

Linear Models for Classification

Oliver Schulte - CMPT 726

Bishop PRML Ch. 4

Classification: Hand-written Digit Recognition

$$\mathbf{x}_i = \begin{array}{|c|} \hline \text{4} \\ \hline \end{array} \quad \mathbf{t}_i = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0)$$

- Each input vector classified into one of K discrete classes
 - Denote classes by \mathcal{C}_k
- Represent input image as a vector $\mathbf{x}_i \in \mathbb{R}^{784}$.
- We have target vector $\mathbf{t}_i \in \{0, 1\}^{10}$
- Given a **training set** $\{(\mathbf{x}_1, \mathbf{t}_1), \dots, (\mathbf{x}_N, \mathbf{t}_N)\}$, learning problem is to construct a “good” function $\mathbf{y}(\mathbf{x})$ from these.
 - $\mathbf{y} : \mathbb{R}^{784} \rightarrow \mathbb{R}^{10}$

Generalized Linear Models

- Similar to previous chapter on linear models for regression, we will use a “linear” model for classification:

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

- This is called a **generalized linear model**
- $f(\cdot)$ is a fixed non-linear function
 - e.g.

$$f(u) = \begin{cases} 1 & \text{if } u \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- **Decision boundary** between classes will be linear function of \mathbf{x}
- Can also apply non-linearity to \mathbf{x} , as in $\phi_i(\mathbf{x})$ for regression

Generalized Linear Models

- Similar to previous chapter on linear models for regression, we will use a “linear” model for classification:

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

- This is called a **generalized linear model**
- $f(\cdot)$ is a fixed non-linear function
 - e.g.

$$f(u) = \begin{cases} 1 & \text{if } u \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- **Decision boundary** between classes will be linear function of \mathbf{x}
- Can also apply non-linearity to \mathbf{x} , as in $\phi_i(\mathbf{x})$ for regression

Generalized Linear Models

- Similar to previous chapter on linear models for regression, we will use a “linear” model for classification:

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

- This is called a **generalized linear model**
- $f(\cdot)$ is a fixed non-linear function
 - e.g.

$$f(u) = \begin{cases} 1 & \text{if } u \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- **Decision boundary** between classes will be linear function of \mathbf{x}
- Can also apply non-linearity to \mathbf{x} , as in $\phi_i(\mathbf{x})$ for regression

Overview

- Linear regression for Classification
- The Fisher Linear Discriminant, or How to Draw a Line Between Classes
- The Perceptron, or The Smallest Neural Net
- Logistic Regression—The Statistician's Classifier

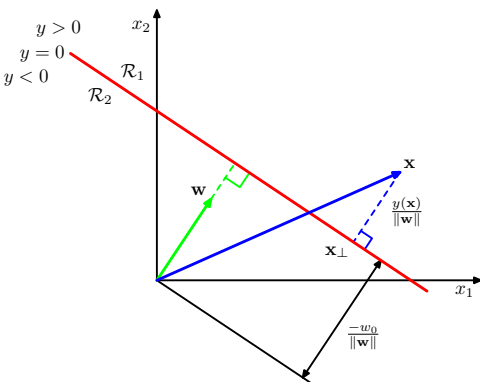
Outline

Discriminant Functions

Generative Models

Discriminative Models

Discriminant Functions with Two Classes



- Start with 2 class problem, $t_i \in \{0, 1\}$
- Simple linear discriminant

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

apply threshold function to get classification

- Decision surface is line; orthogonal to \mathbf{w} .
- Projection of \mathbf{x} in \mathbf{w} dir. is $\frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|}$

Multiple Classes

- A linear discriminant between two classes separates with a hyperplane
- How to use this for multiple classes?
- **One-versus-the-rest** method: build $K - 1$ classifiers, between \mathcal{C}_k and all others
- **One-versus-one** method: build $K(K - 1)/2$ classifiers, between all pairs

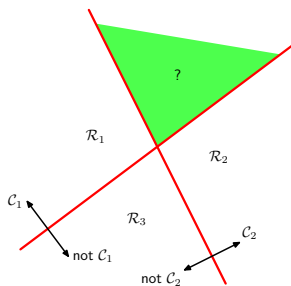
Multiple Classes

- A linear discriminant between two classes separates with a hyperplane
- How to use this for multiple classes?
- **One-versus-the-rest** method: build $K - 1$ classifiers, between \mathcal{C}_k and all others
- **One-versus-one** method: build $K(K - 1)/2$ classifiers, between all pairs

Multiple Classes

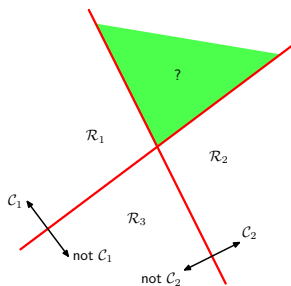
- A linear discriminant between two classes separates with a hyperplane
- How to use this for multiple classes?
- **One-versus-the-rest** method: build $K - 1$ classifiers, between \mathcal{C}_k and all others
- **One-versus-one** method: build $K(K - 1)/2$ classifiers, between all pairs

Multiple Classes



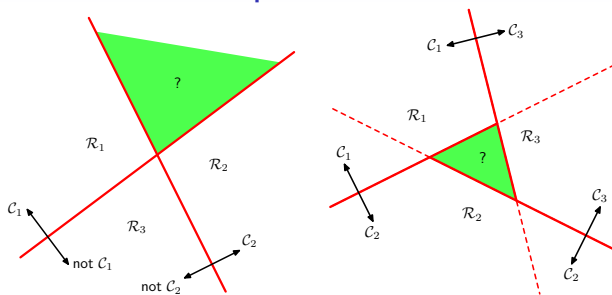
- A linear discriminant between two classes separates with a hyperplane
- How to use this for multiple classes?
- **One-versus-the-rest** method: build $K - 1$ classifiers, between \mathcal{C}_k and all others
- **One-versus-one** method: build $K(K - 1)/2$ classifiers, between all pairs

Multiple Classes



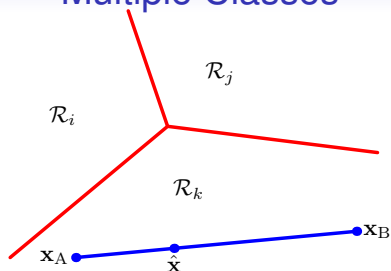
- A linear discriminant between two classes separates with a hyperplane
- How to use this for multiple classes?
- **One-versus-the-rest** method: build $K - 1$ classifiers, between \mathcal{C}_k and all others
- **One-versus-one** method: build $K(K - 1)/2$ classifiers, between all pairs

Multiple Classes



- A linear discriminant between two classes separates with a hyperplane
- How to use this for multiple classes?
- **One-versus-the-rest** method: build $K - 1$ classifiers, between C_k and all others
- **One-versus-one** method: build $K(K - 1)/2$ classifiers, between all pairs

Multiple Classes



- A solution is to build K linear functions:

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

assign \mathbf{x} to class $\max_k y_k(\mathbf{x})$

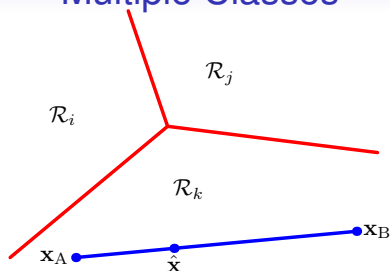
- Gives connected, convex decision regions

$$\hat{\mathbf{x}} = \lambda \mathbf{x}_A + (1 - \lambda) \mathbf{x}_B$$

$$y_k(\hat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_A) + (1 - \lambda) y_k(\mathbf{x}_B)$$

$$\Rightarrow y_k(\hat{\mathbf{x}}) > y_j(\hat{\mathbf{x}}), \forall j \neq k$$

Multiple Classes



- A solution is to build K linear functions:

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

assign \mathbf{x} to class $\max_k y_k(\mathbf{x})$

- Gives connected, convex **decision regions**

$$\hat{\mathbf{x}} = \lambda \mathbf{x}_A + (1 - \lambda) \mathbf{x}_B$$

$$y_k(\hat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_A) + (1 - \lambda) y_k(\mathbf{x}_B)$$

$$\Rightarrow y_k(\hat{\mathbf{x}}) > y_j(\hat{\mathbf{x}}), \forall j \neq k$$

Least Squares for Classification

- How do we learn the decision boundaries (w_k, w_{k0})?
- One approach is to use least squares, similar to regression
- Find W to minimize squared error over all examples and all components of the label vector:

$$E(W) = \frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K (y_k(\mathbf{x}_n) - t_{nk})^2$$

- Some algebra, we get a solution using the pseudo-inverse as in regression

Least Squares for Classification

- How do we learn the decision boundaries (w_k, w_{k0})?
- One approach is to use least squares, similar to regression
- Find W to minimize squared error over all examples and all components of the label vector:

$$E(W) = \frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K (y_k(\mathbf{x}_n) - t_{nk})^2$$

- Some algebra, we get a solution using the pseudo-inverse as in regression

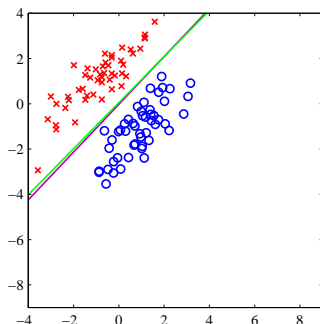
Least Squares for Classification

- How do we learn the decision boundaries (w_k, w_{k0})?
- One approach is to use least squares, similar to regression
- Find W to minimize squared error over all examples and all components of the label vector:

$$E(W) = \frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K (y_k(\mathbf{x}_n) - t_{nk})^2$$

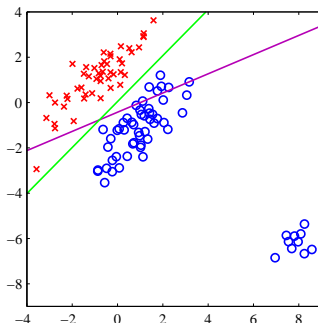
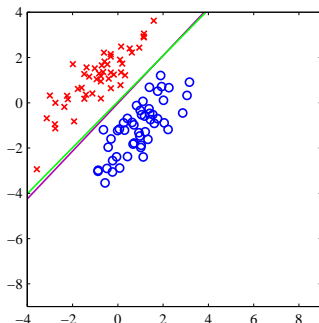
- Some algebra, we get a solution using the pseudo-inverse as in regression

Problems with Least Squares



- Looks okay... least squares decision boundary
 - Similar to logistic regression decision boundary (more later)

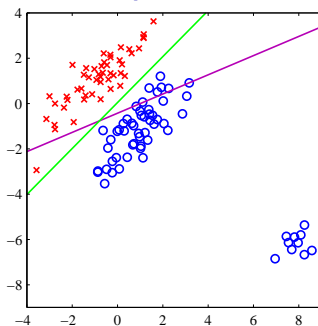
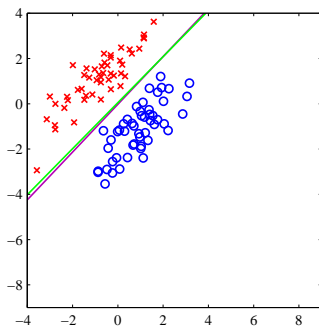
Problems with Least Squares



- Looks okay... least squares decision boundary
 - Similar to logistic regression decision boundary (more later)

- Gets worse by adding easy points?!

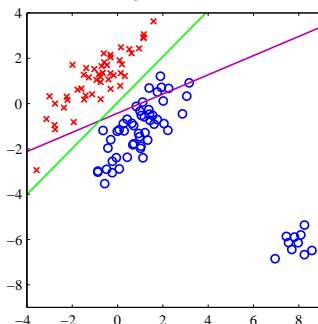
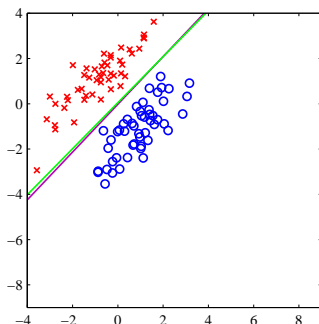
Problems with Least Squares



- Looks okay... least squares decision boundary
 - Similar to logistic regression decision boundary (more later)

- Gets worse by adding easy points?!
- Why?

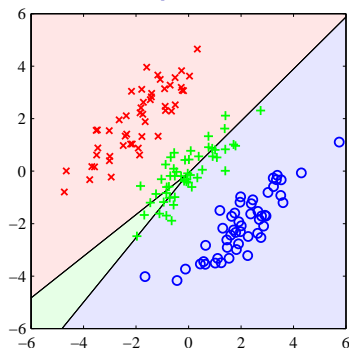
Problems with Least Squares



- Looks okay... **least squares decision boundary**
 - Similar to **logistic regression decision boundary** (more later)

- Gets worse by adding easy points?!
- Why?
 - If target value is 1, points far from boundary will have high value, say 10; this is a large error so the boundary is moved

More Least Squares Problems



- Easily separated by hyperplanes, but not found using least squares!
- Remember that least squares is MLE under Gaussian noise model for continuous target - we've got discrete targets.
- We'll address these problems later with better models
- First, a look at a different criterion for linear discriminant

Fisher's Linear Discriminant

- The two-class linear discriminant acts as a projection:

$$y = \mathbf{w}^T \mathbf{x}$$

Fisher's Linear Discriminant

- The two-class linear discriminant acts as a projection:

$$y = \mathbf{w}^T \mathbf{x} \geq -w_0$$

followed by a threshold

Fisher's Linear Discriminant

- The two-class linear discriminant acts as a projection:

$$y = \mathbf{w}^T \mathbf{x} \geq -w_0$$

followed by a threshold

- In which direction \mathbf{w} should we project?

Fisher's Linear Discriminant

- The two-class linear discriminant acts as a projection:

$$y = \mathbf{w}^T \mathbf{x} \geq -w_0$$

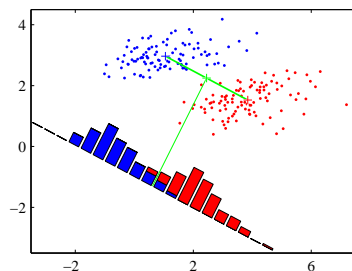
followed by a threshold

- In which direction \mathbf{w} should we project?
- One which separates classes “well”
- Intuition: We want the (projected) centers of the classes to be far apart, and each class (projection) to be clustered around its centre.

Fisher's Linear Discriminant

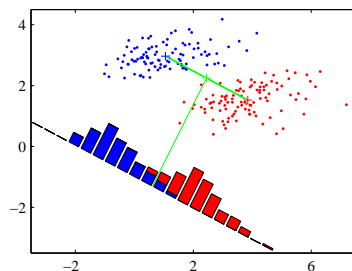
- A natural idea would be to project in the direction of the line connecting class means

Fisher's Linear Discriminant



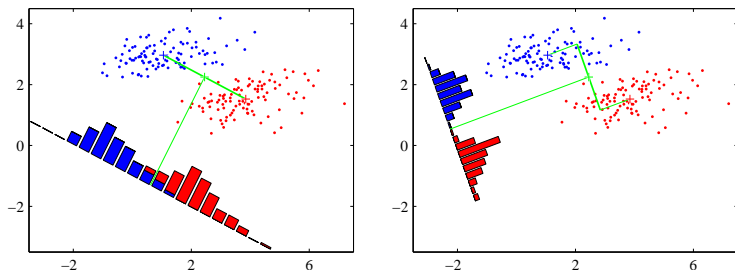
- A natural idea would be to project in the direction of the line connecting class means

Fisher's Linear Discriminant



- A natural idea would be to project in the direction of the line connecting class means
- However, problematic if classes have variance in this direction (ie are not clustered around the mean)

Fisher's Linear Discriminant



- A natural idea would be to project in the direction of the line connecting class means
- However, problematic if classes have variance in this direction (ie are not clustered around the mean)
- Fisher criterion: maximize ratio of inter-class separation (between) to intra-class variance (inside)

Math time - FLD

- Projection $y_n = \mathbf{w}^T \mathbf{x}_n$
- Inter-class separation is distance between class means (good):

$$m_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{w}^T \mathbf{x}_n$$

- Intra-class variance (bad):

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$$

- Fisher criterion:

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

maximize wrt \mathbf{w}

Math time - FLD

- Projection $y_n = \mathbf{w}^T \mathbf{x}_n$
- Inter-class separation is distance between class means (good):

$$m_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{w}^T \mathbf{x}_n$$

- Intra-class variance (bad):

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$$

- Fisher criterion:

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

maximize wrt \mathbf{w}

Math time - FLD

- Projection $y_n = \mathbf{w}^T \mathbf{x}_n$
- Inter-class separation is distance between class means (good):

$$m_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{w}^T \mathbf{x}_n$$

- Intra-class variance (bad):

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$$

- Fisher criterion:

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

maximize wrt \mathbf{w}

Math time - FLD

- Projection $y_n = \mathbf{w}^T \mathbf{x}_n$
- Inter-class separation is distance between class means (good):

$$m_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{w}^T \mathbf{x}_n$$

- Intra-class variance (bad):

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$$

- Fisher criterion:

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

maximize wrt \mathbf{w}

Math time - FLD

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

Between-class covariance:

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

Within-class covariance:

$$\mathbf{S}_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

Lots of math:

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

If covariance \mathbf{S}_W is isotropic (proportional to unit matrix, so little variance within class), reduces to class mean difference vector

Math time - FLD

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

Between-class covariance:

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

Within-class covariance:

$$\mathbf{S}_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

Lots of math:

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

If covariance \mathbf{S}_W is isotropic (proportional to unit matrix, so little variance within class), reduces to class mean difference vector

FLD Summary

- FLD is a **dimensionality reduction** technique (more later in the course)
- Criterion for choosing projection based on class labels
 - Still suffers from outliers (e.g. earlier least squares example)

Perceptrons

- **Perceptrons** is used to refer to many neural network structures (more next week)
- The classic type is a fixed non-linear transformation of input, one layer of adaptive weights, and a threshold:

$$y(\mathbf{x}) = f(\mathbf{w}^T \phi(\mathbf{x}))$$

- Developed by Rosenblatt in the 50s
- The main difference compared to the methods we've seen so far is the learning algorithm

Perceptron Learning

- Two class problem
- For ease of notation, we will use $t = 1$ for class \mathcal{C}_1 and $t = -1$ for class \mathcal{C}_2
- We saw that squared error was problematic
- Instead, we'd like to minimize the number of misclassified examples
 - An example is mis-classified if $\mathbf{w}^T \phi(\mathbf{x}_n) t_n < 0$
 - Perceptron criterion:

$$E_P(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^T \phi(\mathbf{x}_n) t_n$$

sum over mis-classified examples only

Perceptron Learning

- Two class problem
- For ease of notation, we will use $t = 1$ for class \mathcal{C}_1 and $t = -1$ for class \mathcal{C}_2
- We saw that squared error was problematic
- Instead, we'd like to minimize the number of misclassified examples
 - An example is mis-classified if $\mathbf{w}^T \phi(\mathbf{x}_n) t_n < 0$
 - Perceptron criterion:

$$E_P(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^T \phi(\mathbf{x}_n) t_n$$

sum over mis-classified examples only

Perceptron Learning

- Two class problem
- For ease of notation, we will use $t = 1$ for class \mathcal{C}_1 and $t = -1$ for class \mathcal{C}_2
- We saw that squared error was problematic
- Instead, we'd like to minimize the number of misclassified examples
 - An example is mis-classified if $\mathbf{w}^T \phi(\mathbf{x}_n) t_n < 0$
 - Perceptron criterion:

$$E_P(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^T \phi(\mathbf{x}_n) t_n$$

sum over mis-classified examples only

Perceptron Learning

- Two class problem
- For ease of notation, we will use $t = 1$ for class \mathcal{C}_1 and $t = -1$ for class \mathcal{C}_2
- We saw that squared error was problematic
- Instead, we'd like to minimize the number of misclassified examples
 - An example is mis-classified if $\mathbf{w}^T \phi(\mathbf{x}_n) t_n < 0$
 - **Perceptron criterion:**

$$E_P(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^T \phi(\mathbf{x}_n) t_n$$

sum over mis-classified examples only

Perceptron Learning Algorithm

- Minimize the error function using stochastic gradient descent (gradient descent per example):

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w})$$

Perceptron Learning Algorithm

- Minimize the error function using stochastic gradient descent (gradient descent per example):

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \underbrace{\eta \phi(\mathbf{x}_n) t_n}_{\text{if incorrect}}$$

- Iterate over all training examples, only change \mathbf{w} if the example is mis-classified

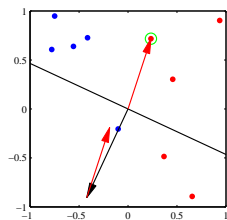
Perceptron Learning Algorithm

- Minimize the error function using stochastic gradient descent (gradient descent per example):

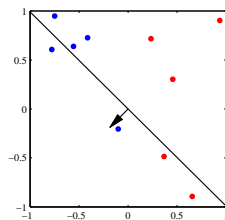
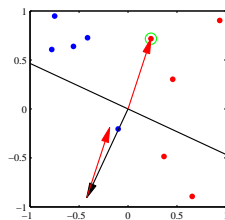
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \underbrace{\eta \phi(\mathbf{x}_n) t_n}_{\text{if incorrect}}$$

- Iterate over all training examples, only change \mathbf{w} if the example is mis-classified
- Guaranteed to converge if data are **linearly separable**
- Will not converge if not
- May take many iterations
- Sensitive to initialization

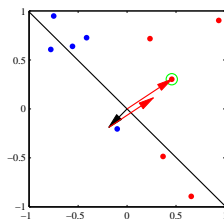
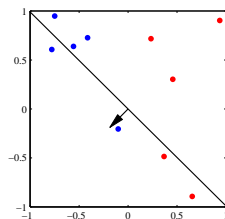
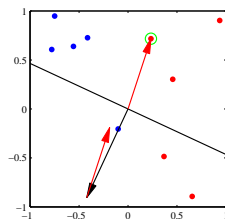
Perceptron Learning Illustration



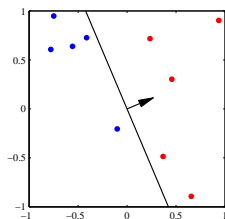
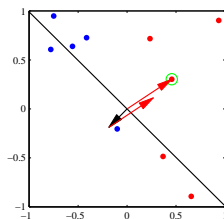
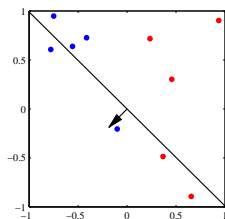
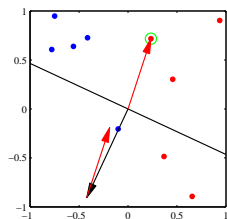
Perceptron Learning Illustration



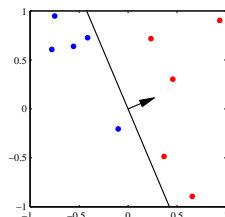
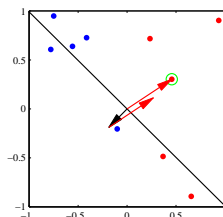
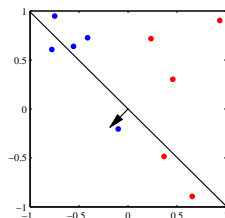
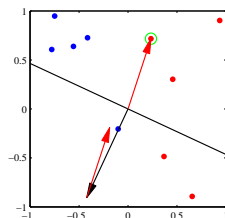
Perceptron Learning Illustration



Perceptron Learning Illustration



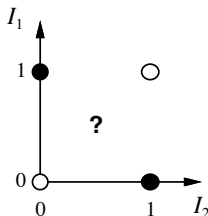
Perceptron Learning Illustration



- Note there are many hyperplanes with 0 error
 - Support vector machines (in a few weeks) have a nice way of choosing one

Limitations of Perceptrons

- Perceptrons can only solve linearly separable problems in feature space
 - Same as the other models in this chapter
- Canonical example of non-separable problem is X-OR
 - Real datasets can look like this too



Outline

Discriminant Functions

Generative Models

Discriminative Models

Intuition for Logistic Regression in Generative Models

- Classification with Joint Probabilities With 2 classes, \mathcal{C}_1 and \mathcal{C}_2 :
- Choose \mathcal{C}_1 if

$$1 < \frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})} = \frac{p(\mathcal{C}_1, \mathbf{x})}{p(\mathcal{C}_2, \mathbf{x})} \iff$$

$$0 < \ln \left(\frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})} \right) = \ln(p(\mathcal{C}_1|\mathbf{x})) - \ln(p(\mathcal{C}_2|\mathbf{x}))$$

- The quantity

$$a = \ln \left(\frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})} \right)$$

is called the **log-odds**.

- **Logistic Regression** assumes that the log-odds are a linear function of the feature vector: $a = \mathbf{w}^T \mathbf{x} + w_0$.
- Often true given assumptions about the **class-conditional densities** $p(\mathbf{x}|\mathcal{C}_k)$.

Intuition for Logistic Regression in Generative Models

- Classification with Joint Probabilities With 2 classes, \mathcal{C}_1 and \mathcal{C}_2 :
- Choose \mathcal{C}_1 if

$$1 < \frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})} = \frac{p(\mathcal{C}_1, \mathbf{x})}{p(\mathcal{C}_2, \mathbf{x})} \iff$$

$$0 < \ln\left(\frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})}\right) = \ln(p(\mathcal{C}_1|\mathbf{x})) - \ln(p(\mathcal{C}_2|\mathbf{x}))$$

- The quantity

$$a = \ln\left(\frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})}\right)$$

is called the **log-odds**.

- **Logistic Regression** assumes that the log-odds are a linear function of the feature vector: $a = \mathbf{w}^T \mathbf{x} + w_0$.
- Often true given assumptions about the **class-conditional densities** $p(\mathbf{x}|\mathcal{C}_k)$.

Intuition for Logistic Regression in Generative Models

- Classification with Joint Probabilities With 2 classes, \mathcal{C}_1 and \mathcal{C}_2 :
- Choose \mathcal{C}_1 if

$$1 < \frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})} = \frac{p(\mathcal{C}_1, \mathbf{x})}{p(\mathcal{C}_2, \mathbf{x})} \iff$$

$$0 < \ln\left(\frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})}\right) = \ln(p(\mathcal{C}_1|\mathbf{x})) - \ln(p(\mathcal{C}_2|\mathbf{x}))$$

- The quantity

$$a = \ln\left(\frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})}\right)$$

is called the **log-odds**.

- **Logistic Regression** assumes that the log-odds are a linear function of the feature vector: $a = \mathbf{w}^T \mathbf{x} + w_0$.
- Often true given assumptions about the **class-conditional densities** $p(\mathbf{x}|\mathcal{C}_k)$.

Intuition for Logistic Regression in Generative Models

- Classification with Joint Probabilities With 2 classes, \mathcal{C}_1 and \mathcal{C}_2 :
- Choose \mathcal{C}_1 if

$$1 < \frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})} = \frac{p(\mathcal{C}_1, \mathbf{x})}{p(\mathcal{C}_2, \mathbf{x})} \iff$$

$$0 < \ln\left(\frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})}\right) = \ln(p(\mathcal{C}_1|\mathbf{x})) - \ln(p(\mathcal{C}_2|\mathbf{x}))$$

- The quantity

$$a = \ln\left(\frac{p(\mathcal{C}_1|\mathbf{x})}{p(\mathcal{C}_2|\mathbf{x})}\right)$$

is called the **log-odds**.

- **Logistic Regression** assumes that the log-odds are a linear function of the feature vector: $a = \mathbf{w}^T \mathbf{x} + w_0$.
- Often true given assumptions about the **class-conditional densities** $p(\mathbf{x}|\mathcal{C}_k)$.

Probabilistic Generative Models

- With 2 classes, \mathcal{C}_1 and \mathcal{C}_2 :

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x})} \text{ Bayes' Rule}$$

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}, \mathcal{C}_1) + p(\mathbf{x}, \mathcal{C}_2)} \text{ Sum rule}$$

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ Product rule}$$

- In **generative models** we specify the class-conditional distribution $p(\mathbf{x}|\mathcal{C}_k)$ which generates the data for each class

Probabilistic Generative Models

- With 2 classes, \mathcal{C}_1 and \mathcal{C}_2 :

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x})} \text{ Bayes' Rule}$$

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}, \mathcal{C}_1) + p(\mathbf{x}, \mathcal{C}_2)} \text{ Sum rule}$$

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ Product rule}$$

- In **generative models** we specify the class-conditional distribution $p(\mathbf{x}|\mathcal{C}_k)$ which generates the data for each class

Probabilistic Generative Models

- With 2 classes, \mathcal{C}_1 and \mathcal{C}_2 :

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x})} \text{ Bayes' Rule}$$

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}, \mathcal{C}_1) + p(\mathbf{x}, \mathcal{C}_2)} \text{ Sum rule}$$

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ Product rule}$$

- In **generative models** we specify the class-conditional distribution $p(\mathbf{x}|\mathcal{C}_k)$ which generates the data for each class

Probabilistic Generative Models

- With 2 classes, \mathcal{C}_1 and \mathcal{C}_2 :

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x})} \text{ Bayes' Rule}$$

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}, \mathcal{C}_1) + p(\mathbf{x}, \mathcal{C}_2)} \text{ Sum rule}$$

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \text{ Product rule}$$

- In **generative models** we specify the class-conditional distribution $p(\mathbf{x}|\mathcal{C}_k)$ which generates the data for each class

Probabilistic Generative Models - Example

- Let's say we observe x which is the current temperature
- Determine if we are in Vancouver (\mathcal{C}_1) or Honolulu (\mathcal{C}_2)
- Generative model:

$$p(\mathcal{C}_1|x) = \frac{p(x|\mathcal{C}_1)p(\mathcal{C}_1)}{p(x|\mathcal{C}_1)p(\mathcal{C}_1) + p(x|\mathcal{C}_2)p(\mathcal{C}_2)}$$

- $p(x|\mathcal{C}_1)$ is distribution over typical temperatures in Vancouver
 - e.g. $p(x|\mathcal{C}_1) = \mathcal{N}(x; 10, 5)$
- $p(x|\mathcal{C}_2)$ is distribution over typical temperatures in Honolulu
 - e.g. $p(x|\mathcal{C}_2) = \mathcal{N}(x; 25, 5)$
- Class priors $p(\mathcal{C}_1) = 0.1, p(\mathcal{C}_2) = 0.9$
- $p(\mathcal{C}_1|x = 15) = \frac{0.0484 \cdot 0.1}{0.0484 \cdot 0.1 + 0.0108 \cdot 0.9} \approx 0.33$

Probabilistic Generative Models - Example

- Let's say we observe x which is the current temperature
- Determine if we are in Vancouver (\mathcal{C}_1) or Honolulu (\mathcal{C}_2)
- Generative model:

$$p(\mathcal{C}_1|x) = \frac{p(x|\mathcal{C}_1)p(\mathcal{C}_1)}{p(x|\mathcal{C}_1)p(\mathcal{C}_1) + p(x|\mathcal{C}_2)p(\mathcal{C}_2)}$$

- $p(x|\mathcal{C}_1)$ is distribution over typical temperatures in Vancouver
 - e.g. $p(x|\mathcal{C}_1) = \mathcal{N}(x; 10, 5)$
- $p(x|\mathcal{C}_2)$ is distribution over typical temperatures in Honolulu
 - e.g. $p(x|\mathcal{C}_2) = \mathcal{N}(x; 25, 5)$
 - Class priors $p(\mathcal{C}_1) = 0.1, p(\mathcal{C}_2) = 0.9$
- $p(\mathcal{C}_1|x = 15) = \frac{0.0484 \cdot 0.1}{0.0484 \cdot 0.1 + 0.0108 \cdot 0.9} \approx 0.33$

Generalized Linear Models

- Suppose we have built a model for predicting the log-odds. We can use it to compute the class probability as follows.

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \exp(-a)} \equiv \sigma(a) \end{aligned}$$

$$\begin{aligned} \text{where } a &= \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \ln \frac{p(\mathbf{x}, \mathcal{C}_1)}{p(\mathbf{x}, \mathcal{C}_2)}. \end{aligned}$$

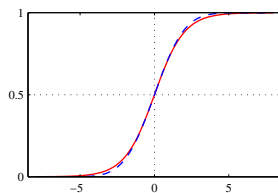
Generalized Linear Models

- Suppose we have built a model for predicting the log-odds. We can use it to compute the class probability as follows.

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \exp(-a)} \equiv \sigma(a) \end{aligned}$$

$$\begin{aligned} \text{where } a &= \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \ln \frac{p(\mathbf{x}, \mathcal{C}_1)}{p(\mathbf{x}, \mathcal{C}_2)}. \end{aligned}$$

Logistic Sigmoid



- The function $\sigma(a) = \frac{1}{1 + \exp(-a)}$ is known as the logistic sigmoid
- It squashes the real axis down to $[0, 1]$
- It is continuous and differentiable

Multi-class Extension

- There is a generalization of the logistic sigmoid to $K > 2$ classes:

$$\begin{aligned} p(C_k|\mathbf{x}) &= \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \\ \text{where } a_k &= \ln p(\mathbf{x}|C_k)p(C_k) \end{aligned}$$

- a. k. a. **softmax function**
 - If some $a_k \gg a_j$, $p(C_k|\mathbf{x})$ goes to 1

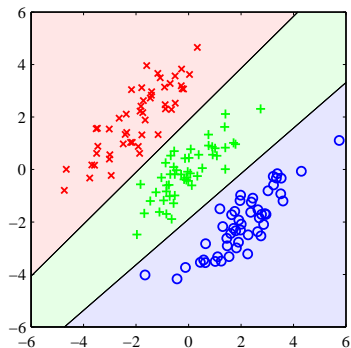
Multi-class Extension

- There is a generalization of the logistic sigmoid to $K > 2$ classes:

$$\begin{aligned} p(C_k|\mathbf{x}) &= \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \\ \text{where } a_k &= \ln p(\mathbf{x}|C_k)p(C_k) \end{aligned}$$

- a. k. a. **softmax function**
 - If some $a_k \gg a_j$, $p(C_k|\mathbf{x})$ goes to 1

Example Logistic Regression



Gaussian Class-Conditional Densities

- Back to the log-odds a in the logistic sigmoid for 2 classes
- Let's assume the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ are Gaussians, and have the same covariance matrix Σ :

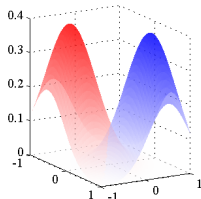
$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) \right\}$$

- a takes a simple form:

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} = \mathbf{w}^T \mathbf{x} + w_0$$

- Note that quadratic terms $\mathbf{x}^T \Sigma^{-1} \mathbf{x}$ cancel

Gaussian Class-Conditional Densities



- Back to the log-odds a in the logistic sigmoid for 2 classes
- Let's assume the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ are Gaussians, and have the same covariance matrix Σ :

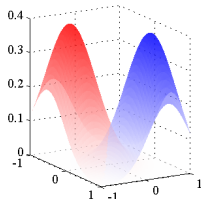
$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) \right\}$$

- a takes a simple form:

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} = \mathbf{w}^T \mathbf{x} + w_0$$

- Note that quadratic terms $\mathbf{x}^T \Sigma^{-1} \mathbf{x}$ cancel

Gaussian Class-Conditional Densities



- Back to the log-odds a in the logistic sigmoid for 2 classes
- Let's assume the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ are Gaussians, and have the same covariance matrix Σ :

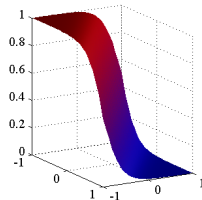
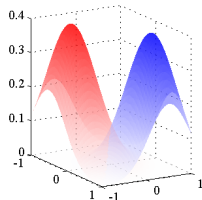
$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) \right\}$$

- a takes a simple form:

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} = \mathbf{w}^T \mathbf{x} + w_0$$

- Note that quadratic terms $\mathbf{x}^T \Sigma^{-1} \mathbf{x}$ cancel

Gaussian Class-Conditional Densities



- Back to the log-odds a in the logistic sigmoid for 2 classes
- Let's assume the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ are Gaussians, and have the same covariance matrix Σ :

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) \right\}$$

- a takes a simple form:

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} = \mathbf{w}^T \mathbf{x} + w_0$$

- Note that quadratic terms $\mathbf{x}^T \Sigma^{-1} \mathbf{x}$ cancel

Maximum Likelihood Learning

- We can fit the parameters to this model using **maximum likelihood**
 - Parameters are $\mu_1, \mu_2, \Sigma^{-1}, p(\mathcal{C}_1) \equiv \pi, p(\mathcal{C}_2) \equiv 1 - \pi$
 - Refer to as θ
- For a datapoint x_n from class \mathcal{C}_1 ($t_n = 1$):

$$p(x_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(x_n|\mathcal{C}_1) = \pi\mathcal{N}(x_n|\mu_1, \Sigma)$$

- For a datapoint x_n from class \mathcal{C}_2 ($t_n = 0$):

$$p(x_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(x_n|\mathcal{C}_2) = (1 - \pi)\mathcal{N}(x_n|\mu_2, \Sigma)$$

Maximum Likelihood Learning

- We can fit the parameters to this model using **maximum likelihood**
 - Parameters are $\mu_1, \mu_2, \Sigma^{-1}, p(C_1) \equiv \pi, p(C_2) \equiv 1 - \pi$
 - Refer to as θ
- For a datapoint \mathbf{x}_n from class C_1 ($t_n = 1$):

$$p(\mathbf{x}_n, C_1) = p(C_1)p(\mathbf{x}_n|C_1) = \pi\mathcal{N}(\mathbf{x}_n|\mu_1, \Sigma)$$

- For a datapoint \mathbf{x}_n from class C_2 ($t_n = 0$):

$$p(\mathbf{x}_n, C_2) = p(C_2)p(\mathbf{x}_n|C_2) = (1 - \pi)\mathcal{N}(\mathbf{x}_n|\mu_2, \Sigma)$$

Maximum Likelihood Learning

- We can fit the parameters to this model using **maximum likelihood**
 - Parameters are $\mu_1, \mu_2, \Sigma^{-1}, p(\mathcal{C}_1) \equiv \pi, p(\mathcal{C}_2) \equiv 1 - \pi$
 - Refer to as θ
- For a datapoint \mathbf{x}_n from class \mathcal{C}_1 ($t_n = 1$):

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\mathbf{x}_n|\mu_1, \Sigma)$$

- For a datapoint \mathbf{x}_n from class \mathcal{C}_2 ($t_n = 0$):

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1 - \pi)\mathcal{N}(\mathbf{x}_n|\mu_2, \Sigma)$$

Maximum Likelihood Learning

- The likelihood of the training data is:

$$p(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^N [\pi \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n}$$

- As usual, \ln is our friend:

$$\ell(\mathbf{t}; \theta) = \sum_{n=1}^N \underbrace{t_n \ln \pi + (1 - t_n) \ln(1 - \pi)}_{\pi} + \underbrace{t_n \ln N_1 + (1 - t_n) \ln N_2}_{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}}$$

- Maximize for each separately

Maximum Likelihood Learning

- The likelihood of the training data is:

$$p(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^N [\pi \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n}$$

- As usual, \ln is our friend:

$$\ell(\mathbf{t}; \theta) = \sum_{n=1}^N \underbrace{t_n \ln \pi + (1 - t_n) \ln(1 - \pi)}_{\pi} + \underbrace{t_n \ln N_1 + (1 - t_n) \ln N_2}_{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}}$$

- Maximize for each separately

Maximum Likelihood Learning - Class Priors

- Maximization with respect to the class priors parameter π is straightforward:

$$\frac{\partial}{\partial \pi} \ell(\mathbf{t}; \theta) = \sum_{n=1}^N \frac{t_n}{\pi} - \frac{1 - t_n}{1 - \pi}$$

$$\Rightarrow \pi = \frac{N_1}{N_1 + N_2}$$

- N_1 and N_2 are the number of training points in each class
- Prior is simply the fraction of points in each class

Maximum Likelihood Learning - Class Priors

- Maximization with respect to the class priors parameter π is straightforward:

$$\frac{\partial}{\partial \pi} \ell(\mathbf{t}; \theta) = \sum_{n=1}^N \frac{t_n}{\pi} - \frac{1 - t_n}{1 - \pi}$$

$$\Rightarrow \pi = \frac{N_1}{N_1 + N_2}$$

- N_1 and N_2 are the number of training points in each class
- Prior is simply the fraction of points in each class

Maximum Likelihood Learning - Class Priors

- Maximization with respect to the class priors parameter π is straightforward:

$$\frac{\partial}{\partial \pi} \ell(\mathbf{t}; \theta) = \sum_{n=1}^N \frac{t_n}{\pi} - \frac{1 - t_n}{1 - \pi}$$

$$\Rightarrow \pi = \frac{N_1}{N_1 + N_2}$$

- N_1 and N_2 are the number of training points in each class
- Prior is simply the fraction of points in each class

Maximum Likelihood Learning - Gaussian Parameters

- The other parameters can also be found in the same fashion
- Class means:

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^N t_n \mathbf{x}_n$$

$$\boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n=1}^N (1 - t_n) \mathbf{x}_n$$

- Means of training examples from each class
- Shared covariance matrix:

$$\boldsymbol{\Sigma} = \frac{N_1}{N} \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T + \frac{N_2}{N} \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T$$

- Weighted average of class covariances

Maximum Likelihood Learning - Gaussian Parameters

- The other parameters can also be found in the same fashion
- Class means:

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^N t_n \mathbf{x}_n$$

$$\boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n=1}^N (1 - t_n) \mathbf{x}_n$$

- Means of training examples from each class
- Shared covariance matrix:

$$\boldsymbol{\Sigma} = \frac{N_1}{N} \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T + \frac{N_2}{N} \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T$$

- Weighted average of class covariances

Probabilistic Generative Models Summary

- Fitting Gaussian using ML criterion is sensitive to outliers
- Simple linear form for a in logistic sigmoid occurs for more than just Gaussian distributions
 - Arises for any distribution in the [exponential family](#), a large class of distributions

Outline

Discriminant Functions

Generative Models

Discriminative Models

Probabilistic Discriminative Models

- Generative model made assumptions about form of class-conditional distributions (e.g. Gaussian)
 - Resulted in logistic sigmoid of linear function of x
- Discriminative model - explicitly use functional form

$$p(\mathcal{C}_1|x) = \frac{1}{1 + \exp(-w^T x + w_0)}$$

and find w directly

- For the generative model we had $2M + M(M + 1)/2 + 1$ parameters
 - M is dimensionality of x
- Discriminative model will have $M + 1$ parameters

Probabilistic Discriminative Models

- Generative model made assumptions about form of class-conditional distributions (e.g. Gaussian)
 - Resulted in logistic sigmoid of linear function of \mathbf{x}
- Discriminative model - explicitly use functional form

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x} + w_0)}$$

and find \mathbf{w} directly

- For the generative model we had $2M + M(M + 1)/2 + 1$ parameters
 - M is dimensionality of \mathbf{x}
- Discriminative model will have $M + 1$ parameters

Probabilistic Discriminative Models

- Generative model made assumptions about form of class-conditional distributions (e.g. Gaussian)
 - Resulted in logistic sigmoid of linear function of \mathbf{x}
- Discriminative model - explicitly use functional form

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x} + w_0)}$$

and find \mathbf{w} directly

- For the generative model we had $2M + M(M + 1)/2 + 1$ parameters
 - M is dimensionality of \mathbf{x}
- Discriminative model will have $M + 1$ parameters

Maximum Likelihood Learning - Discriminative Model

- As usual we can use the maximum likelihood criterion for learning

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{1-t_n} ; \text{ where } y_n = p(C_1|\mathbf{x}_n)$$

- Taking ln and derivative gives:

$$\nabla \ell(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \mathbf{x}_n$$

- This time no closed-form solution since $y_n = \sigma(\mathbf{w}^T \mathbf{x})$
- Could use (stochastic) gradient descent
 - But Iterative Reweighted Least Squares (IRLS) is a better technique.

Maximum Likelihood Learning - Discriminative Model

- As usual we can use the maximum likelihood criterion for learning

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{1-t_n} ; \text{ where } y_n = p(C_1|\mathbf{x}_n)$$

- Taking ln and derivative gives:

$$\nabla \ell(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \mathbf{x}_n$$

- This time no closed-form solution since $y_n = \sigma(\mathbf{w}^T \mathbf{x})$
- Could use (stochastic) gradient descent
 - But Iterative Reweighted Least Squares (IRLS) is a better technique.

Maximum Likelihood Learning - Discriminative Model

- As usual we can use the maximum likelihood criterion for learning

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{1-t_n} ; \text{ where } y_n = p(C_1|\mathbf{x}_n)$$

- Taking ln and derivative gives:

$$\nabla \ell(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \mathbf{x}_n$$

- This time no closed-form solution since $y_n = \sigma(\mathbf{w}^T \mathbf{x})$
- Could use (stochastic) gradient descent
 - But Iterative Reweighted Least Squares (IRLS) is a better technique.

Maximum Likelihood Learning - Discriminative Model

- As usual we can use the maximum likelihood criterion for learning

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{1-t_n} ; \text{ where } y_n = p(C_1|\mathbf{x}_n)$$

- Taking ln and derivative gives:

$$\nabla \ell(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \mathbf{x}_n$$

- This time no closed-form solution since $y_n = \sigma(\mathbf{w}^T \mathbf{x})$
- Could use (stochastic) gradient descent
 - But Iterative Reweighted Least Squares (IRLS) is a better technique.

Generative vs. Discriminative

- Generative models
 - Can generate synthetic example data
 - Perhaps accurate classification is equivalent to accurate synthesis
 - Support learning with missing data
 - Tend to have more parameters
 - Require good model of class distributions
- Discriminative models
 - Only usable for classification
 - Don't solve a harder problem than you need to
 - Tend to have fewer parameters
 - Require good model of decision boundary

Conclusion

- Readings: Ch. 4.1.1-4.1.4, 4.1.7, 4.2.1-4.2.2, 4.3.1-4.3.3
- Generalized linear models $y(\mathbf{x}) = f(\mathbf{w}^T \mathbf{x} + w_0)$
- Threshold/max function for $f(\cdot)$
 - Minimize with least squares
 - Fisher criterion - class separation
 - Perceptron criterion - mis-classified examples
- Probabilistic models: logistic sigmoid / softmax for $f(\cdot)$
 - Generative model - assume class conditional densities in exponential family; obtain sigmoid
 - Discriminative model - directly model posterior using sigmoid (a. k. a. **logistic regression**, though classification)
 - Can learn either using maximum likelihood
- **All of these models are limited to linear decision boundaries in feature space**