Machine Learning

Linear Discriminant Functions II - Lecture VI

Course Outline

Basic Concepts

- Parametric Method,
- Bayesian Learning and Nonparametrics Methods
- Clustering and Mixture of Gaussians

Classification Approaches

- Linear Discriminants
- Ensemble Methods and Boosting
- Randomized Trees, Forest

Deep Learning

- Foundations
- Optimization

Reinforcement Learning

- Classical Reinforcement Learning
- Deep Reinforcement Learning

Videos for This Lecture

- Repetition Video (Part 0)
- Gradient descent (Part 1)
- Logistic regression (Part 2)
- Note on error function (Part 3)

Today's Topics

Gradient Descent

Logistic Regression

- Probabilistic discriminative models
- Logistic sigmoid (logit function)
- Cross-entropy error
- Iteratively Reweighted Least Squares

Note on Error Functions

- Ideal error function
- Quadratic error
- Cross-entropy error

Slide source: B. Leibe Slides ML, 2015

Recap: Learning Discriminant Functions

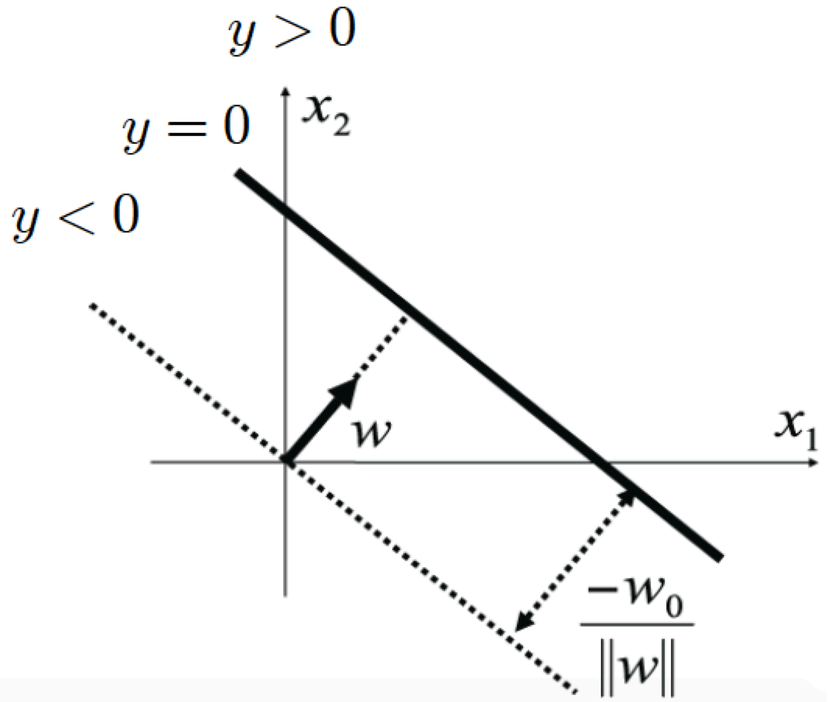
Basic idea

- Directly encode decision boundary
- Minimize misclassification probability directly

Linear discriminant functions

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

weight vector — "bias" (=threshold)



- 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**.

Slide source: B. Leibe Slides ML, 2015

Recap: Least-Squares Classification

Simplest Approach

- Directly try to minimise the **sum-of-squares error**

$$E(\mathrm{w}) = \sum_{n=1}^{N} (y(\mathrm{x}_n; \mathrm{w}) - \mathrm{t}_n)^2 \qquad E_D(\tilde{\mathrm{W}}) = \frac{1}{2}\mathrm{Tr}\{(\tilde{\mathrm{X}}\tilde{\mathrm{W}} - \mathrm{T})^T)(\tilde{\mathrm{X}}\tilde{\mathrm{W}} - \mathrm{T})\}$$

- Setting the derivative to zero yields

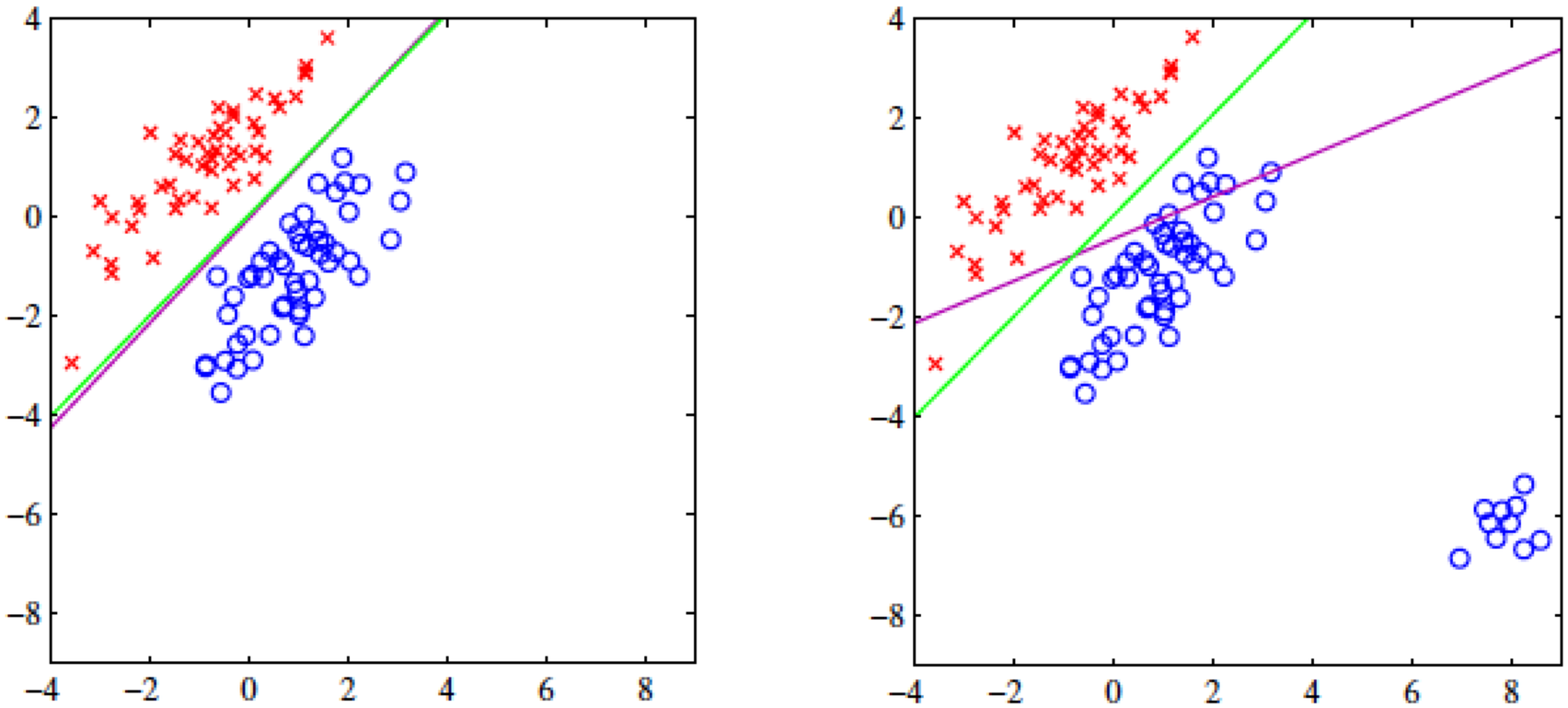
$$\tilde{\mathrm{W}} = (\tilde{\mathrm{X}}^T\tilde{\mathrm{X}})^{-1}\tilde{\mathrm{X}}^T\mathrm{T} = \tilde{\mathrm{X}}^+\mathrm{T} = (\tilde{\mathrm{X}}^T\tilde{\mathrm{X}})^{-1}\tilde{\mathrm{X}}^T\mathrm{T}$$

- We then obtain the discriminant function as

$$\mathrm{y}(\mathrm{x}) = \tilde{\mathrm{W}}^T\tilde{\mathrm{x}} = \mathrm{T}^T(\tilde{\mathrm{X}}^+)^T\tilde{\mathrm{x}}$$

➡Exact, closed-form solution for the discriminant function parameters.

Recap: Problems with Least Squares



Least-squares is very sensitive to outliers!

- The error function penalises predictions that are “too correct”

Slide source: B. Leibe Slides ML, 2015

Recap: Generalized Linear Models

Generalized linear model $y(\mathrm{x}) = g(\mathrm{w}^T\mathrm{x} + w_0)$

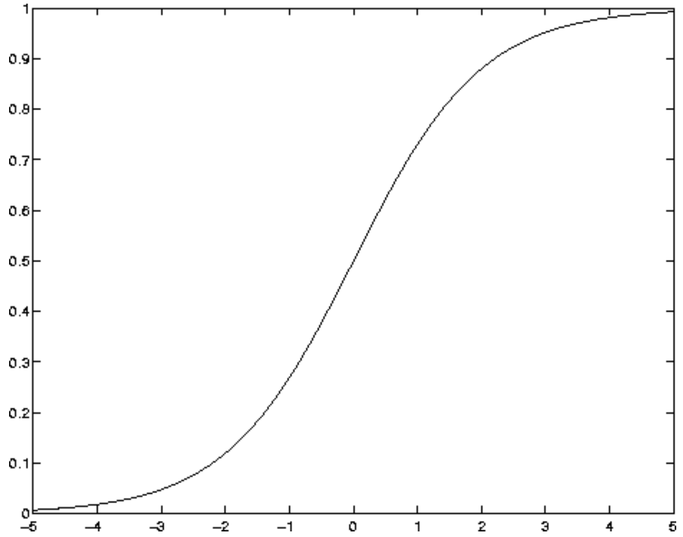
- *g(.)* is called an **activation function** and maybe nonlinear.
- The decision surface correspond to

$$y(\mathrm{x}) = const. \Leftrightarrow \mathrm{w}^T\mathrm{x} + w_0 = const.$$

- If *g* is monotonous (which is typically the case), the resulting decision boundaries are still linear functions of x.

Advantages of the non-linearity

- Can be used to bound the influence of outliers and "too correct" data points.
- When using a sigmoid for g(.), we can interpret the y(x) as posterior probabilities.



$$g(a) = \frac{1}{1+exp(-a)}$$

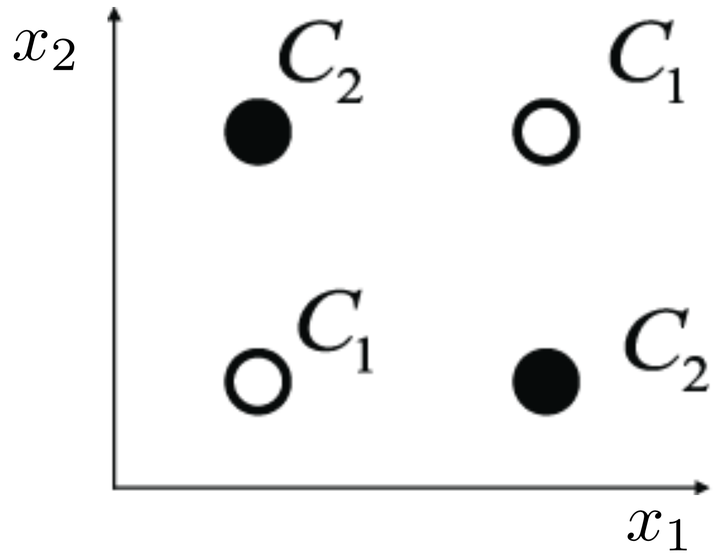
Slide source: B. Leibe Slides ML, 2015

Recap: Linear Separability

Up to now: restrictive assumption

- Only consider linear decision boundaries

Classical counterexample: XOR



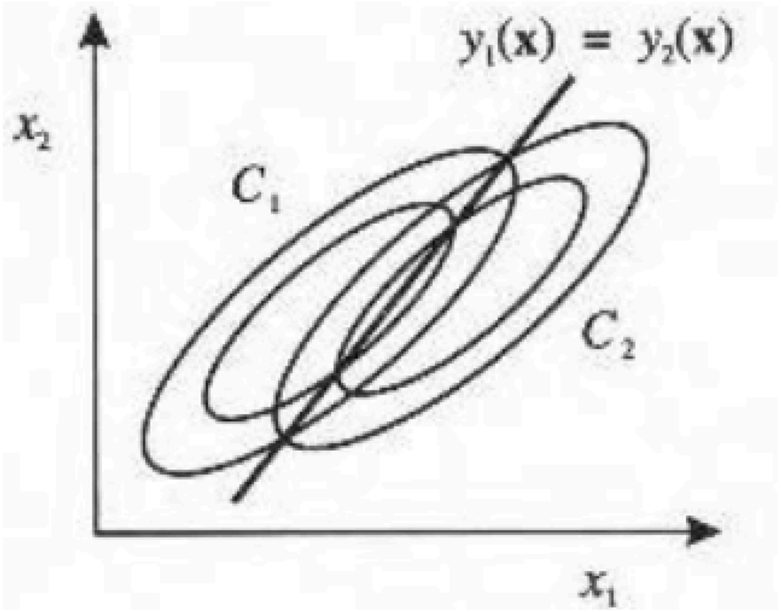
Linear Separability

Even if the data is not linearly separable, a linear decision boundary may still be „optimal“.

- Generalization
- E.g. in the case of Normal distributed data (with equal covariance matrices)

Choice of the right discriminant function is important and should be based on

- Prior knowledge (of the general functional form)
- Empirical comparison of alternative models
- Linear discriminants are often used as banchmark.



Slide source: B. Leibe Slides ML, 2015

Generalized Linear Discriminants

Generalization

- Transform vector x with M nonlinear basis functions $\phi_j(\mathrm{x})$:

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

- Purpose of $\phi_j(\mathrm{x})$ basis functions
- Allow non-linear decision boundaries.
- By choosing the right ϕ_j every continuous function can (in principle) be approximated with arbitrary accuracy.

Notation

$$y_k(\mathrm{x}) = \sum_{j=0}^{M} w_{kj}\phi_j(\mathrm{x}) \text{ with } \phi_0(\mathrm{x}) = 1$$

Slide source: B. Leibe Slides ML, 2015

Generalized Linear Discriminants

Model

$$y_k(\mathrm{x}) = \sum_{j=0}^{M} w_{kj}\phi_j(\mathrm{x}) = y_k(\mathrm{x};\mathrm{w})$$

K functions (outputs)

Learning in Neural Networks

- Single-layer networks: ϕ_j are fixed, only weights w are learned.
- Multi-layer networks: both the w and the ϕ_j are learned.

Slide source: B. Leibe Slides ML, 2015

Part 1, Video LinearDiscriminantFunctionII_p1

- Gradient Descent
- Summary: properties and limitations

Today's Topics

Gradient Descent

Logistic Regression

- Probabilistic discriminative models
- Logistic sigmoid (logit function)
- Cross-entropy error
- Iteratively Reweighted Least Squares

Note on Error Functions

- Ideal error function
- Quadratic error
- Cross-entropy error

Slide source: B. Leibe Slides ML, 2015

Gradient Descent

Learning the weights w:

- N training data points: $\mathrm{X} = \{\mathrm{x}_1, ..., \mathrm{x}_N\}$
- K outputs of decision functions: $y_k(\mathrm{x}_n; \mathrm{w})$
- Target vector for each data point: $\mathrm{T} = \{\mathrm{t}_1, ..., \mathrm{t}_N\}$
- Error function (least-squares error) of linear model

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\left(y_k(\mathbf{x}_n;\mathbf{w}) - t_{kn}\right)^2$$

$$= \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\left(\sum_{j=1}^{M} w_{kj}\phi_j(\mathbf{x}_n) - t_{kn}\right)^2$$

Slide source: B. Leibe Slides ML, 2015

Gradient Descent

Problem

- The error function can in general no longer be minimized in closed form.

Idea (Gradient Descent)

- Use interactive minimisation technique

Slide source: B. Leibe Slides ML, 2015

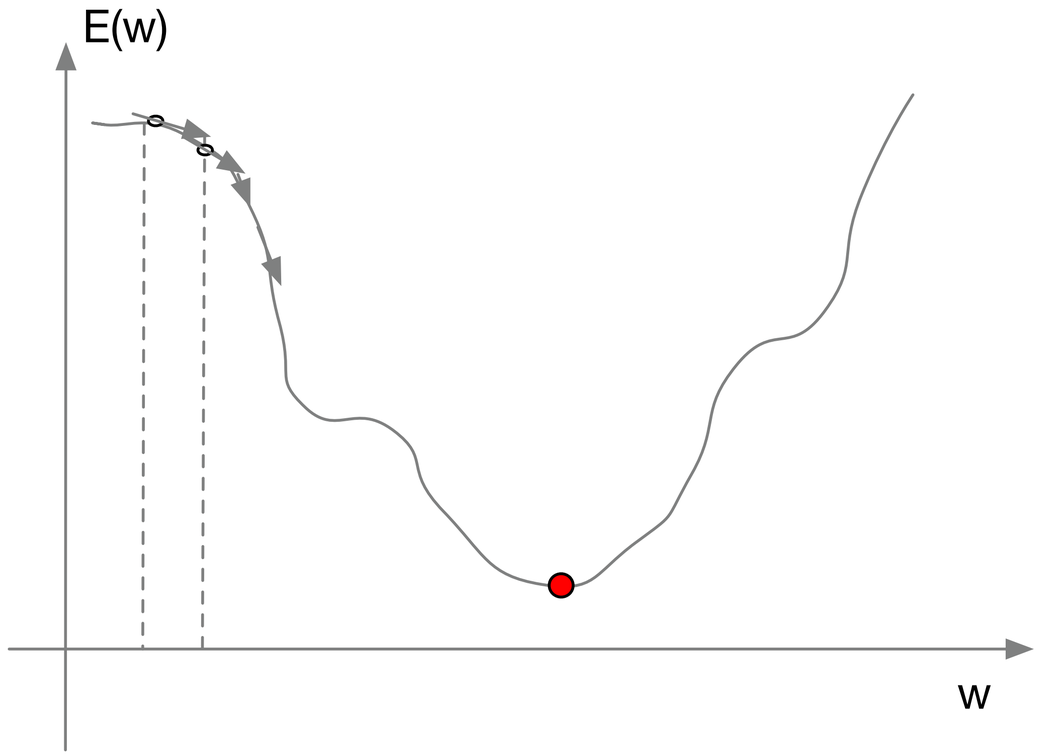
Gradient Descent

Idea(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.

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

η: Learning rate



- This simple scheme corresponds to a 1st-order Taylor expansion (There are more complex procedures available).

Slide source: B. Leibe Slides ML, 2015

Gradient Descent - Basic Strategies

"Batch learning"

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

η: Learning rate

- Compute the gradient based on all training data:

$$\frac{\partial E(\mathbf{w})}{\partial w_{kj}}$$

Slide source: B. Leibe Slides ML, 2015

Gradient Descent - Basic Strategies

"Sequential updating"

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

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

η: Learning rate

- Compute the gradient based on a single data point at a time:

$$\frac{\partial E(\mathbf{w})}{\partial w_{kj}}$$

Slide source: B. Leibe Slides ML, 2015

Gradient Descent

Error function

$$
\begin{aligned}
E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}) &= \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} \left(\sum_{j=1}^{M} w_{kj} \phi_j(\mathbf{x}_n) - t_{kn} \right)^2 \\
E_n(\mathbf{w}) &= \frac{1}{2} \sum_{k=1}^{K} \left(\sum_{j=1}^{M} w_{kj} \phi_j(\mathbf{x}_n) - t_{kn} \right)^2 \\
\frac{\partial E_n(\mathbf{w})}{\partial w_{kj}} &= \left(\sum_{\tilde{j}=1}^{M} w_{k\tilde{j}} \phi_{\tilde{j}}(\mathbf{x}_n) - t_{kn} \right) \phi_j(\mathbf{x}_n) \\
&= \left(y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn} \right) \phi_j(\mathbf{x}_n)
\end{aligned}
$$

Slide source: B. Leibe Slides ML, 2015

Gradient Descent

Delta rule (=LMS rule)

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

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

where

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

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

Slide source: B. Leibe Slides ML, 2015

Gradient Descent

Cases with differentiable, non-linear activation function

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

Gradient descent

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

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

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

Slide source: B. Leibe Slides ML, 2015

Summary: Generalized Linear Discriminants

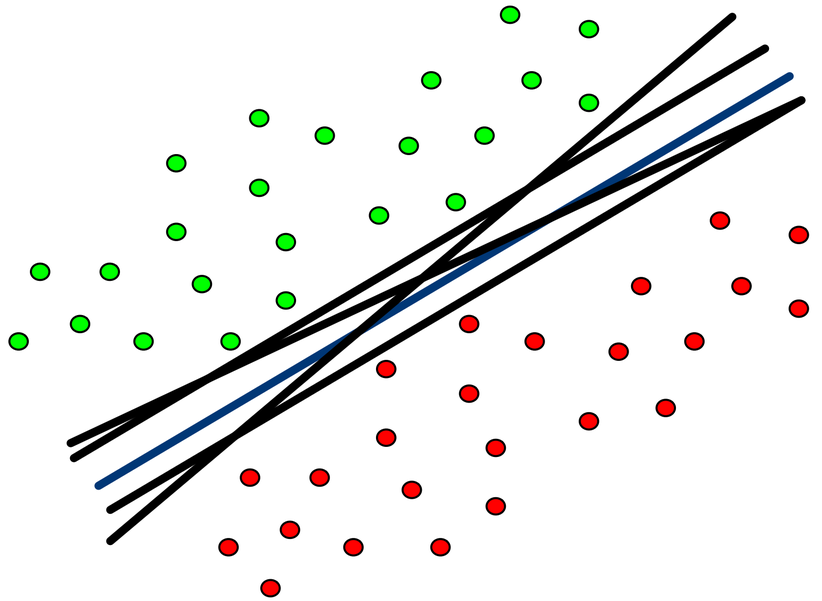
Properties

- General class of decision functions.
- Nonlinearity g(.) and basis functions ϕ_j allow us to address linearly non-separable problems.
- Shown simple sequential learning approach for parameter estimation using gradient descent.
- Better 2nd oder gradient descent approach available (e.g. Newton-Raphson)

Summary: Generalized Linear Discriminants

Limitations / Caveats

- Flexibility of model is limited by curse of dimensionality
  - g(.) and ϕ_j often introduce additional parameters.
  - Models are either limited to lower-dimensional input space or need to share parameters
- Linearly separable case often leads to overfitting.
  - Several possible parameter choices minimise training error



Slide source: B. Leibe Slides ML, 2015

Part 2, Video LinearDiscriminantFunctionII_p2

- Probabilistic discriminative models
- Logistic sigmoid (logit function)
- Cross-entropy error
- Iteratively Reweighted Least Squares

Today's Topics

Gradient Descent

Logistic Regression

- Probabilistic discriminative models
- Logistic sigmoid (logit function)
- Cross-entropy error
- Iteratively Reweighted Least Squares

Note on Error Functions

- Ideal error function
- Quadratic error
- Cross-entropy error

Slide source: B. Leibe Slides ML, 2015

Probabilistic Discriminative Models

We have seen that we can write $p(C_1|\mathrm{x}) = \sigma(a)$

Logistic sigmoid function

$= \frac{1}{1+exp(-a)}$

We can obtain the familiar probabilistic model by setting

$$a = \ln \frac{p(\mathrm{x}|C_1)p(C_1)}{p(\mathrm{x}|C_2)p(C_2)}$$

Or we can use generalised linear discriminant models

$$a = w^T x$$

$$a = w^T \phi(x)$$

Probabilistic Discriminative Models

In the following, we will consider models of the form

$$p(C_1|\phi) = y(\phi) = \sigma(\mathrm{w}^T\phi)$$
$$p(C_2|\phi) = 1 - p(C_1|\phi)$$

This model is called **logistic regression**.

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

$$p(C_1|\phi) = \frac{p(\phi|C_1)p(C_1)}{p(\phi|C_1)p(C_1)+p(\phi|C_2)p(C_2)}$$

Any ideas?

Comparison

Let's look at the number of parameters…

- Assume we have M-dimensional feature space ϕ
- And assume we represent $p(\phi|C_k)$ and $p(C_k)$ 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 regressions!

$$p(C_1|\phi) = y(\phi) = \sigma(\mathrm{w}^T\phi)$$

  - Just the values of **w** => *M* parameters

=> For large M, logistic regression has clear advantages!

Logistic Sigmoid

Properties

- Definition: $\sigma(a) = \frac{1}{1+exp(-a)}$
- Inverse: $a = \ln(\frac{\sigma}{1-\sigma})$ “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 $n = 1, ..., N$,
where $\phi_n = \phi(\mathrm{x}_n)$ and $t_n \in \{0, 1\}, \mathrm{t} = (t_1, ..., t_N)^T$

With $y_n = p(C_1|\phi_n)$, we can write likelihood as

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

$$p(t\,|\,w) = \prod_{n=1}^{N} p(C_1\,|\,x_n)^{t_n}(1 - p(C_1\,|\,x_n))^{1-t_n}$$

$p(C_2\,|\,x_n)$

$$= \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}, \text{ where } y_n = p(C_1\,|\,x_n)$$

Logistic Regression

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

With $y_n = p(C_1|\phi_n)$, we can write likelihood as

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

Define the error function as the negative log-likelihood

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

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

Gradient of the Error Function

Error function

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

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

Gradient

$$\nabla E(\mathrm{w}) = -\sum_{n=1}^{N}\{t_n \frac{\frac{d}{d\mathrm{w}}y_n}{y_n} + (1-t_n)\frac{\frac{d}{d\mathrm{w}}(1-y_n)}{(1-y_n)}\}$$

$$= -\sum_{n=1}^{N}\{t_n \frac{y_n(1-y_n)}{y_n}\phi_n - (1-t_n)\frac{y_n(1-y_n)}{1-y_n}\phi_n\}$$

$$= -\sum_{n=1}^{N}\{(t_n - t_n y_n - y_n + t_n y_n)\phi_n\}$$

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

Gradient of the Error Function

Gradient for logistic regression

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

Does it 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

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

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

Properties:

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

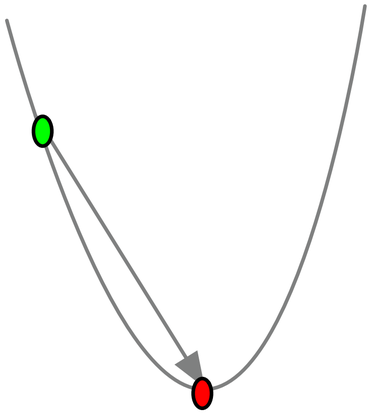
Newton-Raphson for Least-Squares Estimation

Let's first apply Newton-Raphson method to the least-square error function

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

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

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N}\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T = \boldsymbol{\Phi}^T\boldsymbol{\Phi} \qquad \text{where} \quad \boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1^T \\ \vdots \\ \boldsymbol{\phi}_N^T \end{bmatrix}$$



Resulting update scheme:

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

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

Closed-form solution!

Newton-Raphson for Least-Squares Estimation

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

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

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

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

$$\frac{dy_n}{d\mathbf{w}} = y_n(1-y_n)\boldsymbol{\phi}_n$$

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 **w** weighting matrix **R**.

Iteratively Reweighted Least Squares

Update equations

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

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

Again very similar form (normal equations)

- But now with non-constant weighting metric **R** (depends on **w**).
- Need to apply normal equations iteratively.
- Iteratively Reweighted Least-Squares (IRLS)

Summary: Logistic Regression

Properties

- Directly represent posterior distribution $p(\phi|C_k)$
- Requires fewer parameters than modelling 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 exist
  - Iterative optimisation (IRLS)
- Both online and batch optimizations exist

Limitations / Caveats

- Logistic regression tends to systematically overestimate odds ratios when the sample size is less than ~ 500

Part 3, Video LinearDiscriminantFunctionII_p3

- Ideal error function
- Quadratic error
- Cross-entropy error

Today's Topics

Gradient Descent

Logistic Regression

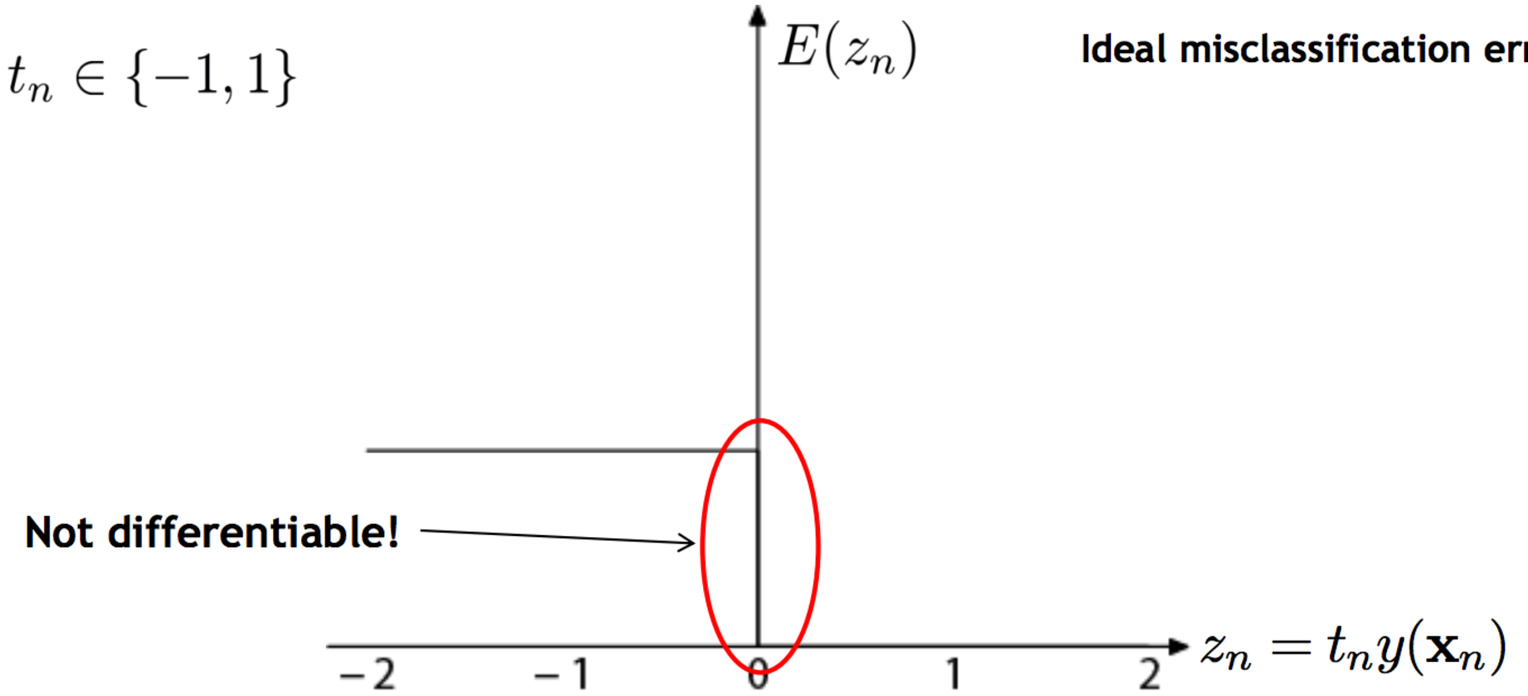
- Probabilistic discriminative models
- Logistic sigmoid (logit function)
- Cross-entropy error
- Iteratively Reweighted Least Squares

Note on Error Functions

- Ideal error function
- Quadratic error
- Cross-entropy error

Slide source: B. Leibe Slides ML, 2015

Note on Error Functions



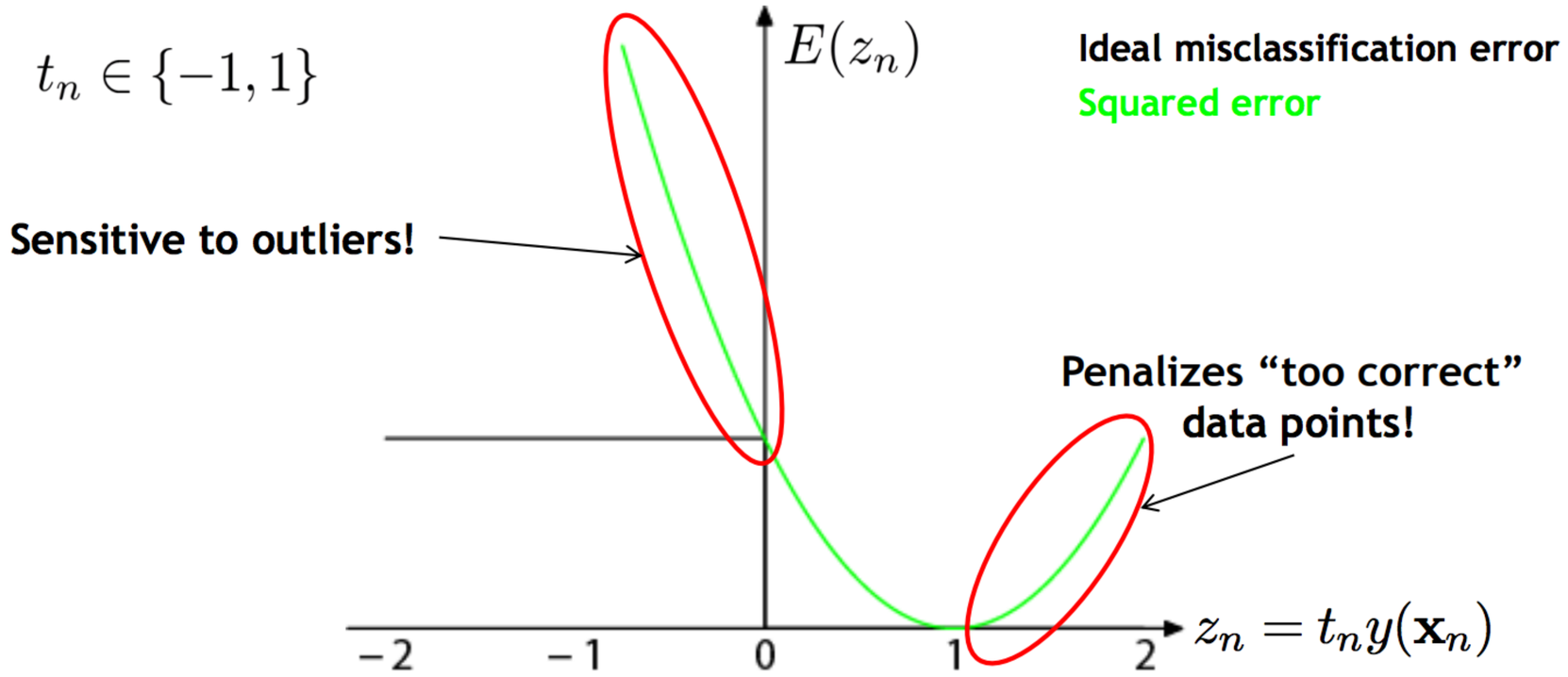
Ideal misclassification error function (black)

- 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.

Slide source: B. Leibe Slides ML, 2015

Note on Error Functions



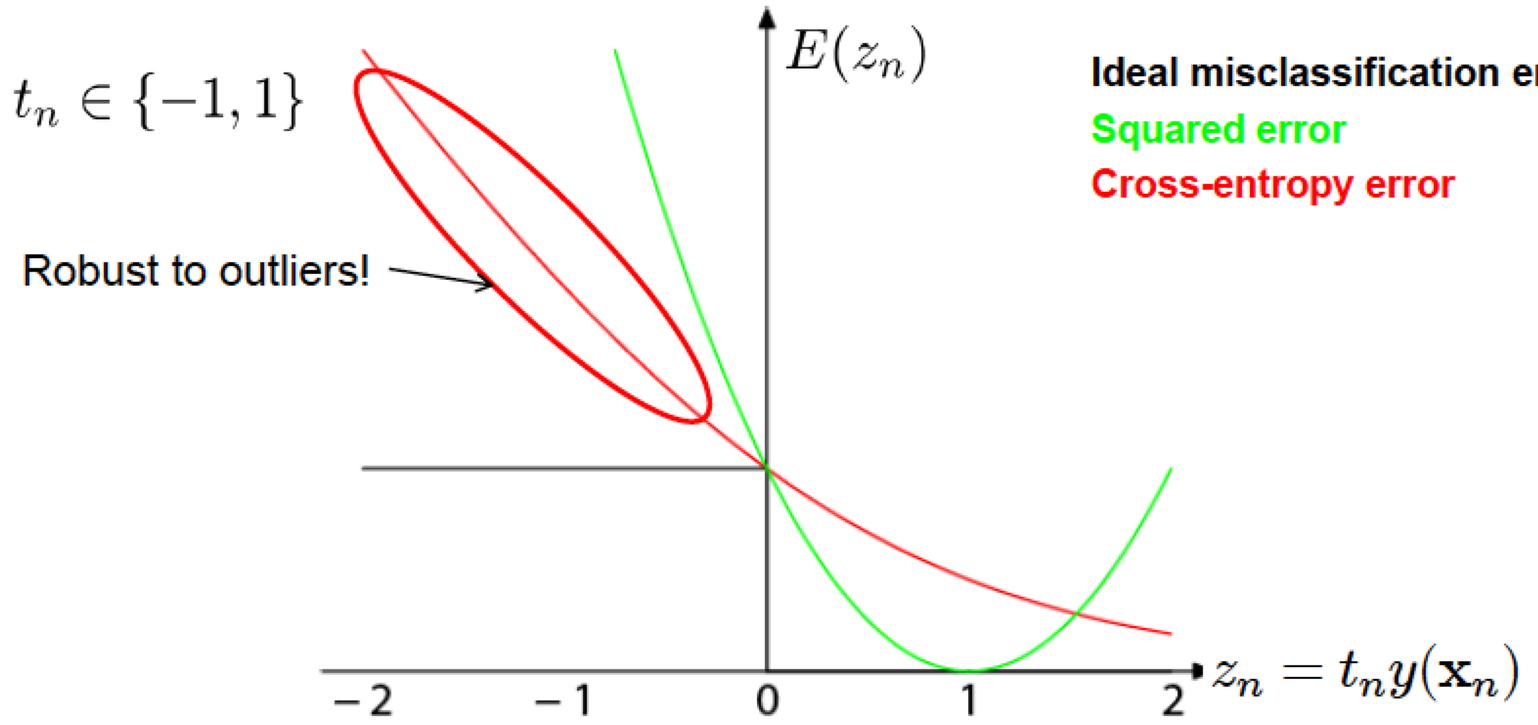
Squared error used in Least-Squares Classification

- 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.

Slide source: B. Leibe Slides ML, 2015

Note on Error Functions



Cross-Entropy Error

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

Slide source: B. Leibe Slides ML, 2015

Overview: Error Functions

Ideal Misclassification Error

- This is what we would like to optimize.
- But cannot compute gradients here

Quadratic Error

- Easy to optimize, closed-form solutions exist.
- But not robust to outliers.

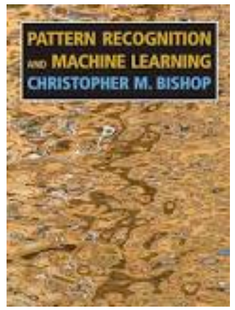
Cross-Entropy Error

- Minimizer of this function is given by posterior class probabilities.
- Concave error function, unique minimum exists.
- But no closed-form solution, requires iterative estimation.

=> Analysis tool to compare classification approaches

Readings

Bishop’s book



More information on Linear Discriminant Functions can be found in Chapter 4 of Bishop’s book (in particular Chapter 4.1).

Slide is taken from B. Leibe, ML, 2017

You should be able to answer …

- Which type of generalisation can be added to linear discriminant?
- What are basis functions?
- What is idea behind gradient descent?
- Batch learning vs. Sequential update
- What is delta rule?
- Properties and limitations of generalised linear discriminant functions.
- Sigmoid function: formula, implementation, properties.
- Which model called Logistic regression?
- Which error function is used in logistic regression?
- Logistic regression properties and limitations.
- Advantages and disadvantages of quadratic error function and cross-entropy error function
- What Newton-Raphson gradient descent scheme: idea and weight update formula.