LECTURE 2: Linear and quadratic classifiers

- Part 1: Bayesian Decision Theory
  - The Likelihood Ratio Test
  - Maximum A Posteriori and Maximum Likelihood
  - Discriminant functions

- Part 2: Quadratic classifiers
  - Bayes classifiers for normally distributed classes
  - Euclidean and Mahalanobis distance classifiers
  - Numerical example

- Part 3: Linear classifiers
  - Gradient descent
  - The perceptron rule
  - The pseudo-inverse solution
  - Least mean squares
Part 1: Bayesian Decision Theory
The Likelihood Ratio Test (1)

- Assume we are to classify an object based on the evidence provided by a measurement (or feature vector) $x$

- Would you agree that a reasonable decision rule would be the following?
  
  - "Choose the class that is most ‘probable’ given the observed feature vector $x$"
  
  - More formally: Evaluate the posterior probability of each class $P(\omega_i|x)$ and choose the class with largest $P(\omega_i|x)$
The Likelihood Ratio Test (2)

Let us examine this decision rule for a two-class problem

- In this case the decision rule becomes

\[
\text{if } P(\omega_1 | x) > P(\omega_2 | x) \quad \text{choose } \omega_1 \\
\text{else } \quad \text{choose } \omega_2
\]

- Or, in a more compact form

\[
P(\omega_1 | x) \geq P(\omega_2 | x)
\]

- Applying Bayes theorem

\[
\frac{P(x | \omega_1)P(\omega_1)}{P(x)} \geq \frac{P(x | \omega_2)P(\omega_2)}{P(x)}
\]
The Likelihood Ratio Test (3)

- P(x) does not affect the decision rule so it can be eliminated*. Rearranging the previous expression

\[
\Lambda(x) = \frac{P(x \mid \omega_1)}{P(x \mid \omega_2)} \frac{\omega_1}{\omega_2} \frac{P(\omega_1)}{P(\omega_2)}
\]

- The term \( \Lambda(x) \) is called the **likelihood ratio**, and the decision rule is known as the **likelihood ratio test**
**Likelihood Ratio Test: an example (1)**

- Given a classification problem with the following class conditional densities:

\[
P(x \mid \omega_1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-4)^2}
\]

\[
P(x \mid \omega_2) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-10)^2}
\]

- Derive a classification rule based on the Likelihood Ratio Test (assume equal priors)
**Likelihood Ratio Test: an example (2)**

**Solution**

- Substituting the given likelihoods and priors into the LRT expression:

\[
\Lambda(x) = \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-4)^2}}{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-10)^2}}
\]

- Simplifying, changing signs and taking logs:

\[
(x - 4)^2 - (x - 10)^2 < 0
\]

- Which yields:

\[
\begin{align*}
\omega_1 &< x < 7 < \omega_2 \\
\end{align*}
\]

- This LRT result makes intuitive sense since the likelihoods are identical and differ only in their mean value.
The probability of error

- The probability of error is “the probability of assigning x to the wrong class”
  - For a two-class problem, \( P[\text{error}|x] \) is simply

\[
P(\text{error} | x) = \begin{cases} 
P(\omega_1 | x) & \text{if we decide } \omega_2 \\
P(\omega_2 | x) & \text{if we decide } \omega_1 
\end{cases}
\]

- It makes sense that the classification rule be designed to minimize the average prob. of error \( P[\text{error}] \) across all possible values of x

\[
P(\text{error}) = \int_{-\infty}^{+\infty} P(\text{error}, x)dx = \int_{-\infty}^{+\infty} P(\text{error} | x)P(x)dx
\]

- To minimize \( P(\text{error}) \) we minimize the integrand \( P(\text{error}|x) \) at each \( x \): choose the class with maximum posterior \( P(\omega_i|x) \)

- This is called the **MAXIMUM A POSTERIORI (MAP) RULE**
Minimizing probability of error

- We “prove” the optimality of the MAP rule graphically
  - The right plot shows the posterior for each of the two classes
  - The bottom plots shows the P(error) for the MAP rule and an alternative decision rule
  - Which one has lower P(error) (color-filled area)?
The Bayes Risk (1)

- So far we have assumed that the penalty of misclassifying $\omega_1$ as $\omega_2$ is the same as the reciprocal
  - In general, this is not the case
  - Misclassifications in the fish sorting lead to different costs
  - Medical diagnostics errors are very asymmetric

- We capture this concept with a cost function $C_{ij}$
  - $C_{ij}$ represents the cost of choosing class $\omega_i$ when class $\omega_j$ is the true class

- And define the Bayes Risk as the expected value of the cost

\[
\mathcal{R} = E[C] = \sum_{i=1}^{2} \sum_{j=1}^{2} C_{ij} \cdot P[\text{choose } \omega_i \text{ and } x \in \omega_j] = \sum_{i=1}^{2} \sum_{j=1}^{2} C_{ij} \cdot P[x \in R_i | \omega_j] \cdot P[\omega_j]
\]
The Bayes Risk (2)

What is the decision rule that minimizes the Bayes Risk?

- It can be shown* that the minimum risk can be achieved by using the following decision rule:

\[
\frac{P(x | \omega_1)}{P(x | \omega_2)} = \frac{\omega_1}{\omega_2} \frac{C_{12} - C_{22}}{C_{21} - C_{11}} \frac{P[\omega_2]}{P[\omega_1]}
\]

- *For an intuitive proof visit my lecture notes at TAMU

Notice any similarities with the LRT?
The Bayes Risk: an example (1)

Consider a classification problem with two classes defined by the following likelihood functions:

\[
P(x | \omega_1) = \frac{1}{\sqrt{2\pi} \sqrt{3}} e^{\frac{1}{2}x^2/3} \\
P(x | \omega_2) = \frac{1}{\sqrt{2\pi}} e^{\frac{1}{2}(x-2)^2}
\]

What is the decision rule that minimizes \(P[\text{error}]\)?

- Assume \(P[\omega_1]=P[\omega_2]=0.5\), \(C_{11}=C_{22}=0\), \(C_{12}=1\) and \(C_{21}=3^{1/2}\)
The Bayes Risk: an example (2)

\[ \Lambda(x) = \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{3}} e^{-\frac{1}{2} \left( \frac{x^2}{3} \right)} > 1 \implies \]

\[ \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{(x-2)^2}{3} \right)} < \frac{1}{\omega_2} \implies \]

\[ e^{-\frac{1}{2} \left( \frac{(x-2)^2}{3} \right)} < \frac{1}{\omega_2} \implies \]

\[ -\frac{1}{2} \left( \frac{x^2}{3} + \frac{1}{2} (x-2)^2 \right) > 0 \implies \]

\[ -\frac{1}{2} \left( \frac{x^2}{3} \right) + \frac{1}{2} (x-2)^2 < \frac{1}{\omega_2} \]

\[ 2x^2 - 12x + 12 > 0 \implies x = 4.73, 1.27 \]
Variations of the LRT

- The LRT that minimizes the Bayes Risk is called the Bayes Criterion

\[
\Lambda(x) = \begin{cases} 
\frac{\omega_1}{\omega_2} & \frac{P(x \mid \omega_1)}{P(x \mid \omega_2)} > \frac{(C_{12} - C_{22}) P[\omega_2]}{(C_{21} - C_{11}) P[\omega_1]} \\
\frac{\omega_2}{\omega_1} & \frac{P(x \mid \omega_2)}{P(x \mid \omega_1)} < \frac{(C_{21} - C_{11}) P[\omega_1]}{(C_{12} - C_{22}) P[\omega_2]} 
\end{cases}
\]

- Maximum A Posteriori Criterion
  - Sometimes we will be interested in minimizing \( P[\text{error}] \), which is a special case of the Bayes Criterion if we use a zero-one cost function

\[
C_{ij} = \begin{cases} 
0 & i = j \\
1 & i \neq j 
\end{cases} \Rightarrow \Lambda(x) = \begin{cases} 
\frac{\omega_1}{\omega_2} & \frac{P(x \mid \omega_1)}{P(x \mid \omega_2)} > \frac{P(\omega_2)}{P(\omega_1)} \\
\frac{\omega_2}{\omega_1} & \frac{P(x \mid \omega_2)}{P(x \mid \omega_1)} < \frac{P(\omega_1)}{P(\omega_2)} 
\end{cases} \iff \frac{P(\omega_1 \mid x)}{P(\omega_2 \mid x)} > \frac{P(\omega_2)}{P(\omega_1)}
Variations of the LRT

- **Maximum Likelihood**
  - Finally, the simplest form of the LRT is obtained for the case of equal priors $P(\omega_i) = 1/2$ and zero-one cost function:
    
    $C_{ij} = \begin{cases} 
    0 & i = j \\
    1 & i \neq j
    \end{cases}$
    
    $P(\omega_i) = \frac{1}{C} \quad \forall i$

    $\Rightarrow \Lambda(x) = \frac{P(x | \omega_1)}{P(x | \omega_2)} > 1$
    
    $\omega_1$
    
    $\omega_2$

  - When would you want to use an ML criterion?
Multi-class problems

- The previous decision rules were derived for two-class problems, but generalize gracefully for multiple classes:

  - To minimize $P[\text{error}]$ choose the class $\omega_i$ with highest $P(\omega_i | x)$
    \[ \omega_i = \arg\max_{1 \leq i \leq C} P(\omega_i | x) \]

  - To minimize Bayes risk choose the class $\omega_i$ with lowest $R(\omega_i | x)$
    \[ \omega_i = \arg\min_{1 \leq i \leq C} R(\omega_i | x) = \arg\min_{1 \leq i \leq C} \sum_{j=1}^{C} C_{ij} P(\omega_j | x) \]
**Discriminant functions (1)**

- All these decision rules have the same structure
  - At each point $x$ in feature space choose class $\omega_i$ which maximizes (or minimizes) some measure $g_i(x)$
  - This structure can be formalized with a set of discriminant functions $g_i(x)$, $i=1..C$, and the following decision rule

  \[
  \text{"assign } x \text{ to class } \omega_i \text{ if } g_i(x) > g_j(x) \quad \forall j \neq i"
  \]

- We can then express the three basic decision rules (Bayes, MAP and ML) in terms of Discriminant Functions:

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Discriminant Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayes</td>
<td>$g_i(x) = -\mathcal{R}(\alpha_i</td>
</tr>
<tr>
<td>MAP</td>
<td>$g_i(x) = P(\omega_i</td>
</tr>
<tr>
<td>ML</td>
<td>$g_i(x) = P(x</td>
</tr>
</tbody>
</table>
Discriminant functions (2)

Therefore, we can visualize the decision rule as a network that computes C discriminant functions and selects the class corresponding to the largest discriminant.
Recapping…

- The LRT is a theoretical result that can only be applied if we have complete knowledge of the likelihoods $P[x|\omega_i]$
  - $P[x|\omega_i]$ generally unknown, but can be estimated from data

- If the form of the likelihood is known (e.g., Gaussian) the problem is simplified b/c we only need to estimate the parameters of the model (e.g., mean and covariance)
  - This leads to a classifier known as QUADRATIC, which we cover next

- If the form of the likelihood is unknown, the problem becomes much harder, and requires a technique known as non-parametric density estimation
  - This technique is covered in lecture 3
Part 2: Quadratic classifiers
For Normally distributed classes, the discriminant functions reduce to very simple expressions

- The (multivariate) Gaussian density can be defined as

\[
p(x) = \frac{1}{(2\pi)^{n/2} \det(\Sigma)^{1/2}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)
\]

- Using the Bayes rule, the MAP discriminant function can be written as

\[
g_i(x) = P(\omega_i | x) = \frac{P(x | \omega_i)P(\omega_i)}{P(x)} = \frac{1}{(2\pi)^{n/2} \det(\Sigma_i)^{1/2}} \exp \left( -\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \right) P(\omega_i) \frac{1}{P(x)}
\]
Bayes classifier for Gaussian classes (2)

- Eliminating constant terms
  \[ g_i(x) = |\Sigma_i|^{-1/2} \exp \left[ -\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \right] P(\omega_i) \]

- Taking logs
  \[ g_i(x) = -\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) - \frac{1}{2} \log(|\Sigma_i|) + \log(P(\omega_i)) \]

- This is known as a QUADRATIC discriminant function (because it is a function of the square of x)

- In the next few slides we will analyze what happens to this expression under different assumptions for the covariance
Case 1: $\Sigma_i = \sigma^2 I$ (1)

- This situation occurs when the features are statistically independent, and have the same variance for all classes
  - In this case, the quadratic discriminant function becomes
    \[
    g_i(x) = -\frac{1}{2} (x - \mu_i)^T (\sigma^2 I)^{-1} (x - \mu_i) - \frac{1}{2} \log(\|\sigma^2 I\|) + \log(P(\omega_i)) = -\frac{1}{2\sigma^2} (x - \mu_i)^T (x - \mu_i) + \log(P(\omega_i))
    \]
  - Assuming equal priors and dropping constant terms
    \[
    g_i(x) = -(x - \mu_i)^T (x - \mu_i) = -\sum_{i=1}^{\text{DIM}} (x_i - \mu_i)^2
    \]
  - This is called an Euclidean-distance or nearest mean classifier

From [Schalkoff, 1992]
Case 1: $\sum_i = \sigma^2 I$ (1)

- This is probably the simplest statistical classifier that you can build:
  - “Assign an unknown example to the class whose center is the closest using the Euclidean distance”
Case 1: $\Sigma_i = \sigma^2 I$, example

$$\mu_1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}^T \quad \mu_2 = \begin{bmatrix} 7 \\ 4 \end{bmatrix}^T \quad \mu_3 = \begin{bmatrix} 2 \\ 5 \end{bmatrix}^T$$

$$\Sigma_1 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \Sigma_3 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$
Case 2: $\Sigma_i=\Sigma$ ($\Sigma$ non-diagonal)

- All the classes have the same covariance matrix, but the matrix is not diagonal
  - In this case, the quadratic discriminant becomes
    \[
    g_i(x) = \frac{-1}{2} (x - \mu_i)^T \Sigma^{-1} (x - \mu_i) - \frac{1}{2} \log(|\Sigma|) + \log(P(\omega_i))
    \]
  - Assuming equal priors and eliminating constant terms
    \[
    g_i(x) = -(x - \mu_i)^T \Sigma^{-1} (x - \mu_i)
    \]
  - This is known as a Mahalanobis-distance classifier
The Mahalanobis distance

- The quadratic term is called the **Mahalanobis distance**, a very important metric in SPR
  - The Mahalanobis metric is a vector distance that uses a $\Sigma^{-1}$ norm
  - $\Sigma^{-1}$ can be thought of as a “stretching factor” on the space
  - Note that for an identity covariance matrix ($\Sigma=I$), the Mahalanobis distance becomes the familiar Euclidean distance

\[
\|x_i - \mu\|_{\Sigma^{-1}}^2 = K
\]

\[
\|x_i - \mu\|^2 = K
\]
Case 2: $\Sigma_i = \Sigma$ ($\Sigma$ non-diagonal), example

$\mu_1 = \begin{bmatrix} 3 \\ 1 \\ 0.7 \end{bmatrix}, \; \Sigma_1 = \begin{bmatrix} 1 & 2 \\ 0.7 & 2 \end{bmatrix}$

$\mu_2 = \begin{bmatrix} 5 \\ 1 \\ 0.7 \end{bmatrix}, \; \Sigma_2 = \begin{bmatrix} 1 & 4 \\ 0.7 & 2 \end{bmatrix}$

$\mu_3 = \begin{bmatrix} 2 \\ 1 \\ 0.7 \end{bmatrix}, \; \Sigma_3 = \begin{bmatrix} 2 & 5 \\ 0.7 & 2 \end{bmatrix}$
Case 3: $\Sigma_i \neq \Sigma_j$ general case, example

$\mu_1 = [3 \ 2]^\top$  \hspace{1cm} $\mu_2 = [5 \ 4]^\top$  \hspace{1cm} $\mu_3 = [2 \ 5]^\top$

$\Sigma_1 = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$  \hspace{1cm} $\Sigma_2 = \begin{bmatrix} 1 & -1 \\ -1 & 7 \end{bmatrix}$  \hspace{1cm} $\Sigma_3 = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 3 \end{bmatrix}$
Numerical example (1)

- Derive a linear discriminant function for the two-class 3D classification problem defined by

\[
\mu_1 = [0 \ 0 \ 0]^T; \ \mu_2 = [1 \ 1 \ 1]^T; \ \Sigma_1 = \Sigma_2 = \begin{bmatrix} 1/4 & 0 & 0 \\ 0 & 1/4 & 0 \\ 0 & 0 & 1/4 \end{bmatrix}; \ p(\omega_2) = 2p(\omega_1)
\]

- Anybody would dare to sketch the likelihood densities and decision boundary for this problem?
Numerical example (2)

**Solution**

\[
g_i(x) = -\frac{1}{2} (x - \mu_i)^T \Sigma^{-1} (x - \mu_i) - \frac{1}{2} \log |\Sigma| + \log P(\omega_i) \propto -\frac{1}{2} \begin{bmatrix} x - \mu_x \\ y - \mu_y \\ z - \mu_z \end{bmatrix}^T \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x - \mu_x \\ y - \mu_y \\ z - \mu_z \end{bmatrix} + \log P(\omega_i)
\]

\[
g_1(x) = -\frac{1}{2} \begin{bmatrix} x - 0 \\ y - 0 \\ z - 0 \end{bmatrix}^T \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x - 0 \\ y - 0 \\ z - 0 \end{bmatrix} + \log \frac{1}{3};
\]

\[
g_2(x) = -\frac{1}{2} \begin{bmatrix} x - 1 \\ y - 1 \\ z - 1 \end{bmatrix}^T \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x - 1 \\ y - 1 \\ z - 1 \end{bmatrix} + \log \frac{2}{3}
\]
Numerical example (3)

Solution (continued)

\[ g_1(x) \geq g_2(x) \]

\[ -2(x^2 + y^2 + z^2) + \log_3 1 > -2(x-1)^2 + (y-1)^2 + (z-1)^2 + \log_3 2 \]

\[ x+y+z \geq \frac{6 - \log_2 4}{4} = 1.32 \]

- Classify the test example \( x_u = [0.1 \ 0.7 \ 0.8]^T \)

\[ 0.1 + 0.7 + 0.8 = 1.6 > 1.32 \Rightarrow x_u \in \omega_2 \]
Conclusions

- **The Euclidean distance classifier is Bayes-optimal** if
  - Gaussian classes and equal covariance matrices proportional to the identity matrix and equal priors

- **The Mahalanobis dist. classifier is Bayes-optimal** if
  - Gaussian classes and equal covariance matrices and equal priors

*Bayes optimal means that the classifier yields the minimum \( P[\text{error}] \), which is the best ANY classifier can achieve*
Part 3: Linear classifiers
Linear Discriminant Functions (1)

- The objective of this section is to present methods for learning linear discriminant functions of the form

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

\[ \begin{cases} g(x) > 0 & x \in \omega_1 \\ g(x) < 0 & x \in \omega_2 \end{cases} \]

- where \( w \) is the weight vector and \( w_0 \) is the threshold or bias
- Similar discriminant functions were derived in the previous section as a special case of the quadratic classifier

- In this chapter, the discriminant functions will be derived in a non-parametric fashion, this is, no assumptions will be made about the underlying densities
Linear Discriminant Functions (2)

- For convenience, we will focus on binary classification
  - Extension to the multicategory case can be easily achieved by
    - Using $\omega_i/not \omega_i$ dichotomies
    - Using $\omega_i/\omega_i$ dichotomies
Gradient descent (1)

- Gradient descent is a general method for function minimization
  - From basic calculus, we know that the minimum of a function $J(x)$ is defined by the zeros of the gradient
    \[ x^* = \arg\min x [J(x)] \Rightarrow \nabla_x J(x) = 0 \]
  - Only in very special cases this minimization function has a closed form solution
  - In some other cases, a closed form solution may exist, but is numerically ill-posed or impractical (e.g., memory requirements)
Gradient descent (2)

- Gradient descent finds the minimum in an iterative fashion by moving in the direction of steepest descent

1. Start with an arbitrary solution $x(0)$
2. Compute the gradient $\nabla_x J(x(k))$
3. Move in the direction of steepest descent:
   $$x(k + 1) = x(k) - \eta \nabla_x J(x(k))$$
4. Go to 1 (until convergence)

- where $\eta$ is a learning rate
Perceptron learning (1)

Let’s now consider the problem of solving a binary classification problem with a linear discriminant.

- As usual, assume we have a dataset \( X = \{x^{(1)}, x^{(2)}, \ldots x^{(N)}\} \) containing examples from the two classes.
- For convenience, we will absorb the intercept \( w_0 \) by augmenting the feature vector \( x \) with an additional constant dimension:

\[
\begin{align*}
    w^T x + w_0 &= \begin{bmatrix} w_0 & w^T \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} = a^T y
\end{align*}
\]

From [Duda, Hart and Stork, 2001]
Perceptron learning (2)

- Keep in mind that our objective is to find a vector $a$ such that

$$g(x) = a^T y \begin{cases} > 0 & x \in \omega_1 \\ < 0 & x \in \omega_2 \end{cases}$$

- To simplify the derivation, we will “normalize” the training set by replacing all examples from class $\omega_2$ by their negative

$$y \leftarrow [-y] \ \forall y \in \omega_2$$

- This allows us to ignore class labels and look for a weight vector such that

$$a^T y > 0 \ \forall y$$

From [Duda, Hart and Stork, 2001]
Perceptron learning (3)

To find this solution we must first define an objective function $J(a)$

- A good choice is what is known as the **Perceptron** criterion

$$J_P(a) = \sum_{y \in Y_M} (-a^T y)$$

- where $Y_M$ is the set of examples *misclassified* by $a$
- Note that $J_P(a)$ is non-negative since $a^T y < 0$ for misclassified samples
Perceptron learning (4)

To find the minimum of $J_P(a)$, we use gradient descent

- The gradient is defined by

$$\nabla_a J_P(a) = \sum_{y \in Y_M} (-y)$$

- And the gradient descent update rule becomes

$$a(k + 1) = a(k) + \eta \sum_{y \in Y_M(k)} y$$

- This is known as the **perceptron batch update rule**.

  - The weight vector may also be updated in an “on-line” fashion, this is, after the presentation of each individual example

    $$a(k + 1) = a(k) + \eta y^{(i)}$$ **Perceptron rule**

- Where $y^{(i)}$ is an example that has been misclassified by $a(k)$
Perceptron learning (5)

- If classes are linearly separable, the perceptron rule is guaranteed to converge to a valid solution.
- However, if the two classes are not linearly separable, the perceptron rule will not converge.
  - Since no weight vector $a$ can correctly classify every sample in a non-separable dataset, the corrections in the perceptron rule will never cease.
  - One ad-hoc solution to this problem is to enforce convergence by using variable learning rates $\eta(k)$ that approach zero as $k$ approaches infinite.
Perceptron learning example

Consider the following classification problem
- class $\omega_1$ defined by feature vectors $x=\{[0, 0]^T, [0, 1]^T\}$;
- class $\omega_2$ defined by feature vectors $x=\{[1, 0]^T, [1, 1]^T\}$.

Apply the perceptron algorithm to build a vector ‘a’ that separates both classes.
- Use learning rate $\eta=1$ and $a(0)=[1, -1, -1]^T$.
- Update the vector ‘a’ on a per-example basis
- Present examples in the order in which they were given above.
- Draw a scatterplot of the data, and the separating line you found with the perceptron rule.
The classical Minimum Squared Error (MSE) criterion provides an alternative to the perceptron rule

- The perceptron rule seeks a weight vector $a^T$ that satisfies the inequality $a^T y^{(i)} > 0$
  - The perceptron rule only considers misclassified samples, since these are the only ones that violate the above inequality
- Instead, the MSE criterion looks for a solution to the equality $a^T y^{(i)} = b^{(i)}$, where $b^{(i)}$ are some pre-specified target values (e.g., class labels)
  - As a result, the MSE solution uses ALL of the samples in the training set

From [Duda, Hart and Stork, 2001]
**Minimum Squared Error solution (2)**

- The system of equations solved by MSE is

\[
\begin{bmatrix}
y_0^{(1)} & y_1^{(1)} & \cdots & y_D^{(1)} \\
y_0^{(2)} & y_1^{(2)} & \cdots & y_D^{(2)} \\
\vdots & \vdots & \ddots & \vdots \\
y_0^{(N)} & y_1^{(N)} & \cdots & y_D^{(N)}
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
\vdots \\
a_D
\end{bmatrix}
= 
\begin{bmatrix}
b_0^{(1)} \\
b_1^{(2)} \\
\vdots \\
b_D^{(N)}
\end{bmatrix}
\Leftrightarrow Ya = b
\]

- where \( a \) is the weight vector, each row in \( Y \) is a training example, and each row in \( b \) is the corresponding class label.

- For consistency, we will continue assuming that examples from class \( \omega_2 \) have been replaced by their negative vector, although this is not a requirement for the MSE solution.

From [Duda, Hart and Stork, 2001]
An exact solution to $Y a = b$ can sometimes be found

- If the number of (independent) equations ($N$) is equal to the number of unknowns ($D+1$), the exact solution is defined by

$$a = Y^{-1} b$$

In practice, however, $Y$ will be singular so its inverse $Y^{-1}$ does not exist

- $Y$ will commonly have more rows (examples) than columns (unknown), which yields an over-determined system, for which an exact solution cannot be found
Minimum Squared Error solution (4)

The solution in this case is to find a weight vector that minimizes some function of the error between the model (aY) and the desired output (b)

- In particular, MSE seeks to minimize the sum of the squares of these errors:

\[
J_{MSE}(a) = \sum_{i=1}^{N} (a^T y^{(i)} - b^{(i)})^2 = \| aY - b \|^2
\]

- which, as usual, can be found by setting its gradient to zero
The pseudo-inverse solution (1)

- The gradient of the objective function is
  \[ \nabla_a J_{\text{MSE}}(a) = \sum_{i=1}^{N} 2(a^T y^{(i)} - b^{(i)}) y^{(i)} = 2Y^T (Ya - b) = 0 \]

  - with zeros defined by
  \[ Y^T Ya = Y^T b \]

  - Notice that \( Y^T Y \) is now a square matrix!

- If \( Y^T Y \) is nonsingular, the MSE solution becomes
  \[ a = \left( Y^T Y \right)^{-1} Y^T b = Y^\dagger b \]

  - where the matrix \( Y^\dagger = (Y^T Y)^{-1} Y^T \) is known as the pseudo-inverse of \( Y \) (\( Y^\dagger Y = I \))

  - Note that, in general, \( YY^\dagger \neq I \) in general
Least-mean-squares solution (1)

- The objective function $J_{MSE}(a) = ||Ya-b||^2$ can also be found using a gradient descent procedure
  - This avoids the problems that arise when $Y^TY$ is singular
  - In addition, it also avoids the need for working with large matrices
- Looking at the expression of the gradient, the obvious update rule is
  \[ a(k+1) = a(k) + \eta(k) Y^T (b - Ya(k)) \]
  - It can be shown that if $\eta(k) = \eta(1)/k$, where $\eta(1)$ is any positive constant, this rule generates a sequence of vectors that converge to a solution to $Y^T(Ya-b)=0$
Least-mean-squares solution (2)

- The storage requirements of this algorithm can be reduced by considering each sample sequentially.

\[
a(k + 1) = a(k) + \eta(k)(b^T - y^T a(k)) y^T
\]

This is known as the Widrow-Hoff, least-mean-squares (LMS) or delta rule [Mitchell, 1997]

\[
a(k + 1) = a(k) + \eta(k) Y^T (b - Ya(k))
\]

From [Duda, Hart and Stork, 2001]