Advanced Machine Learning
Lecture 11

Linear Discriminants Revisited

21.11.2016

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

leibe@vision.rwth-aachen.de

Talk Announcement

- **Yann LeCun (FaceBook AI)**
  28.11. 15:00-16:30h, SuperC 6th floor

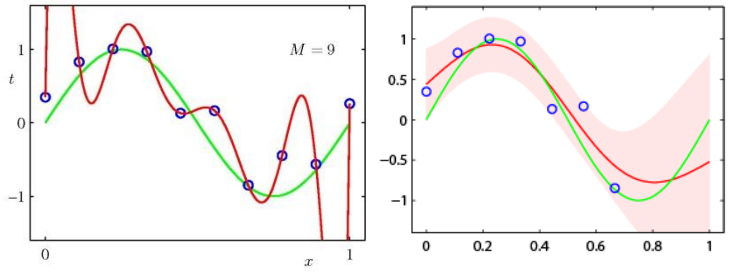
  The rapid progress of AI in the last few years are largely the result of advances in deep learning and neural nets, combined with the availability of large datasets and fast GPUs. We now have systems that can recognize images with an accuracy that rivals that of humans. This will lead to revolutions in several domains such as autonomous transportation and medical image analysis. But all of these systems currently use supervised learning in which the machine is trained with inputs labeled by humans. The challenge of the next several years is to let machines learn from raw, unlabeled data, such as video or text. This is known as predictive (or unsupervised) learning. Intelligent systems today do not possess "common sense", which humans and animals acquire by observing the world, by acting in it, and by understanding the physical constraints of it. I will argue that the ability of machines to learn predictive models of the world is a key component of that will enable significant progress in AI. The main technical difficulty is that the world is only partially predictable. A general formulation of unsupervised learning that deals with partial predictability will be presented. The formulation connects many well-known approaches to unsupervised learning, as well as new and exciting ones such as adversarial training.

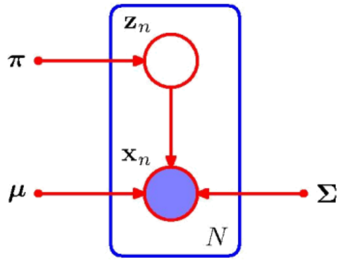
- **No lecture next Monday - go see the talk!**

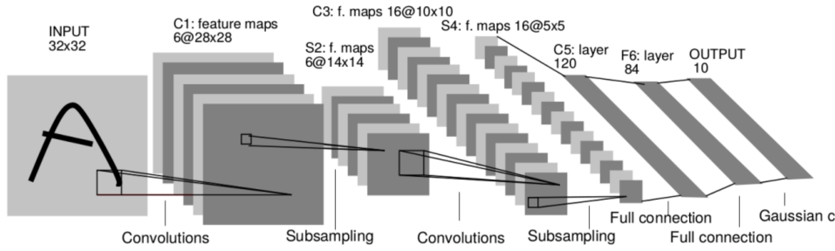
This Lecture: *Advanced Machine Learning*

- Regression Approaches
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
  - Gaussian Processes
- Approximate Inference
  - Sampling Approaches
  - MCMC
- Deep Learning
  - Linear Discriminants
  - Neural Networks
  - Backpropagation
  - CNNs, RNNs, RBMs, etc.

$$f : \mathcal{X} \to \mathbb{R}$$







Recap: Importance Sampling

- Approach
  - Approximate expectations directly (but does not enable to draw samples from $p(\mathbf{z})$ directly).
  - Goal: $$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z}$$
- Idea
  - Use a proposal distribution $q(\mathbf{z})$ from which it is easy to sample.
  - Express expectations in the form of a finite sum over samples $\{\mathbf{z}^{(l)}\}$ drawn from $q(\mathbf{z})$.

$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z} = \int f(\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}q(\mathbf{z})d\mathbf{z}$$

$$\simeq \frac{1}{L}\sum_{l=1}^{L}\underbrace{\frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})}}_{\text{Importance weights}}f(\mathbf{z}^{(l)})$$

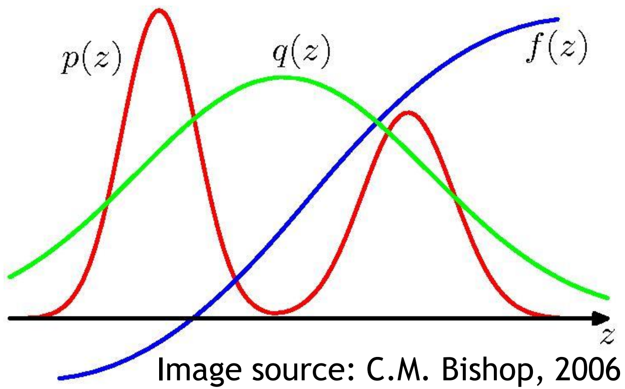


Image source: C.M. Bishop, 2006

Recap: MCMC – Markov Chain Monte Carlo

- Overview
  - Allows to sample from a large class of distributions.
  - Scales well with the dimensionality of the sample space.
- Idea
  - We maintain a record of the current state $\mathbf{z}^{(\tau)}$
  - The proposal distribution depends on the current state: $q(\mathbf{z}|\mathbf{z}^{(\tau)})$
  - The sequence of samples forms a Markov chain $\mathbf{z}^{(1)}, \mathbf{z}^{(2)},\ldots$
- Approach
  - At each time step, we generate a candidate sample from the proposal distribution and accept the sample according to a criterion.
  - Different variants of MCMC for different criteria.

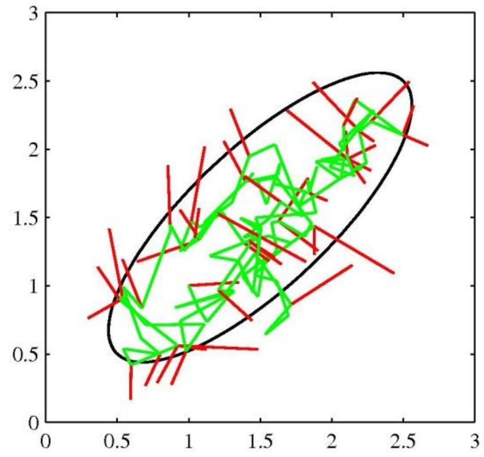


Image source: C.M. Bishop, 2006

Recap: Markov Chains - Properties

- **Invariant distribution**
  - A distribution is said to be invariant (or stationary) w.r.t. a Markov chain if each step in the chain leaves that distribution invariant.
  - Transition probabilities:

$$T\left(\mathbf{z}^{(m)}, \mathbf{z}^{(m+1)}\right) = p\left(\mathbf{z}^{(m+1)} | \mathbf{z}^{(m)}\right)$$

  - For homogeneous Markov chain, distribution $p^{*}(\mathbf{z})$ is invariant if:

$$p^{\star}(\mathbf{z}) = \sum_{\mathbf{z}'} T(\mathbf{z}', \mathbf{z})\, p^{\star}(\mathbf{z}')$$

- **Detailed balance**
  - Sufficient (but not necessary) condition to ensure that a distribution is invariant:

$$p^{\star}(\mathbf{z}) T(\mathbf{z}, \mathbf{z}') = p^{\star}(\mathbf{z}') T(\mathbf{z}', \mathbf{z})$$

  - A Markov chain which respects *detailed balance* is reversible.

Slide credit: Bernt Schiele

Recap: MCMC – Metropolis Algorithm

- **Metropolis algorithm** [Metropolis et al., 1953]
  - Proposal distribution is symmetric: $q(\mathbf{z}_A|\mathbf{z}_B) = q(\mathbf{z}_B|\mathbf{z}_A)$
  - The new candidate sample $\mathbf{z}^*$ is accepted with probability

$$A(\mathbf{z}^\star, \mathbf{z}^{(\tau)}) = \min\left(1, \frac{\tilde{p}(\mathbf{z}^\star)}{\tilde{p}(\mathbf{z}^{(\tau)})}\right)$$

  ⇒ New candidate samples always accepted if $\tilde{p}(\mathbf{z}^\star) \geq \tilde{p}(\mathbf{z}^{(\tau)})$.
  - The algorithm sometimes accepts a state with lower probability.

- **Metropolis-Hastings algorithm**
  - Generalization: Proposal distribution not necessarily symmetric.
  - The new candidate sample $\mathbf{z}^*$ is accepted with probability

$$A(\mathbf{z}^\star, \mathbf{z}^{(\tau)}) = \min\left(1, \frac{\tilde{p}(\mathbf{z}^\star) q_k(\mathbf{z}^{(\tau)}|\mathbf{z}^\star)}{\tilde{p}(\mathbf{z}^{(\tau)}) q_k(\mathbf{z}^\star|\mathbf{z}^{(\tau)})}\right)$$

  - where k labels the members of the set of considered transitions.

Random Walks

- Example: Random Walk behavior
  - Consider a state space consisting of the integers $z \in \mathbb{Z}$ with initial state $z(1) = 0$ and transition probabilities

$$\begin{aligned} p(z^{(\tau+1)} = z^{(\tau)}) &= 0.5 \\ p(z^{(\tau+1)} = z^{(\tau)} + 1) &= 0.25 \\ p(z^{(\tau+1)} = z^{(\tau)} - 1) &= 0.25 \end{aligned}$$

- Analysis
  - Expected state at time τ : $\mathbb{E}[z^{(\tau)}] = 0$
  - Variance: $\mathbb{E}[(z^{(\tau)})^2] = \tau/2$
  - After τ steps, the random walk has only traversed a distance that is on average proportional to $\sqrt{\tau}$.

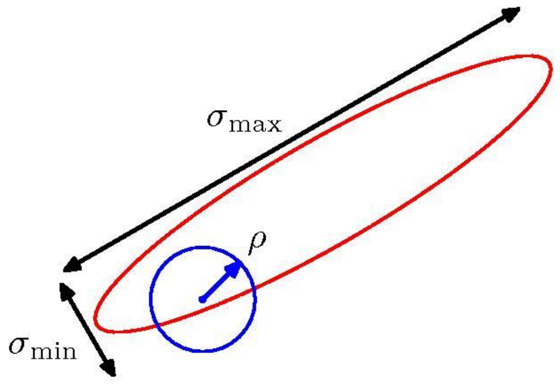
$\Rightarrow$ Central goal in MCMC is to avoid random walk behavior!

MCMC – Metropolis-Hastings Algorithm

- Schematic illustration
  - For continuous state spaces, a common choice of proposal distribution is a Gaussian centered on the current state.

  $\Rightarrow$ What should be the variance of the proposal distribution?
    - Large variance: rejection rate will be high for complex problems.
    - The scale ρ of the proposal distribution should be as large as possible without incurring high rejection rates.

    $\Rightarrow$ ρ should be of the same order as the smallest length scale $\sigma_{\min}$.



  - This causes the system to explore the distribution by means of a random walk.
    - Undesired behavior: number of steps to arrive at state that is independent of original state is of order $(\sigma_{\max}/\sigma_{\min})^2$.
    - Strong correlations can slow down the Metropolis(-Hastings) algorithm!

Image source: C.M. Bishop, 2006

Gibbs Sampling

- Approach
  - MCMC-algorithm that is simple and widely applicable.
  - May be seen as a special case of Metropolis-Hastings.
- Idea
  - Sample variable-wise: replace $\mathbf{z}_i$ by a value drawn from the distribution $p(z_i|\mathbf{z}_{\setminus i})$.
    - This means we update one coordinate at a time.
  - Repeat procedure either by cycling through all variables or by choosing the next variable.

Gibbs Sampling

- Example
  - Assume distribution $p(z_1, z_2, z_3)$.
  - Replace $z_1^{(\tau)}$ with new value drawn from $z_1^{(\tau+1)} \sim p(z_1|z_2^{(\tau)}, z_3^{(\tau)})$
  - Replace $z_2^{(\tau)}$ with new value drawn from $z_2^{(\tau+1)} \sim p(z_2|z_1^{(\tau+1)}, z_3^{(\tau)})$
  - Replace $z_3^{(\tau)}$ with new value drawn from $z_3^{(\tau+1)} \sim p(z_3|z_1^{(\tau+1)}, z_2^{(\tau+1)})$
  - And so on…

Gibbs Sampling

- Properties
  - Since the components are unchanged by sampling: $\mathbf{z}^*{}_{\backslash k} = \mathbf{z}_{\backslash k}$.
  - The factor that determines the acceptance probability in the Metropolis-Hastings is thus determined by

$$A(\mathbf{z}^\star, \mathbf{z}) = \frac{p(\mathbf{z}^\star)q_k(\mathbf{z}|\mathbf{z}^\star)}{p(\mathbf{z})q_k(\mathbf{z}^\star|\mathbf{z})} = \frac{p(z_k^\star|\mathbf{z}^\star_{\backslash k})p(\mathbf{z}^\star_{\backslash k})p(z_k|\mathbf{z}^\star_{\backslash k})}{p(z_k|\mathbf{z}_{\backslash k})p(\mathbf{z}_{\backslash k})p(z_k^\star|\mathbf{z}_{\backslash k})} = 1$$

  - (we have used $q_k(\mathbf{z}^*|\mathbf{z}) = p(z^*{}_k|\mathbf{z}_{\backslash k})$ and $p(\mathbf{z}) = p(z_k|\mathbf{z}_{\backslash k})\, p(\mathbf{z}_{\backslash k})$).
  - I.e. we get an algorithm which always accepts!

$\Rightarrow$ If you can compute (and sample from) the conditionals, you can apply Gibbs sampling.

$\Rightarrow$ The algorithm is completely parameter free.

$\Rightarrow$ Can also be applied to subsets of variables.

Slide adapted from Zoubin Ghahramani

Discussion

- Gibbs sampling benefits from few free choices and convenient features of conditional distributions:
  - Conditionals with a few discrete settings can be explicitly normalized:

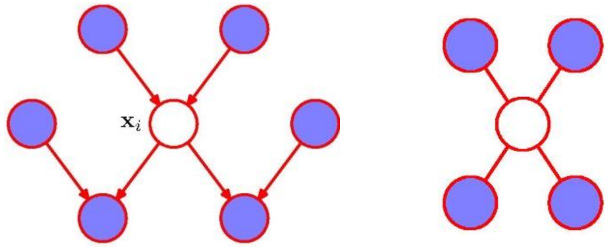
$$p(x_i|\mathbf{x}_{j\neq i}) = \frac{p(x_i, \mathbf{x}_{j\neq i})}{\sum_{x'_i} p(x'_i, \mathbf{x}_{j\neq i})}$$

This sum is small and easy.

  - Continuous conditionals are often only univariate.

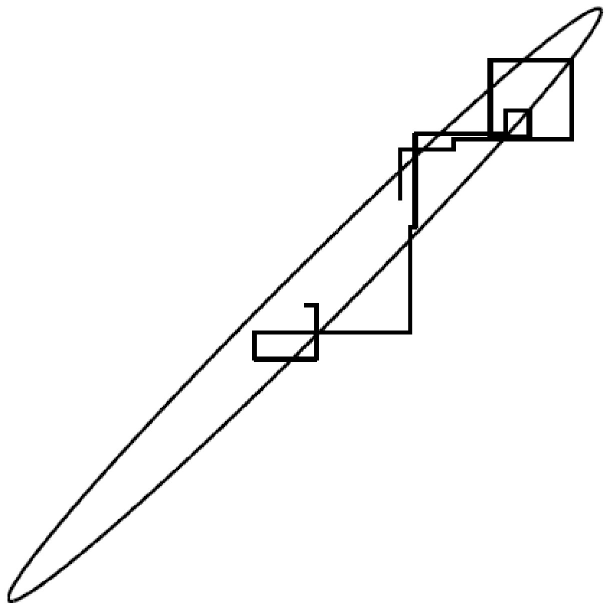
  $\Rightarrow$ amenable to standard sampling methods.

  - In case of graphical models, the conditional distributions depend only on the variables in the corresponding Markov blankets.



Gibbs Sampling

- **Example**
  - **20 iterations of Gibbs sampling on a bivariate Gaussian.**



  - **Note: strong correlations can slow down Gibbs sampling.**

Slide credit: Zoubin Ghahramani

How Should We Run MCMC?

- Arbitrary initialization means starting iterations are bad
  - Discard a "burn-in" period.
- How do we know if we have run for long enough?
  - You don't. That's the problem.
- The samples are not independent
  - Solution 1: Keep only every M^{th} sample ("thinning").
  - Solution 2: Keep all samples and use the simple Monte Carlo estimator on MCMC samples
    - It is consistent and unbiased if the chain has "burned in".

  $\Rightarrow$ Use thinning only if computing $f(\mathbf{x}^{(s)})$ is expensive.
- For opinion on thinning, multiple runs, burn in, etc.
  - Charles J. Geyer, Practical Markov chain Monte Carlo, Statistical Science. 7(4):473{483, 1992. (http://www.jstor.org/stable/2246094)

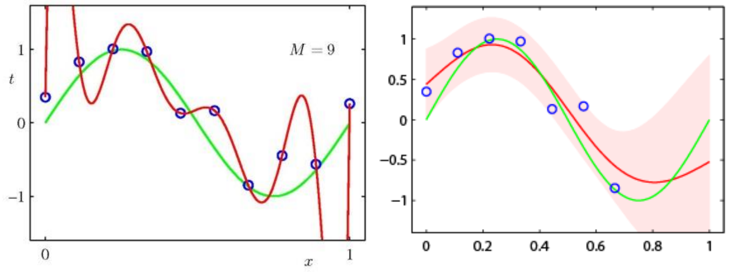
Summary: Approximate Inference

- Exact Bayesian Inference often intractable.
- Rejection and Importance Sampling
  - Generate independent samples.
  - Impractical in high-dimensional state spaces.
- Markov Chain Monte Carlo (MCMC)
  - Simple & effective (even though typically computationally expensive).
  - Scales well with the dimensionality of the state space.
  - Issues of convergence have to be considered carefully.
- Gibbs Sampling
  - Used extensively in practice.
  - Parameter free
  - Requires sampling conditional distributions.

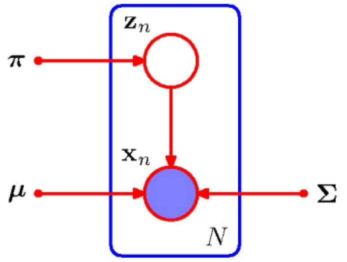
This Lecture: *Advanced Machine Learning*

$$f : \mathcal{X} \to \mathbb{R}$$

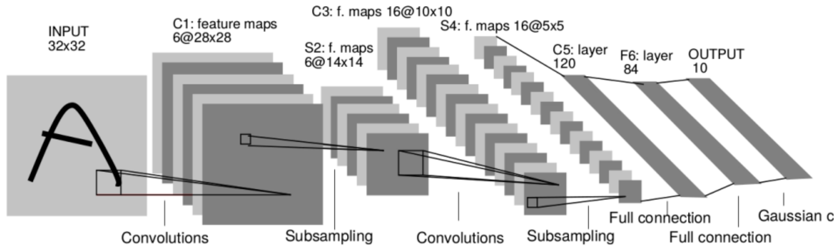
- Regression Approaches
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
  - Gaussian Processes



- Approximate Inference
  - Sampling Approaches
  - MCMC



- Deep Learning
  - Linear Discriminants
  - Neural Networks
  - Backpropagation
  - CNNs, RNNs, RBMs, etc.



We've finally got there!

Deep Learning

- We've finally got there! Yay! But...
  - What *is* it?
  - *Why* is it a thing?
  - Why is it a thing *now*?

- In order to understand that, let's look at some background first:
  - Linear Discriminants (this lecture)
  - Neural Networks
  - Backpropagation
  - How to get them to work
  - Specific types of networks (CNN, RNN, ResNets, ...)

Topics of This Lecture

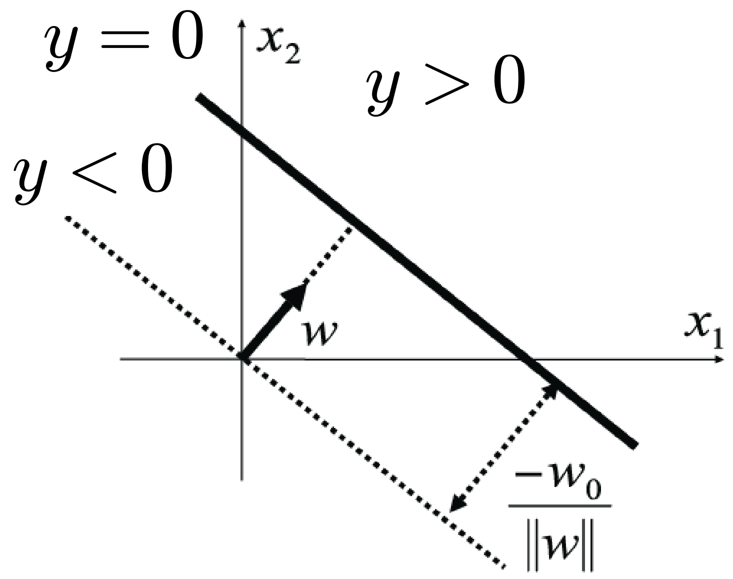
- Linear Discriminants Revisited (from ML lecture)
  - Linear Discriminants
  - Least-Squares Classification
  - Generalized Linear Discriminants
  - Gradient Descent
- Logistic Regression
  - Probabilistic discriminative models
  - Logistic sigmoid (logit function)
  - Cross-entropy error
  - Gradient descent
  - Note on error functions
- Softmax Regression
  - Multi-class generalization
  - Properties

Recap: Linear Discriminant Functions

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

- Linear discriminant functions

$$y(\mathbf{x}) = \underbrace{\mathbf{w}^{\mathrm{T}}}_{\text{weight vector}}\mathbf{x} + \underbrace{w_0}_{\text{"bias" (= threshold)}}$$



  - $\mathbf{w}$, 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: Least-Squares Classification

- Simplest approach
  - Directly try to minimize the sum-of-squares error

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

  - Setting the derivative to zero yields

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \sum_{n=1}^{N} \left(\mathbf{w}^\top \mathbf{x}_n - t_n\right) \mathbf{x}_n = \mathbf{X}\mathbf{X}^\top \mathbf{w} - \mathbf{X}\mathbf{t} \overset{!}{=} 0$$

$$\mathbf{w} = \left(\mathbf{X}\mathbf{X}^\top\right)^{-1} \mathbf{X}\mathbf{t}$$

$\Rightarrow$ Exact, closed-form solution for the parameters.

Recap: Multi-Class Case

- **General classification problem**
  - Let's consider K classes described by linear models

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}, \qquad k = 1, \ldots, K$$

  - We can group those together using vector notation

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}}$$

where

$$\widetilde{\mathbf{W}} = [\widetilde{\mathbf{w}}_1, \ldots, \widetilde{\mathbf{w}}_K] = \begin{bmatrix} w_{10} & \ldots & w_{K0} \\ w_{11} & \ldots & w_{K1} \\ \vdots & \ddots & \vdots \\ w_{1D} & \ldots & w_{KD} \end{bmatrix}$$

  - The output will again be in 1-of-K notation.

$\Rightarrow$ We can directly compare it to the target value $\mathbf{t} = [t_1, \ldots, t_k]^{\mathrm{T}}$.

Recap: Multi-Class Case

- Classification problem in matrix notation
  - For the entire dataset, we can write

$$\mathbf{Y}(\widetilde{\mathbf{X}}) = \widetilde{\mathbf{X}}\widetilde{\mathbf{W}}$$

and compare this to the target matrix $\mathbf{T}$ where

$$\widetilde{\mathbf{W}} = [\widetilde{\mathbf{w}}_1, \ldots, \widetilde{\mathbf{w}}_K]$$

$$\widetilde{\mathbf{X}} = \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_N^{\mathrm{T}} \end{bmatrix} \qquad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{t}_N^{\mathrm{T}} \end{bmatrix}$$

  - Result of the comparison:

$$\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T}$$

Goal: Choose $\widetilde{\mathbf{W}}$ such that this is minimal!

Recap: Multi-Class Least-Squares

- **Multi-class case**
  - We can formulate the sum-of-squares error in matrix notation

$$E(\widetilde{\mathbf{W}}) = \sum_{n=1}^{N}\sum_{k=1}^{K}(y(\mathbf{x}_n;\mathbf{w}_k) - t_{kn})^2$$
$$= \frac{1}{2}\mathrm{Tr}\left\{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^\top(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})\right\}$$

  - Setting the derivative to zero yields

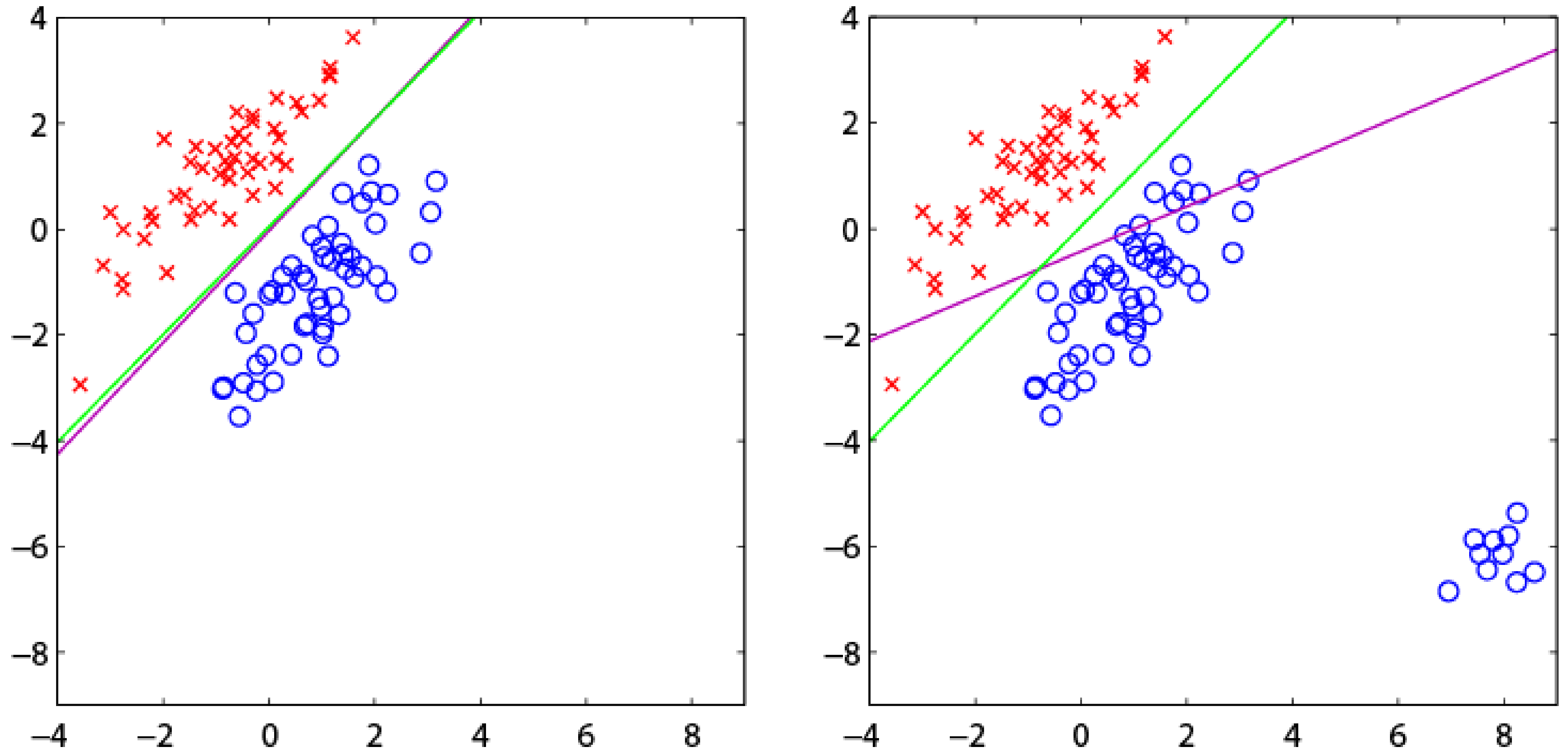
$$\widetilde{\mathbf{W}} = \widetilde{\mathbf{X}}^\dagger\mathbf{T} = (\widetilde{\mathbf{X}}^\top\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^\top\mathbf{T}$$

  - We then obtain the discriminant function as

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^\top\widetilde{\mathbf{x}} = \mathbf{T}^\top\left(\widetilde{\mathbf{X}}^\dagger\right)^\top\widetilde{\mathbf{x}}$$

$\Rightarrow$ Exact, closed-form solution for the discriminant function parameters.

Recap: Problems with Least Squares



- **Least-squares is very sensitive to outliers!**
  - **The error function penalizes predictions that are "too correct".**

Image source: C.M. Bishop, 2006

Recap: Generalized Linear Models

- Generalized linear model

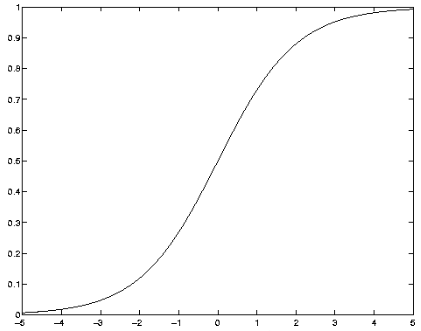
$$y(\mathbf{x}) = g(\mathbf{w}^\top \mathbf{x} + w_0)$$

  - $g(\,\cdot\,)$ is called an activation function and may be nonlinear.
  - The decision surfaces correspond to

$$y(\mathbf{x}) = const. \quad \Leftrightarrow \quad \mathbf{w}^\top \mathbf{x} + w_0 = const.$$

  - If g is monotonous (which is typically the case), the resulting decision boundaries are still linear functions of $\mathbf{x}$.

- Advantages of the non-linearity
  - Can be used to bound the influence of outliers and "too correct" data points.
  - When using a sigmoid for $g(\cdot)$, we can interpret the $y(\mathbf{x})$ as posterior probabilities.



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

Recap: Extension to Nonlinear Basis Fcts.

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

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

  - Basis functions $\phi_j(\mathbf{x})$ allow non-linear decision boundaries.
  - By choosing the right ϕ_j, every continuous function can (in principle) be approximated with arbitrary accuracy.
  - Disadvantage: minimization no longer in closed form.

- **Notation**

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

Recap: Gradient Descent

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

RWTH AACHEN UNIVERSITY

Recap: 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

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

RWTH AACHEN UNIVERSITY

Recap: Gradient Descent

- **Example: Quadratic error function**

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

- **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)**

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

Summary: Generalized Linear Discriminants

- Properties
  - General class of decision functions.
  - Nonlinearity $g(\cdot)$ and basis functions ϕ_j allow us to address linearly non-separable problems.
  - Shown simple sequential learning approach for parameter estimation using gradient descent.
- Limitations / Caveats
  - Flexibility of model is limited by curse of dimensionality
    - $g(\cdot)$ 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 minimize training error.

Topics of This Lecture

- Linear Discriminants Revisited
  - Linear Discriminants
  - Least-Squares Classification
  - Generalized Linear Discriminants
  - Gradient Descent
- Logistic Regression
  - Probabilistic discriminative models
  - Logistic sigmoid (logit function)
  - Cross-entropy error
  - Gradient descent
  - Note on error functions
- Softmax Regression
  - Multi-class generalization
  - Properties

Recap: Probabilistic Discriminative Models

- 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.

- Properties
  - Probabilistic interpretation
  - But discriminative method: only focus on decision hyperplane
  - Advantageous for high-dimensional spaces, requires less parameters than explicitly modeling $p(\phi|\mathcal{C}_k)$ and $p(\mathcal{C}_k)$.

Recap: 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:** $$\frac{d\sigma}{da} = \sigma(1-\sigma)$$

RWTH AACHEN UNIVERSITY

Recap: 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

$$\boxed{\begin{aligned} y_n &= \sigma(\mathbf{w}^T\phi_n) \\ \frac{dy_n}{d\mathbf{w}} &= y_n(1-y_n)\phi_n \end{aligned}}$$

- **Error function**

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

- **Gradient**

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

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…

RWTH AACHEN UNIVERSITY

Recap: Iteratively Reweighted Least Squares

- Result of applying Newton-Raphson to logistic regression

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

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

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

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

$\Rightarrow$ Iteratively Reweighted Least-Squares (IRLS)

RWTH AACHEN UNIVERSITY

Summary: Logistic Regression

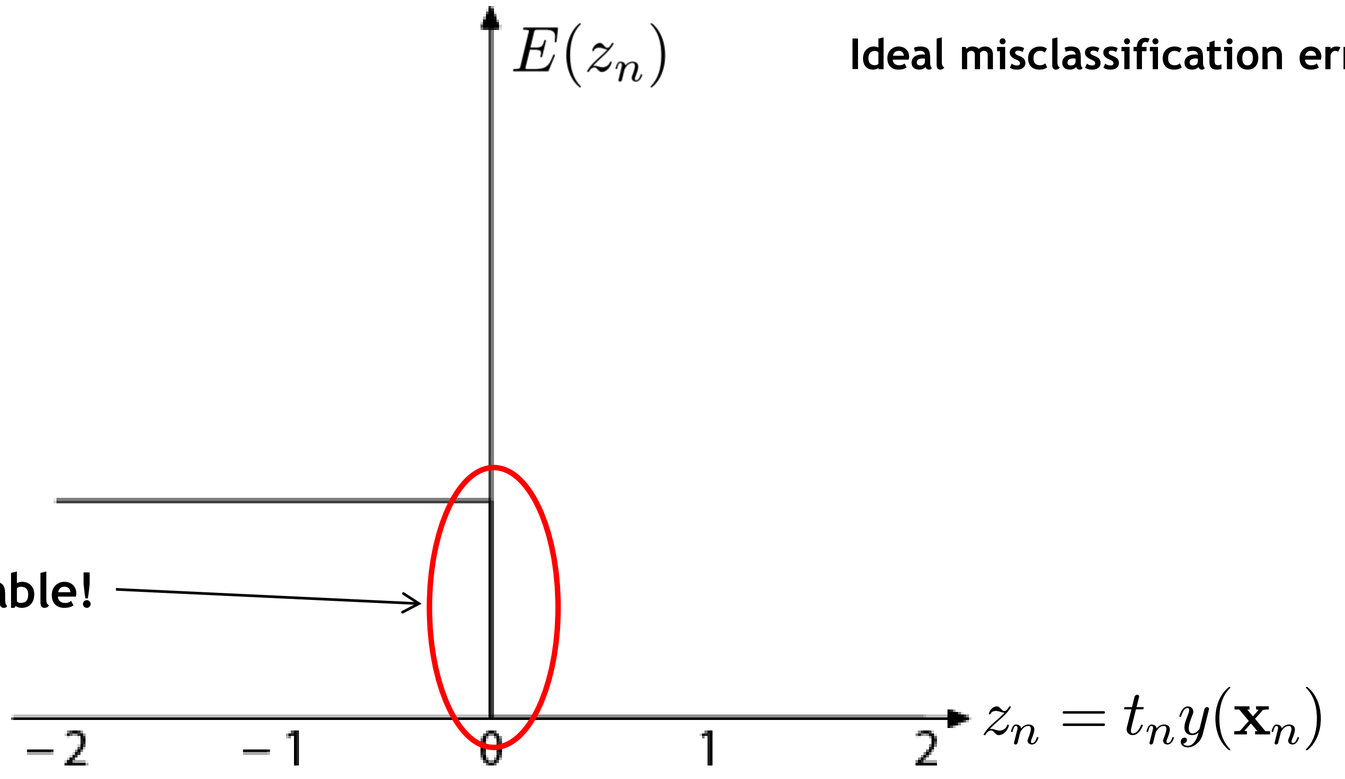
- Properties
  - Directly represent posterior distribution $p(\phi|\mathcal{C}_k)$
  - Requires fewer parameters than modeling the likelihood + prior.
  - Very often used in statistics.
  - It can be shown that the cross-entropy error function is concave
    - Optimization leads to unique minimum
    - But no closed-form solution exists
    - Iterative optimization (IRLS)
  - Both online and batch optimizations exist
- Caveat
  - Logistic regression tends to systematically overestimate odds ratios when the sample size is less than ~500.

Topics of This Lecture

- Linear Discriminants Revisited
  - Linear Discriminants
  - Least-Squares Classification
  - Generalized Linear Discriminants
  - Gradient Descent
- Logistic Regression
  - Probabilistic discriminative models
  - Logistic sigmoid (logit function)
  - Cross-entropy error
  - Gradient descent
  - Note on error functions
- Softmax Regression
  - Multi-class generalization
  - Properties

A Note on Error Functions

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



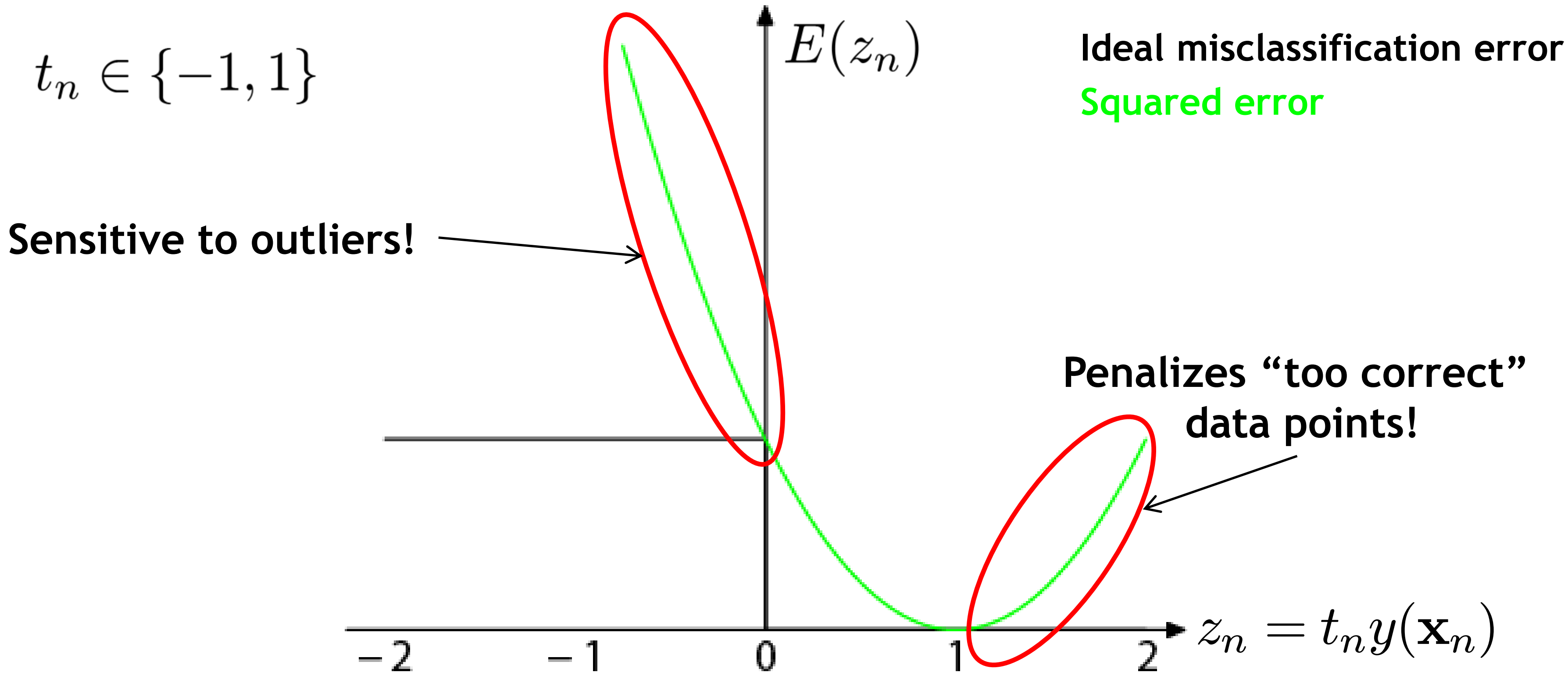
- **Ideal misclassification error function (black)**
  - This is what we want to approximate,
  - Unfortunately, it is not differentiable.
  - The gradient is zero for misclassified points.

$\Rightarrow$ **We cannot minimize it by gradient descent.**

Image source: Bishop, 2006

A Note on Error Functions

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



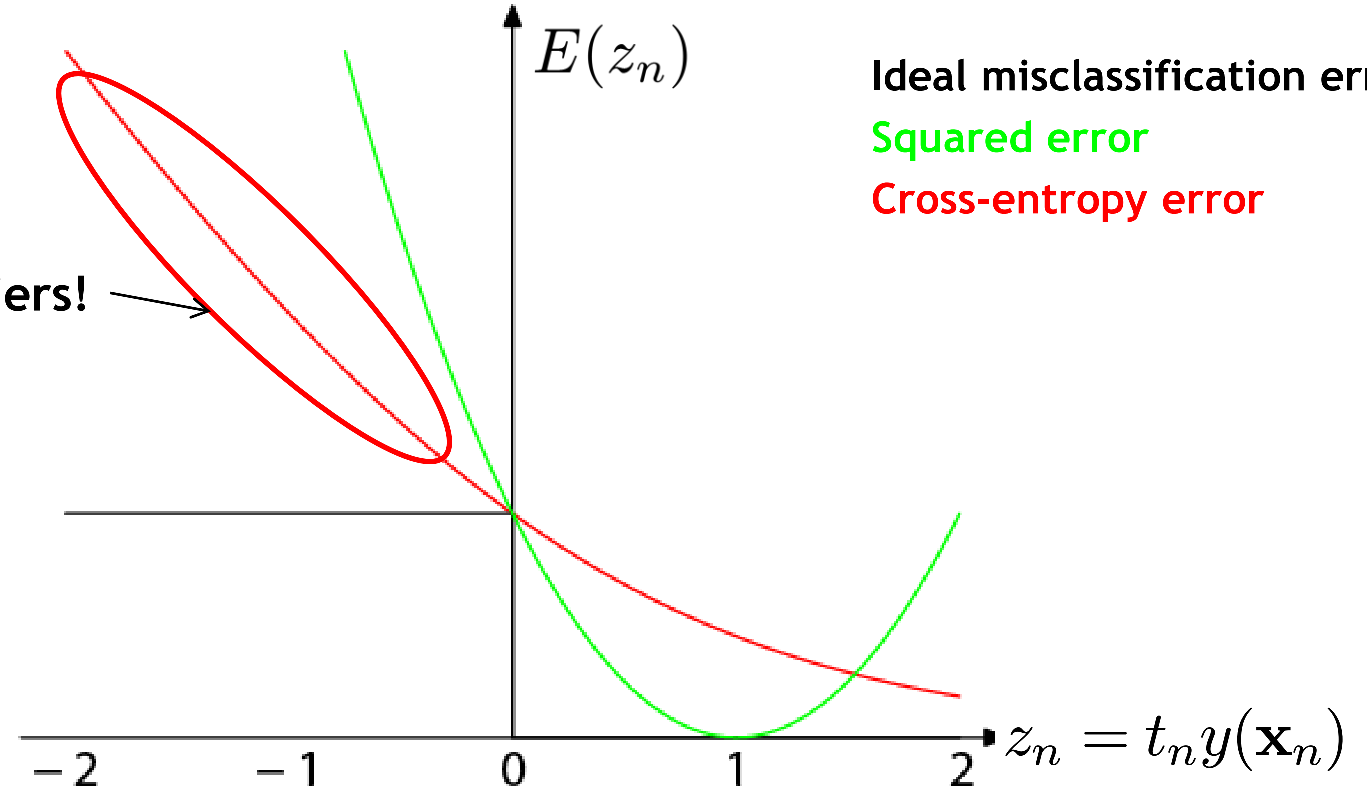
- Squared error used in Least-Squares Classification
  - Very popular, leads to closed-form solutions.
  - However, sensitive to outliers due to squared penalty.
  - Penalizes "too correct" data points

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

Image source: Bishop, 2006

A Note on Error Functions

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



Ideal misclassification error
Squared error
Cross-entropy error

- **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

- Linear Discriminants Revisited
  - Linear Discriminants
  - Least-Squares Classification
  - Generalized Linear Discriminants
  - Gradient Descent
- Logistic Regression
  - Probabilistic discriminative models
  - Logistic sigmoid (logit function)
  - Cross-entropy error
  - Gradient descent
  - Note on error functions
- Softmax Regression
  - **Multi-class generalization**
  - **Properties**

Softmax Regression

- Multi-class generalization of logistic regression
  - In logistic regression, we assumed binary labels $t_n \in \{0,1\}$
  - Softmax generalizes this to K values in 1-of-K notation.

$$\mathbf{y}(\mathbf{x};\mathbf{w}) = \begin{bmatrix} P(y=1|\mathbf{x};\mathbf{w}) \\ P(y=2|\mathbf{x};\mathbf{w}) \\ \vdots \\ P(y=K|\mathbf{x};\mathbf{w}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K}\exp(\mathbf{w}_j^\top \mathbf{x})} \begin{bmatrix} \exp(\mathbf{w}_1^\top \mathbf{x}) \\ \exp(\mathbf{w}_2^\top \mathbf{x}) \\ \vdots \\ \exp(\mathbf{w}_K^\top \mathbf{x}) \end{bmatrix}$$

  - This uses the softmax function

$$\frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

  - Note: the resulting distribution is normalized.

Softmax Regression Cost Function

- Logistic regression
  - Alternative way of writing the cost function

$$\begin{aligned} E(\mathbf{w}) &= -\sum_{n=1}^{N} \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} \\ &= -\sum_{n=1}^{N}\sum_{k=0}^{1} \{\mathbb{I}\left(t_n = k\right) \ln P\left(y_n = k|\mathbf{x}_n;\mathbf{w}\right)\} \end{aligned}$$

- Softmax regression
  - Generalization to K classes using indicator functions.

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} \left\{ \mathbb{I}\left(t_n = k\right) \ln \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x})} \right\}$$

Optimization

- Again, no closed-form solution is available
  - Resort again to Gradient Descent
  - Gradient

$$\nabla_{\mathbf{w}_k} E(\mathbf{w}) = -\sum_{n=1}^{N} \left[\mathbb{I}\left(t_n = k\right) \ln P\left(y_n = k | \mathbf{x}_n; \mathbf{w}\right)\right]$$

- Note
  - $\nabla_{\mathbf{w}k} E(\mathbf{w})$ is itself a vector of partial derivatives for the different components of $\mathbf{w}_k$.
  - We can now plug this into a standard optimization package.

Summary

- We have now an understanding of
  - Generalized Linear Discriminants as basic tools
  - Different loss functions and their effects
  - Softmax generalization to multi-class classification

- In the next lecture, we will see
  - How they are related to Neural Networks.
  - How we can use our new background to get a better understanding of *what NNs actually do*.

References and Further Reading

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

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