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
  \[
  E(w) = \sum_{n=1}^{N} (y(x_n; w) - t_n)^2
  \]
  \[
  E_D(W) = \frac{1}{2} \text{Tr} \left\{ (XW - T)(XW - T)^T \right\}
  \]
  - Setting the derivative to zero yields
  \[
  W = (X^T X)^{-1}X^T T = (X^T X)^{-1}X^T
  \]
  - We then obtain the discriminant function as
  \[
  y(x) = W^T x = T^T (X^T x)
  \]
  \[ \Rightarrow \] 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
    \[ y(x) = \text{const.} \Leftrightarrow w^T x + w_0 = \text{const.} \]
  - 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.
Linear Separability

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

- Classical counterexample: XOR

\[ y_k(x) = \sum_{j=0}^{M} w_{kj} \phi_j(x) + w_{k0} \]

- 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

Generalized Linear Discriminants

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

- Properties
  - Global
    - A small change in \( x \) affects all basis functions.

- Result
  - If we use polynomial basis functions, the decision boundary will be a polynomial function of \( x \).
    - Nonlinear decision boundaries
    - However, we still solve a linear problem in \( \phi(x) \).

Linear Basis Function Models

- Generalized Linear Discriminant Model

  \[ y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x) \]

  - where \( \phi_j(x) \) are known as basis functions.
  - Typically, \( \phi_0(x) = 1 \), so that \( w_0 \) acts as a bias.
  - In the simplest case, we use linear basis functions: \( \phi(x) = x \).

- Let’s take a look at some other possible basis functions...

Linear Basis Function Models (2)

- Polynomial basis functions

- Properties
  - Local
    - A small change in \( x \) affects only nearby basis functions.
  - \( \mu_j \) and \( s \) control location and scale (width).

- Result
  - If we use polynomial basis functions, the decision boundary will be a polynomial function of \( x \).
    - Nonlinear decision boundaries
    - However, we still solve a linear problem in \( \phi(x) \).

Linear Basis Function Models (3)

- Gaussian basis functions

  \[ \phi_j(x) = \exp \left\{ -\frac{(x - \mu_j)^2}{2s^2} \right\} \]

- Properties
  - Local
    - A small change in \( x \) affects only nearby basis functions.
  - \( \mu_j \) and \( s \) control location and scale (width).

Linear Basis Function Models (4)

- Sigmoid basis functions

  \[ \phi_j(x) = \sigma \left( \frac{x - \mu_j}{s} \right) \]

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

- Properties
  - Local
    - A small change in \( x \) affects only nearby basis functions.
Graduate Descent

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

  - Error function (least-squares error) of linear model
    \[
    E(w) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k(x_n; w) - t_k) \frac{1}{M} \sum_{j=1}^{M} \phi_j(x_n) - t_k)^2
    \]
    \[
    E_n(w) = \frac{1}{2} \sum_{k=1}^{K} \left( \sum_{j=1}^{M} w_{kj} \phi_j(x_n) - t_k \right)^2
    \]
    \[
    \frac{\partial E_n(w)}{\partial w_{kj}} = \sum_{j=1}^{M} w_{kj} \phi_j(x_n) - t_k \frac{\partial \phi_j(x_n)}{\partial w_{kj}}
    \]

• “Batch learning”
  \[
  w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(w)}{\partial w_{kj}} |_{w^{(\tau)}}
  \]
  \( \eta \): Learning rate

• “Sequential updating”
  \[
  E(w) = \sum_{n=1}^{N} E_n(w)
  \]
  \[
  w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E_n(w)}{\partial w_{kj}} |_{w^{(\tau)}}
  \]
  \( \eta \): Learning rate

  - Compute the gradient based on a single data point at a time:
    \[
    \frac{\partial E_n(w)}{\partial w_{kj}} = \phi_j(x_n) \left( y_k(x_n; w) - t_k \right)
    \]

  - Compute the gradient based on all training data:
    \[
    \frac{\partial E(w)}{\partial w_{kj}} = \sum_{n=1}^{N} \sum_{k=1}^{K} \left( \sum_{j=1}^{M} w_{kj} \phi_j(x_n) - t_k \right) \phi_j(x_n)
    \]

Basic Strategies

• Note on Error Functions
  - Ideal error function
  - Quadratic error
  - Cross-entropy error

• Iteratively Reweighted Least Squares

• Machine Learning
  - Probabilistic discriminative models
  - Logistic signode (logit function)
  - Cross-entropy error
  - Iteratively Reweighted Least Squares

• Note on Error Functions
  - Closed form
  - More complex procedures available

• Error function (entropy error)
  - For decision functions
  - For classification problems

• Error function (class generalization)
  - Adaptive boosting
  - Neural networks

• Error function (softmax)
  - Softmax regression

• Error function (softmax)
  - Softmax (logit function)
  - Class-specific output

• Error function (cross-entropy)
  - Cross-entropy error
Delta rule (=LMS rule)

\[ w_{kj}^{(r+1)} = w_{kj}^{(r)} - \eta (y_k(x_n; w) - t_{kn}) \phi_j(x_n) \]

where

\[ \delta_{kn} = y_k(x_n; w) - t_{kn} \]

\[ \Rightarrow \text{Simply feed back the input data point, weighted by the classification error.} \]

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

\[ a \equiv \ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)} \]

- Or we can use generalized linear discriminant models

\[ a = w^T x \]

\[ \text{or} \quad a = w^T \phi(x) \]

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

\[ \frac{\partial E_n(w)}{\partial w_{kj}} = \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn}) \phi_j(x_n) \]

\[ w_{kj}^{(r+1)} = w_{kj}^{(r)} - \eta \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn}) \phi_j(x_n) \]

\[ \delta_{kn} = \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(x_n; w) - t_{kn}) \]

Summary: Generalized Linear Discriminants

- Properties
  - General class of decision functions.
  - Nonlinearity \( g() \) 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 are available (e.g. Newton-Raphson), but they are more expensive to compute.

- Limitations / Caveats
  - Flexibility of model is limited by curse of dimensionality
    - \( g() \) 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.

Topics of This Lecture

- Gradient Descent

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

- Softmax Regression
  - Multi-class generalization
  - Gradient descent solution

- Note on Error Functions
  - Ideal error function
  - Quadratic error
  - Cross-entropy error

Probabilistic Discriminative Models

- In the following, we will consider models of the form

\[ p(C_1 | \phi) = y(\phi) = \sigma(w^T \phi) \]

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 $p(\phi|C_i)$ and $p(C_i)$ 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 $w$ ⇒ 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

- 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 (\textit{LMS}) rule
  $$w_{k,j}^{(\tau+1)} = w_{k,j}^{(\tau)} - \eta (y_{\tau}(x_n; w) - t_{\tau,n}) \phi_j(x_n)$$
- We can use this to derive a sequential estimation algorithm.
  - However, this will be quite slow...

### 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: $\frac{da}{da} = \sigma(1 - \sigma)$

### Gradient of the Error Function

- Error function
  $$E(w) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln (1 - y_n)\}$$
- Gradient
  $$\nabla E(w) = -\sum_{n=1}^{N} \left\{ t_n \frac{\partial \ln y_n}{\partial w} + (1 - t_n) \frac{\partial \ln (1 - y_n)}{\partial w} \right\}$$
  $$= -\sum_{n=1}^{N} \left\{ t_n \frac{1 - y_n}{y_n} \phi_n - (1 - t_n) \frac{y_n}{1 - y_n} \phi_n \right\}$$
  $$= \sum_{n=1}^{N} (t_n - y_n) \phi_n$$

### 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:
  \[ E(w) = \frac{1}{2} \sum_{n=1}^{N} (w^T \phi_n - t_n)^2 \]
  \[ \nabla E(w) = \sum_{n=1}^{N} (w^T \phi_n - t_n) \phi_n = \Phi^T \Phi w - \Phi^T t \]
  \[ H = \nabla^2 E(w) = \sum_{n=1}^{N} \phi_n \phi_n^T = \Phi^T \Phi \]
  where \( \Phi = \left[ \begin{array}{c} \phi_1^T \\ \vdots \\ \phi_N^T \end{array} \right] \)
- Resulting update scheme:
  \[ w^{(r+1)} = w^{(r)} - (\Phi^T \Phi)^{-1} \Phi^T (y - t) \]
  = \( (\Phi^T \Phi)^{-1} \Phi^T t \) \quad Closed-form solution!

Iteratively Reweighted Least Squares

- Update equations
  \[ w^{(r+1)} = w^{(r)} - \left( \Phi^T R \Phi \right)^{-1} \Phi^T (y - t) \]
  \[ = \left( \Phi^T R \Phi \right)^{-1} \left\{ \Phi^T R \Phi w^{(r)} - \Phi^T (y - t) \right\} \]
  \[ = \left( \Phi^T R \Phi \right)^{-1} \Phi^T R z \]
  where \( z = \Phi w^{(r)} - R^{-1} (y - t) \)
- Again very similar form (normal equations)
  - But now with non-constant weighting 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 \( p(\theta|C) \)
  - 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 \( \sim 500 \).

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.
  \[ y(x; w) = \begin{cases} 
  P(y = 1|x; w) = \frac{1}{\sum_{j=1}^{K} \exp(w_j^T x)} \exp(w_1^T x) \\
  \vdots \\
  P(y = K|x; w) = \frac{1}{\sum_{j=1}^{K} \exp(w_j^T x)} \exp(w_K^T x) 
\end{cases} \]
  - This uses the \texttt{softmax} function
    \[ \text{softmax}(a_k) = \frac{\exp(a_k)}{\sum_{j} \exp(a_j)} \]
  - Note: the resulting distribution is normalized.
Softmax Regression Cost Function

- Logistic regression
  - Alternative way of writing the cost function
    \[
    E(w) = - \sum_{n=1}^{N} \left\{ t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right\}
    \]
    \[
    = - \sum_{n=1}^{N} \sum_{k=1}^{K} \left\{ I(t_n = k) \ln P(y_n = k|x_n; w) \right\}
    \]
  - Softmax regression
    - Generalization to K classes using indicator functions.

\[
E(w) = - \sum_{n=1}^{N} \sum_{k=1}^{K} \left\{ I(t_n = k) \ln \frac{\exp(w^T x_n)}{\sum_{j=1}^{K} \exp(w^T x_n)} \right\}
\]

Optimization

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

\[
\nabla_w E(w) = - \sum_{n=1}^{N} \left\{ I(t_n = k) \ln P(y_n = k|x_n; w) \right\}
\]

- Note
  - \(\nabla_w E(w)\) is itself a vector of partial derivatives for the different components of \(w\).
  - We can now plug this into a standard optimization package.

Topics of This Lecture

- Gradient Descent
- Logistic Regression
  - Probabilistic discriminative models
  - Logistic sigmoid (logit function)
  - Cross-entropy error
  - Iteratively Reweighted Least Squares
- Softmax Regression
  - Multi-class generalization
  - Gradient descent solution
- Note on Error Functions
  - Ideal error function
  - Quadratic error
  - Cross-entropy error

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.

- 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.

Note on Error Functions

- Ideal misclassification error function (black)
  - This is what we want to approximate (error = #misclassifications)
  - 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.

⇒ **Looking at the error function this way gives us an analysis tool to compare the properties of 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