VC dimension of all hyperplanes in $\mathbb{R}^n$ is $n+1$.

Image source: C. Burges, 1998

VC Dimension

- **Intuitive feeling (unfortunately wrong)**
  - **The VC dimension has a direct connection with the number of parameters.**

- **Counterexample**

$$f(x;\alpha) = g(\sin(\alpha x))$$

$$g(x) = \begin{cases} 1, & x > 0 \\ -1, & x \cdot \ 0 \end{cases}$$

  - **Just a single parameter α.**
  - **Infinite VC dimension**
    - **Proof: Choose** $x_i = 10^{-i}, \quad i = 1, \ldots, \ell$

$$\alpha = \pi \left(1 + \sum_{i=1}^{\ell} \frac{(1 - y_i)10^i}{2} \right)$$

Upper Bound on the Risk

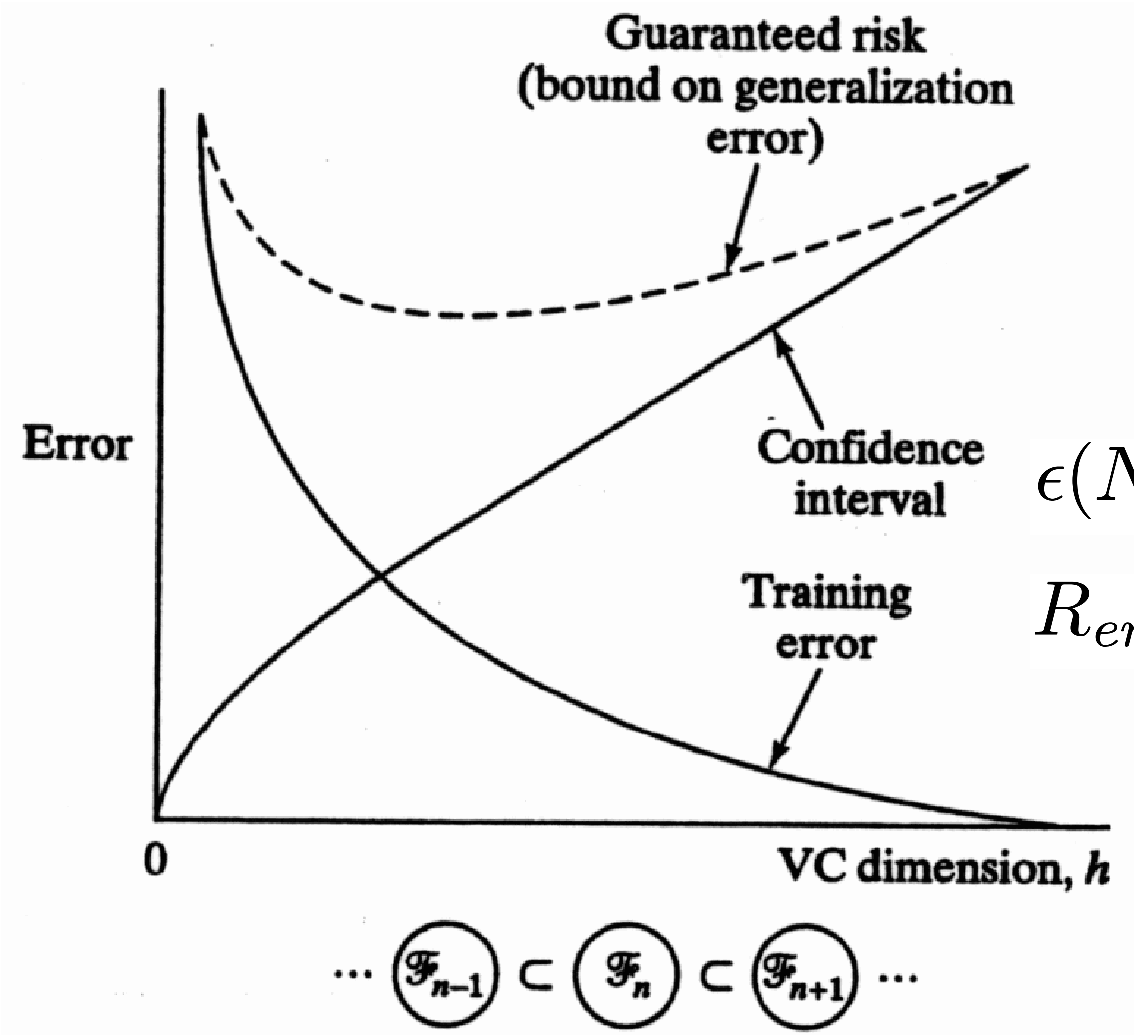
- **Important result (Vapnik 1979, 1995)**
  - **With probability $(1\text{-}\eta)$, the following bound holds**

$$R(\alpha) \cdot \quad R_{emp}(\alpha) + \underbrace{\sqrt{\frac{h(\log(2N/h)+1) - \log(\eta/4)}{N}}}_{\text{“VC confidence”}}$$

  - **This bound is independent of $P_{X,Y}(\mathbf{x}, y)$!**
  - **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**

$$R(\alpha) \cdot \quad R_{emp}(\alpha) + \epsilon(N, p^*, h)$$

Upper Bound on the Risk



$\epsilon(N, p^*, h)$

$R_{emp}(\alpha)$

Structural Risk Minimization

- **How can we implement this?**

$$R(\alpha) \cdot \quad R_{emp}(\alpha) + \epsilon(N, p^*, h)$$

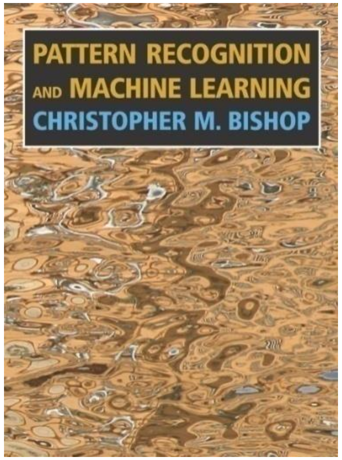
- **Classic approach**
  - **Keep** $\epsilon(N, p^*, h)$ **constant and minimize** $R_{emp}(\alpha)$ **.**
  - $\epsilon(N, p^*, h)$ **can be kept constant by controlling the model parameters.**

- **Support Vector Machines (SVMs)**
  - **Keep** $R_{emp}(\alpha)$ **constant and minimize** $\epsilon(N, p^*, h)$ **.**
  - **In fact:** $R_{emp}(\alpha) = 0$ **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.

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



- Additional information about Statistical Learning Theory and a more in-depth introduction to SVMs are available in the following tutorial:
  - C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, Vol. 2(2), pp. 121-167 1998.