Course Outline

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

- Discriminative Approaches (5 weeks)
  - Linear Discriminant Functions
  - Support Vector Machines
  - Ensemble Methods & Boosting
  - Randomized Trees, Forests & Ferns

- Generative Models (4 weeks)
  - Bayesian Networks
  - Markov Random Fields

Recap: Linear Discriminant Functions

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

- Linear discriminant functions
  
  \[ y(x) = w^T x + w_0 \]

  - The error function penalizes predictions that are “too correct”.

  - The error function penalizes predictions that are “too correct”.

Recap: Least-Squares Classification

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

  \[ E(w) = \sum_{i=1}^{n} (y(x_i; w) - t_i)^2 \]

  \[ E_D(\hat{W}) = \frac{1}{2} \text{tr} \left\{ (X\hat{W} - T)(X\hat{W} - T)^T \right\} \]

  - Setting the derivative to zero yields

  \[ \hat{W} = (X^T X)^{-1} X^T T \]

  We then obtain the discriminant function as

  \[ y(x) = \hat{W}^T x + \hat{w}_0 = \frac{1}{2} X^T \hat{X} \]

  \[ \Rightarrow \text{Exact, closed-form solution for the discriminant function} \]

Recap: Problems with Least Squares

- Least-squares is very sensitive to outliers!

  - The error function penalizes predictions that are “too correct”.

Recap: Generalized Linear Models

- Generalized linear model

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

  - \( g(\cdot) \) is called an activation function and may be nonlinear.

  - The decision surfaces correspond to

  \[ y(x) = \text{const.} \quad \Leftrightarrow \quad w^T x + w_0 = \text{const.} \]

  - If \( g \) is monotonic (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 \( g(\cdot) \), we can interpret \( y(x) \) as posterior probabilities.
Recap: Linear Separability

- Up to now: restrictive assumption
  - Only consider linear decision boundaries

- Classical counterexample: XOR

\[
\begin{align*}
  \text{XOR: } & C_1 \land \neg C_2 \lor \neg C_1 \land C_2 \\
  \text{OR: } & C_1 \lor C_2 \\
  \text{AND: } & C_1 \land C_2
\end{align*}
\]

Generalization

- 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 benchmark.

Linear Separability

- Model

\[
y_k(x) = \sum_{j=1}^{M} w_{kj} \phi_j(x) = y_k(x; w)
\]

  - \(K\) functions (outputs) \(y_k(x; w)\)

  - Learning in Neural Networks
  - Single-layer networks: \(\phi_j\) are fixed, only weights \(w\) are learned.
  - Multi-layer networks: both the \(w\) and the \(\phi_j\) are learned.

  - In the following, we will not go into details about neural networks in particular, but consider generalized linear discriminants in general...

Generalized Linear Discriminants

- Generalization
  - Transform vector \(x\) with \(M\) nonlinear basis functions \(\phi_j(x)\):

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

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

- Notation

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

Gradient Descent

- Learning the weights \(w\):
  - \(N\) training data points: \(X = \{x_1, \ldots, x_N\}\)
  - \(K\) outputs of decision functions: \(y_k(x_n; w)\)
  - Target vector for each data point: \(t = \{t_1, \ldots, t_K\}\)

  - Error function (least-squares error) of linear model

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

\[
= \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} \left( \sum_{j=1}^{M} w_{kj} \phi_j(x_n) - t_{kn} \right)^2
\]

- \(w(\tau)\) is the weight vector at \(\tau\)-th iteration

\[
\Delta w_{kj} = \frac{\partial E(w)}{\partial w_{kj}}
\]

\[
\eta: \text{Learning rate}
\]

\[
\begin{align*}
  w_{kj}^{(\tau+1)} &= w_{kj}^{(\tau)} - \eta \cdot \frac{\partial E(w)}{\partial w_{kj}} \\
  \eta: &\text{Learning rate}
\end{align*}
\]

- This simple scheme corresponds to a \(1^{\text{st}}\)-order Taylor expansion (There are more complex procedures available).
**Gradient Descent - Basic Strategies**

- "Batch learning"
  
  $$w_{kj}^{(r+1)} = w_{kj}^{(r)} - \eta \frac{\partial E(w)}{\partial w_{kj}} |_{w^{(r)}}$$
  
  $\eta$: Learning rate

- Compute the gradient based on all training data:
  
  $$\frac{\partial E(w)}{\partial w_{kj}}$$

**Gradient Descent**

- Error function
  
  $$E(w) = \sum_{n=1}^{N} E_n(w) = \frac{1}{2} \sum_{n=1}^{N} K \sum_{j=1}^{M} \left( \sum_{k=1}^{K} w_{kj} \phi_j(x_n) - t_{kn} \right)^2$$
  
  $$E_n(w) = \frac{1}{2} K \sum_{j=1}^{M} \left( \sum_{k=1}^{K} w_{kj} \phi_j(x_n) - t_{kn} \right)^2$$
  
  $$\frac{\partial E_n(w)}{\partial w_{kj}} = \sum_{j=1}^{M} w_{kj} \phi_j(x_n) - t_{kn} \phi_j(x_n) = (y_k(x_n; w) - t_{kn}) \phi_j(x_n)$$

**Gradient Descent - Basic Strategies**

- "Sequential updating"
  
  $$E(w) = \sum_{n=1}^{N} E_n(w)$$
  
  $$w_{kj}^{(r+1)} = w_{kj}^{(r)} - \eta \frac{\partial E_n(w)}{\partial w_{kj}} |_{w^{(r)}}$$
  
  $\eta$: Learning rate

- Compute the gradient based on a single data point at a time:
  
  $$\frac{\partial E_n(w)}{\partial w_{kj}}$$

**Gradient Descent**

- Delta rule (=LMS rule)
  
  $$w_{kj}^{(r+1)} = w_{kj}^{(r)} - \eta (y_k(x_n; w) - t_{kn}) \phi_j(x_n)$$
  
  where
  
  $$\delta_{kn} = y_k(x_n; w) - t_{kn}$$

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

**Gradient Descent**

- Cases with differentiable, non-linear activation function
  
  $$y_k(x) = g(a_k) = g \left( \sum_{j=0}^{M} w_{kj} \phi_j(x_n) \right)$$

- Gradient descent
  
  $$\frac{\partial E_n(w)}{\partial w_{kj}} = \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn}) \phi_j(x_n)$$
  
  $$w_{kj}^{(r+1)} = w_{kj}^{(r)} - \eta \delta_{kn} \phi_j(x_n)$$
  
  $$\delta_{kn} = \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn})$$

**Summary: Generalized Linear Discriminants**

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

- Limitations / Caveats
  
  - Flexibility of model is limited by curse of dimensionality
    - $g(\cdot)$ and $\phi_j$ 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.
Fisher's Linear Discriminant Analysis (FLD)

- **Better idea:** Find a projection that maximizes the ratio of the between-class variance to the within-class variance:
  
  \[ J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \]  

  - Usually, this is written as:
    
    \[ J(w) = \frac{w^T S_B w}{w^T S_W w} \]

  - where:
    
    \[ S_B = (m_2 - m_1)(m_2 - m_1)^T \]
    
    \[ S_W = \sum_{k=1}^{2} \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T \]

Classification as Dimensionality Reduction

- **Classification as dimensionality reduction**
  
  - We can interpret the linear classification model as a projection onto a lower-dimensional space.
  
  - E.g., take the \(d\)-dimensional input vector \(x\) and project it down to one dimension by applying the function:
    
    \[ y = w^T x \]

  - If we now place a threshold at \(y \geq -w_0\), we obtain our standard two-class linear classifier.

  - The classifier will have a lower error the better this projection separates the two classes.

  \[ \Rightarrow \text{New interpretation of the learning problem} \]

  - Try to find the projection vector \(w\) that maximizes the class separation.

Examples of good and bad separation:

- **Two questions**
  
  - How to measure class separation?
  
  - How to find the best projection (with maximal class separation)?
What does it mean to apply a linear classifier?

- **What does it mean to apply a linear classifier?**
  
  \[ y(x) = w^T x \]

  - **Weight vector**
  - **Input vector**

- **Classifier interpretation**
  - The weight vector has the same dimensionality as \( x \).
  - Positive contributions where \( \text{sign}(w_i) = \text{sign}(w_j) \).
  - The weight vector identifies which input dimensions are important for positive or negative classification (large \( |w_i| \)) and which ones are irrelevant (near-zero \( w_i \)).
  - If the inputs \( x \) are normalized, we can interpret \( w \) as a "template" vector that the classifier tries to match.

\[ w^T x = |w||x| \cos \theta \]

Multiple Discriminant Analysis

- **Generalization to \( K \) classes**

\[
J(W) = \frac{|W^T S_B W|}{|W^T S_W W|}
\]

- where

\[
W = [w_1, \ldots, w_K] \quad m = \frac{1}{N} \sum_{n=1}^{N} x_n = \frac{1}{K} \sum_{k=1}^{K} N_k m_k
\]

\[
S_B = \sum_{k=1}^{K} N_k (m_k - m)(m_k - m)^T \quad S_W = \sum_{k=1}^{K} \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T
\]

Maximizing \( J(W) \)

- "Rayleigh quotient" \( \Rightarrow \) Generalized eigenvalue problem

\[
J(W) = \frac{|W^T S_B W|}{|W^T S_W W|}
\]

- The columns of the optimal \( W \) are the eigenvectors corresponding to the largest eigenvalues of

\[
S_B W = \lambda_i S_W W_i
\]

- Defining \( V = S_B^{-1} W \), we get

\[
S_B S_W^{-1} S_B^{-1} V = \lambda V
\]

which is a regular eigenvalue problem.

\( \Rightarrow \) Solve to get eigenvectors of \( V \), then from that of \( W \).

- For the \( K \)-class case we obtain (at most) \( K-1 \) projections,
  - (i.e. eigenvectors corresponding to non-zero eigenvalues.)

What Does It Mean?

- **Multiple Discriminant Analysis**

  - Generalization to \( K \) classes

  \[ J(W) = \frac{|W^T S_B W|}{|W^T S_W W|} \]

  - where

  \[
  W = [w_1, \ldots, w_K] \quad m = \frac{1}{N} \sum_{n=1}^{N} x_n = \frac{1}{K} \sum_{k=1}^{K} N_k m_k
  \]

  \[
  S_B = \sum_{k=1}^{K} N_k (m_k - m)(m_k - m)^T \quad S_W = \sum_{k=1}^{K} \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T
  \]

  \[ \begin{align*}
  J(W) &= \frac{|W^T S_B W|}{|W^T S_W W|} \\
  &= \frac{\sum_{i=1}^{K} w_i^T S_B w_i}{\sum_{i=1}^{K} w_i^T S_W w_i}
  \end{align*} \]

Summary: Fisher’s Linear Discriminant

- **Properties**
  - Simple method for dimensionality reduction, preserves class discriminability.
  - Can use parametric methods in reduced-dim. space that might not be feasible in original higher-dim. space.
  - Widely used in practical applications.

- **Restrictions / Caveats**
  - Not possible to get more than \( K-1 \) projections.
  - FLD reduces the computation to class means and covariances.

  \( \Rightarrow \) Implicit assumption that class distributions are unimodal and well-approximated by a Gaussian/hyperellipsoid.
Fisher’s linear discriminant (FLD) and assume we represent $w_p p_p$ classification as dimensionality reduction. Applications together with the class priors, this gives

$$p(C_1 | x) = \sigma(a) = \frac{1}{1 + \exp(-a)}$$

logistic sigmoid function

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

Or we can use generalized linear discriminant models $a = w^T x$

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

This model is called logistic regression.

Probabilistic Discriminative Models

In the following, we will consider models of the form

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

with

$$p(C_2 | \phi) = 1 - p(C_1 | \phi)$$

This model is called logistic regression.

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

Let’s look at the number of parameters…

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

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

Logistic Regression

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

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

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

Define the error function as the negative log-likelihood

$$E(w) = -\ln p(t | w)$$

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

This is the so-called cross-entropy error function.
Gradient of the Error Function

- **Error function**
  \[ E(\mathbf{w}) = -\sum_{n=1}^{N} \left\{ t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right\} \]

- **Gradient**
  \[ \nabla E(\mathbf{w}) = -\sum_{n=1}^{N} \left\{ t_n \frac{\partial}{\partial \mathbf{w}} \ln y_n + (1 - t_n) \frac{\partial}{\partial \mathbf{w}} \ln(1 - y_n) \right\} \]
  \[ = -\sum_{n=1}^{N} \left\{ t_n \frac{n - y_n}{y_n} \phi_n - (1 - t_n) \frac{y_n}{1 - y_n} \phi_n \right\} \]
  \[ = -\sum_{n=1}^{N} \left\{ (t_n - y_n) \phi_n - y_n + t_n \phi_n \right\} \]
  \[ = \sum_{n=1}^{N} (y_n - t_n) \phi_n \]

Gradient of the Error Function

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

- **Does this look familiar to you?**
- **This is the same result as for the Delta (=LMS) rule**
  \[ w^{(2+)} = w^{(2)} - \eta(y_k(\mathbf{x}_n; \mathbf{w}) - t_k) \phi_j(\mathbf{x}_n) \]
- **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**
  \[ \mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \mathbf{H}^{-1} \nabla E(\mathbf{w}) \]
  where \( \mathbf{H} = \nabla^2 E(\mathbf{w}) \) 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:
  \[ E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (\mathbf{w}^T \phi_n - t_n)^2 \]
  \[ \nabla E(\mathbf{w}) = \sum_{n=1}^{N} (\mathbf{w}^T \phi_n - t_n) \phi_n = \Phi^T \Phi \mathbf{w} - \Phi^T \mathbf{t} \]
  \[ \mathbf{H} = \nabla^2 E(\mathbf{w}) = \sum_{n=1}^{N} \phi_n \phi_n^T = \Phi^T \Phi \quad \text{where} \quad \Phi = \begin{bmatrix} \phi_1^T \\ \vdots \\ \phi_N^T \end{bmatrix} \]
- **Resulting update scheme:**
  \[ \mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - (\Phi^T \Phi)^{-1} \Phi^T (\mathbf{w}^{(r)} - \Phi^T \mathbf{t}) \]
  \[ = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t} \quad \text{Closed-form solution!} \]

Newton-Raphson for Logistic Regression

- Now, let’s try Newton-Raphson on the cross-entropy error function:
  \[ E(\mathbf{w}) = -\sum_{n=1}^{N} \left\{ t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right\} \]
  \[ \frac{\partial}{\partial \mathbf{w}} E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n) \phi_n = \Phi^T \mathbf{y} - \Phi^T \mathbf{t} \]
  \[ \mathbf{H} = \nabla^2 E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1 - y_n) \phi_n \phi_n^T = \Phi^T \mathbf{R} \Phi \]
  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**
  \[ \mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t}) \]
  \[ = (\Phi^T \mathbf{R} \Phi)^{-1} \left\{ \Phi^T \mathbf{R} \Phi \mathbf{w}^{(r)} - \Phi^T (\mathbf{y} - \mathbf{t}) \right\} \]
  \[ = (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T \mathbf{R} \mathbf{z} \]
  \[ \text{with} \quad \mathbf{z} = \Phi \mathbf{w}^{(r)} - \mathbf{R}^{-1} (\mathbf{y} - \mathbf{t}) \]

- **Again very similar form (normal equations)**
  - But now with non-constant weighting 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 \( p(C_k | x_i) \)
  - 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

- Fisher’s linear discriminant (FLD)
  - Classification as dimensionality reduction
  - Linear discriminant analysis
  - Multiple discriminant analysis
  - Applications

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

- Note on Error Functions

Note on Error Functions

- **Ideal misclassification error** function (black)
  - This is what we would like to approximate.
  - Unfortunately, it is not differentiable.
  - The gradient is zero for misclassified points.
  - We cannot minimize it by gradient descent.

Comparing Error Functions (Loss 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.

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

\( \Rightarrow \) Analysis tool to compare classification approaches
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 - 4.3).

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