Advanced Machine Learning
Lecture 11

Linear Discriminants Revisited

03.12.2015

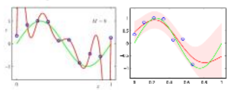
Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de/

leibe@vision.rwth-aachen.de

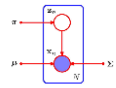
This Lecture: *Advanced Machine Learning*

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

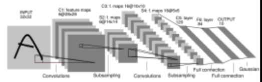
- Regression Approaches
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Gaussian Processes



- Learning with Latent Variables
  - Prob. Distributions & Approx. Inference
  - Mixture Models
  - EM and Generalizations



- 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, RBM, ...)

Topics of This Lecture

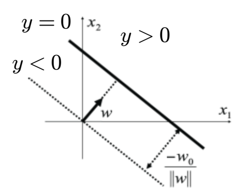
- Linear Discriminants Revisited (from ML lecture)
  - Linear Discriminants
  - Least-Squares Classification
  - Generalized Linear Discriminants
  - Gradient Descent
- Logistic Regression
  - Probabilistic discriminative models
  - Logistic sigmoid (logit function)
  - Cross-entropy error
  - Gradient 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

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

weight vector — "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.

Slide adapted from Bernt Schiele

Recap: Least-Squares Classification

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

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

  - Setting the derivative to zero yields

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

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

⇒ 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.

⇒ 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

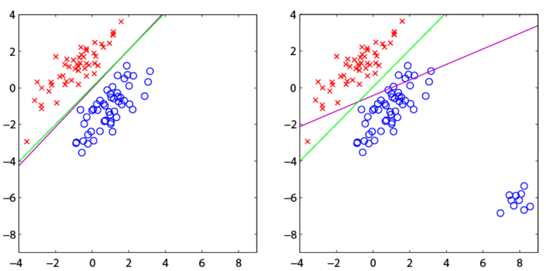
$$\widetilde{\mathbf{W}} = \widetilde{\mathbf{X}}^\dagger \mathbf{T} = (\widetilde{\mathbf{X}}^\top\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^\top \mathbf{T}$$

  - We then obtain the discriminant function as

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

⇒ 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

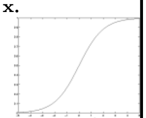
$$y(\mathbf{x}) = g(\mathbf{w}^\top\mathbf{x} + w_0)$$

  - $g(\cdot)$ 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 $\phi_j(\mathbf{x})$:

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

- **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.
  - ⇒ Minimization with Gradient Descent

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**

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

Slide credit: Bernt Schiele

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).

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

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

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)**

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

  - where

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

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

Slide adapted from Bernt Schiele

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)**

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

Slide adapted from Bernt Schiele

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

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

with

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

- 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: $\sigma(a) = \dfrac{1}{1+\exp(-a)}$
  - Inverse: $a = \ln\left(\dfrac{\sigma}{1-\sigma}\right)$ "logit" function
  - Symmetry property:

$$\sigma(-a) = 1 - \sigma(a)$$

  - Derivative: $\dfrac{d\sigma}{da} = \sigma(1-\sigma)$

Recap: Logistic Regression

- Let's consider a data set $\{\phi_n, t_n\}$ with $n = 1,\ldots,N$, where $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$ and $t_n \in \{0,1\}$, $\mathbf{t} = (t_1,\ldots,t_N)^T$.
- With $y_n = p(\mathcal{C}_1|\boldsymbol{\phi}_n)$, 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

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

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

Gradient of the Error Function

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

- Error function

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

- Gradient

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

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…

Recap: Iteratively Reweighted Least Squares

- Result of applying Newton-Raphson to logistic regression

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

$$= (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\left\{\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}\mathbf{w}^{(\tau)} - \mathbf{\Phi}^T(\mathbf{y}-\mathbf{t})\right\}$$

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

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

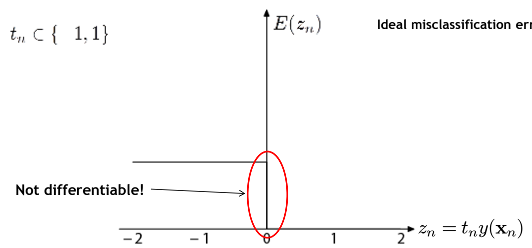
- Very similar form to pseudo-inverse (normal equations)
  - But now with non-constant weighing matrix $\mathbf{R}$ (depends on $\mathbf{w}$).
  - Need to apply normal equations iteratively.
  - ⇒ Iteratively Reweighted Least-Squares (IRLS)

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

Topics of This Lecture

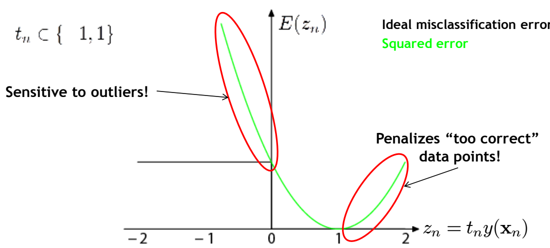
A Note on Error Functions



- 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.

Image source: Bishop, 2006

A Note on Error Functions



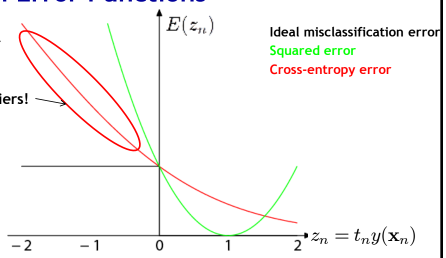
- Squared error used in Least-Squares Classification
  - Very popular, leads to closed-form solutions.
  - However, sensitive to outliers due to squared penalty.
  - Penalizes "too correct" data points
  - ⇒ Generally does not lead to good classifiers.

Image source: Bishop, 2006

A Note on Error Functions

$t_n \subset \{\ \ 1, 1\}$

Robust to outliers!



- **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

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

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

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

Optimization

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

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

- **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