Recap: Linear Discriminant Functions

- Basic idea
  - Directly encode decision boundary
  - Minimize misclassification probability directly.
- Linear discriminant functions
  \[ y(x) = w^T x + w_0 \]
  - \( w, w_0 \) 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
    \[ E(w) = \sum_{n=1}^{N} (y(x_n; w) - t_n)^2 \]
  - Setting the derivative to zero yields
    \[ \mathbf{W} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{T} \]
  - We then obtain the discriminant function as
    \[ y(x) = \mathbf{W}^T \mathbf{x} = \mathbf{T}^T \mathbf{X}^{-1} \mathbf{X}^T \mathbf{x} \]
  - \( \Rightarrow \) 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”.

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
    \[ y(x) = \text{const.} \iff w^T x + w_0 = \text{const.} \]
  - 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 \( g(\cdot) \), we can interpret the \( y(x) \) as posterior probabilities.
Recap: Linear Separability

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

Recap: Probabilistic Discriminative Models

- Fisher’s Linear Discriminant (FLD)
- Start with an initial guess for the parameter values $w^{(0)}$
- By choosing the right $\phi$,...
- Sequential updating leads to...

Advantages

- Transformation allows non-linear decision boundaries.
- By choosing the right $\phi$,...

Disadvantage

- The error function can in general no longer be minimized in closed form.
  - Minimization with Gradient Descent

Recap: Iterative Gradient Descent

- Iterative minimization
  - Start with an initial guess for the parameter values $w^{(0)}$
  - Move towards a (local) minimum by following the gradient.
- Basic strategies
  - “Batch learning” $w^{(r+1)}_{kj} = w^{(r)}_{kj} - \eta \frac{\partial E(w)}{\partial w_{kj}} \bigg|_{w^{(r)}}$
  - “Sequential updating” $w^{(r+1)}_{kj} = w^{(r)}_{kj} - \eta \frac{\partial E_n(w)}{\partial w_{kj}} \bigg|_{w^{(r)}}$
- Simply feed back the input data point, weighted by the classification error.

Recap: Extension to Nonlinear Basis Funcs.

- Generalization
  - Transform vector $x$ with $M$ nonlinear basis functions $\phi_j(x)$:
    $$y_k(x) = \sum_{j=1}^{M} w_{kj}\phi_j(x) + w_{k0}$$
- Advantages
  - Transformation allows non-linear decision boundaries.
  - By choosing the right $\phi$,...
- Disadvantage
  - The error function can in general no longer be minimized in closed form.
  - Minimization with Gradient Descent

Recap: Gradient Descent

- Example: Quadratic error function
  $$E(w) = \sum_{n=1}^{N} (y(x_n; w) - t_n)^2$$
- Sequential updating leads to delta rule (=LMS rule)
  $$w^{(r+1)}_{kj} = w^{(r)}_{kj} - \eta (y_k(x_n; w) - t_{kn}) \phi_j(x_n)$$
  $$\delta_{kn} = y_k(x_n; w) - t_{kn}$$
  - Minimization with Gradient Descent

Recap: 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 (again with quadratic error function)
  $$\frac{\partial E_n(w)}{\partial w_{kj}} = \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn}) \phi_j(x_n)$$
  $$w^{(r+1)}_{kj} = w^{(r)}_{kj} - \eta \delta_{kn} \phi_j(x_n)$$
  $$\delta_{kn} = \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn})$$

Topics of This Lecture

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

How to measure class separation?

- Classification function:
  \[ J(w) = \frac{w^T S_B w}{w^T S_W w} \]

  These problems exist.
  - We could simply measure the separation of the class means.
  \[ (m_2 - m_1) = w^T (m_2 - m_1) \]
  \[ \Rightarrow \text{Choose } w \text{ so as to maximize} \]

- Problems with this approach
  1. This expression can be made arbitrarily large by increasing $|w|$.
  \[ \Rightarrow \text{Need to enforce additional constraint} ||w|| = 1 \]
  \[ \Rightarrow \text{This constrained minimization results in} \]

- Better idea:
  \[ w / (m_2 - m_1) \]

Fisher’s Linear Discriminant Analysis (FLD)

- Better idea:
  \[ J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \text{ with } s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2 \]

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

  \[ \text{where} \]
  \[ S_B = (m_2 - m_1)(m_2 - m_1)^T \]

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

  \[ J(W) = jW^T S_B W j \]

  \[ jW^T S_W W j \]

Multiple Discriminant Analysis

- Generalization to $K$ classes
  \[ J(W) = \frac{|W^T S_B W|}{|W^T S_W W|} \]

  \[ \text{where} \]
  \[ W = [w_1, \ldots, w_K] \]

  \[ m = \frac{1}{N} \sum_{n=1}^{N} x_n = \frac{1}{K} \sum_{k=1}^{K} N_k m_k \]

  \[ S_B = \sum_{k=1}^{K} N_k (m_k - m)(m_k - m)^T \]

  \[ S_W = \sum_{k=1}^{K} \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T \]
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 $S_B W_i = \lambda_i S_W W_i$
  - Defining $V = S_B^{-1} W$, we get
    $$S_W S_B^{-1} S_W V = \lambda V$$
    which is a regular eigenvalue problem.
  - 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?

- What does it mean to apply a linear classifier?
  $$y(x) = \tilde{W}^T x$$
  Weight vector Input vector

- Classifier interpretation
  - The weight vector has the same dimensionality as $x$.
  - Positive contributions where $\text{sign}(x_i) = \text{sign}(w_i)$.
  - 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.

Example Application: Fisherfaces

- Visual discrimination task
  - Training data:
    - $C_1$: Subjects with glasses
    - $C_2$: Subjects without glasses
  - Test:
    - $x$ - glasses?

  Take each image as a vector of pixel values and apply FLD...

Fisherfaces: Interpretability

- Resulting weight vector for "Glasses/NoGlasses"

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.
  - Implicit assumption that class distributions are unimodal and well-approximated by a Gaussian/hyperellipsoid.

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
Probabilistic Discriminative Models

- We have seen that we can write
  \[ p(C_1|x) = \sigma(a) = \frac{1}{1 + \exp(-a)} \]
- 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 \[ a = w^T \phi(x) \]

B. Leibe

Comparison

- 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?
    - Just the values of \( w \) \( \Rightarrow \) For large \( M \), logistic regression has clear advantages!

B. Leibe

Logistic Sigmoid

- Properties
  - Definition: \( \sigma(a) = \frac{1}{1 + \exp(-a)} \)
  - Inverse: \( a = \ln \left( \frac{\sigma}{1 - \sigma} \right) \) \( \Rightarrow \) “logit” function
  - Symmetry property: \( \sigma(-a) = 1 - \sigma(a) \)
  - Derivative: \( \frac{d\sigma}{da} = \sigma(1 - \sigma) \)

B. Leibe

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.

B. Leibe

Gradient of the Error Function

- Error function
  \[ y_n = \sigma(w^T \phi_n) \]
  \[ dE/dw = y_n(1 - y_n) \phi_n \]
- Gradient
  \[ \nabla E(w) = -\sum_{n=1}^{N} \left\{ t_n \frac{d}{d\phi_n} \frac{y_n}{y_n + (1 - y_n)} + (1 - t_n) \frac{d}{d\phi_n} \frac{1 - y_n}{y_n + (1 - y_n)} \right\} \]
  \[ = -\sum_{n=1}^{N} \left\{ t_n \frac{y_n - y_n \phi_n - (1 - y_n) \phi_n}{y_n + (1 - y_n)} \phi_n \right\} \]
  \[ = \sum_{n=1}^{N} \{y_n - t_n\} \phi_n \]
Gradient of the Error Function

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

- Does this look familiar to you?
- This is the same result as for the Delta (=LMS) rule
  \[ w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta (y_k(x_n; w) - t_{kn}) \phi_j(x_n) \]
- 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
  \[ w^{(\tau+1)} = w^{(\tau)} - H^{-1} \nabla E(w) \]
  where \( H = \nabla^2 E(w) \) 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:
  \[ E(w) = \frac{1}{2} \sum_{n=1}^{N} (w^T \phi_n - t_n)^2 \]

- Resulting update rule:
  \[ w^{(\tau+1)} = w^{(\tau)} - (\Phi^T \Phi)^{-1} (\Phi^T \Phi w^{(\tau)} - \Phi^T t) \]
  \[ = (\Phi^T \Phi)^{-1} \Phi^T t \] Closed-form solution!

Iteratively Reweighted Least Squares

- Update equations
  \[ w^{(\tau+1)} = w^{(\tau)} - (\Phi^T R \Phi)^{-1} \Phi^T (y - t) \]
  \[ = (\Phi^T R \Phi)^{-1} \left( \Phi^T R \Phi w^{(\tau)} - \Phi^T (y - t) \right) \]
  \[ = (\Phi^T R \Phi)^{-1} \Phi^T R z \] with \( z = \Phi w^{(\tau)} - R^{-1} (y - t) \)

- Again very similar form (normal equations)
  - But now with non-constant weighing matrix \( R \) (depends on \( w \)).
  - Need to apply normal equations iteratively.
  \( \Rightarrow \) Iteratively Reweighted Least-Squares (IRLS)

Summary: Logistic Regression

- Properties
  - Directly represent posterior distribution \( p(\theta | \mathcal{D}) \)
  - 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

- We have now seen already a number of error functions
  - Ideal misclassification error
  - Quadratic error
  - Cross-entropy error

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.

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