CS 559 Machine Learning

Linear Classification

Yue Ning
Department of Computer Science
Stevens Institute of Technology

Outline

Learning Objectives

1. Understand and able to implement Linear Discriminant Analysis;
2. Understand Perceptron and its limitations;

Classification

Classification task: finding a function f that classifies examples into a given set of categories $\{C_1, C_2, .., C_k\}$

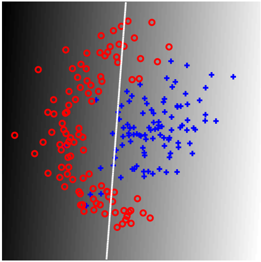
$$\mathbf{x} \quad \rightarrow \quad \boxed{\text{function } f} \quad \rightarrow \quad y \in \{C_1, C_2, .., C_k\}$$

A classification example:

 $\rightarrow$ $\boxed{\text{function } f}$ $\rightarrow$ “Cat”

Binary Classification

Task: Assign each data point to one of two classes.

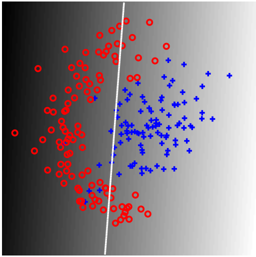


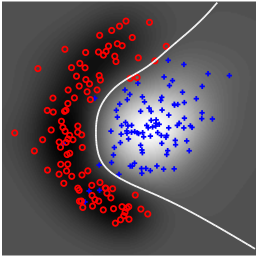
Examples:

- Is there a face in this image?
- Will this neuron spike in response to this stimulus?
- Based on this brain-scan, does this patient have a given disease or not?
- Will this customer buy this product or not?
- Is this person likely to be a democrat/republican?

Notation: we have data $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, with $y_n = 1$ if x_n belongs to class 1 and $y_n = -1$ if x_n belongs to class -1.

Linear Discriminant Functions





Of course, linear algorithms can be used together with nonlinear feature spaces or nonlinear basis functions in order to solve nonlinear classification problems!

Linear discriminants separate the space by a hyperplane, and the parameters define its normal vector.

- Decision function: $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \omega_o$
- Classification:

$$\text{if } f(\mathbf{x}) > 0 \text{ say } \mathbf{x} \text{ belongs to class 1}$$
$$\text{if } f(\mathbf{x}) < 0 \text{ say } \mathbf{x} \text{ belongs to class -1}$$

- The decision-surface has equation $f(\mathbf{x}) = 0$, and is a hyperplane of dimensionality $D - 1$.
- $\mathbf{w}$ is the normal vector to the hyperplane which separates points into the positive class or negative class.
- ω_o determines the location of the decision-surface
- $|f(\mathbf{x})|$ is proproptional to the perpendicular distance to the decision-surface (with factor 1 if $||\mathbf{w}|| = 1$).

Linear Discriminant Functions-Geometrical Properties

- Decision boundary:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \omega_o = 0$$

- Let $\mathbf{x}_1, \mathbf{x}_2$ be two points which lie on the decision boundary

$$f(\mathbf{x}_1) = \mathbf{w}^\top \mathbf{x}_1 + \omega_o = 0, f(\mathbf{x}_2) = \mathbf{w}^\top \mathbf{x}_2 + \omega_o = 0$$
$$\Rightarrow \mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0$$

- $\mathbf{w}$ represents the orthogonal direction to the decision boundary.

Linear Discriminant Functions-Geometrical Properties Cont.

- $\mathbf{w}^{*T} = \frac{\mathbf{w}^T}{||\mathbf{w}||}$
- $\mathbf{w}^{*T}(\mathbf{x} - \mathbf{x}_0)$ is the projection of $(\mathbf{x} - \mathbf{x}_0)$ onto the $\mathbf{w}^*$ direction; $\mathbf{x}_0$ is a point on the decision boundary;
- Thus,

$$\frac{\mathbf{w}^T}{||\mathbf{w}||}(\mathbf{x} - \mathbf{x}_0) = \frac{1}{||\mathbf{w}||}(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mathbf{x}_0)$$

$$= \frac{1}{||\mathbf{w}||}(\mathbf{w}^T\mathbf{x} + \omega_0) = \frac{f(\mathbf{x})}{||\mathbf{w}||}$$

when $\mathbf{x} = \mathbf{0}$, $\frac{f(\mathbf{x})}{||\mathbf{w}||} = \frac{\omega_0}{||\mathbf{w}||}$

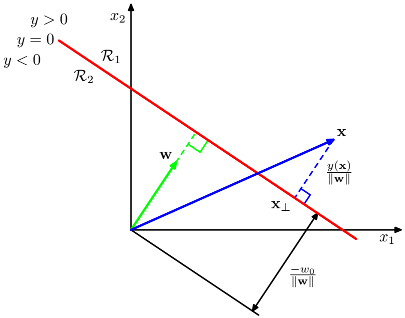
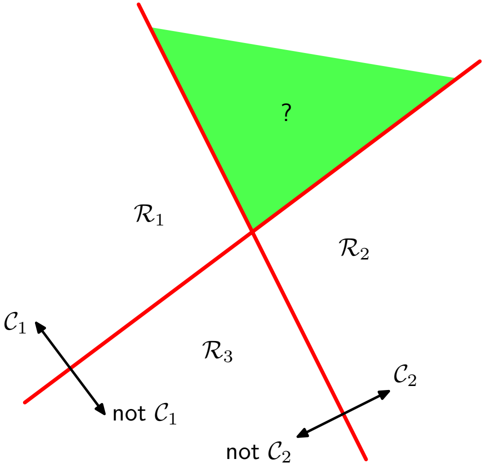


Figure: Signed orthogonal distance of the origin from the decision

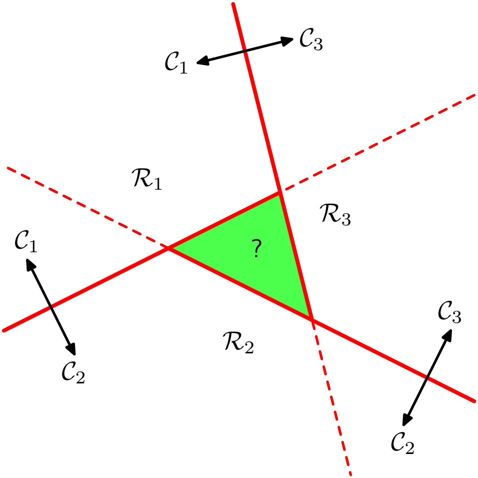
Linear Discriminant Functions: Multiple classes

<u>one-versus-the-rest</u>: K-1 classifiers each of which solves a two-class problem of separating points of C_k from points not in that class.



Linear Discriminant Functions: Multiple classes

one-versus-one: $\frac{K(K-1)}{2}$ binary discriminant functions, one for every possible pair of classes.

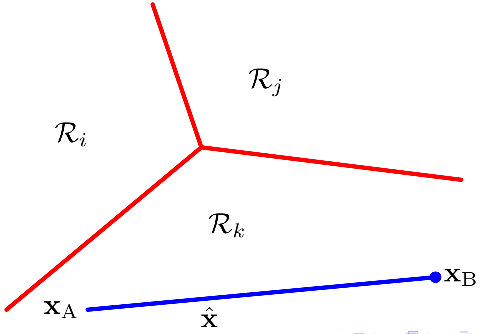


Linear Discriminant Functions: Multiple classes

- Solution: consider a single K-class discriminant comprising K linear functions of the form

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

- Assign a point $\mathbf{x}$ to class C_k if $f_k(\mathbf{x}) > f_j(\mathbf{x}) \forall j \neq k$
- The decision boundary between class C_k and class C_j is given by: $f_k(\mathbf{x}) = f_j(\mathbf{x}) \Rightarrow (\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$

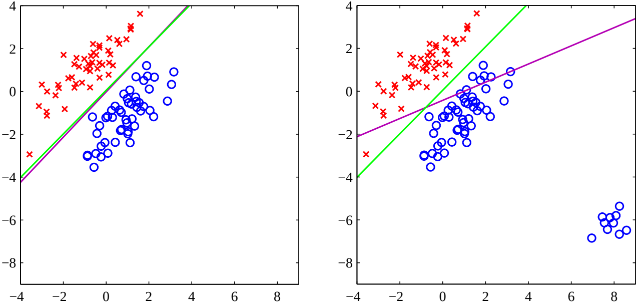


Multiple algorithms and methods

- Mis-classification rate $C(\mathbf{w}) = \frac{1}{N}\sum_n \delta\left[f(\mathbf{x}_n) = y_n\right]$ (i.e. average number of errors) difficult to optimize over $\mathbf{w}$, and might have multiple solutions.
- Many algorithms can be derived by replacing C with another cost function that can be optimized.
- Linear classification algorithms:
  1. Least-square classification
  2. Fisher's linear Discriminant
  3. Logistic regression
  4. Support Vector Machines
  5. Rosenblatts' perceptron

Least square classification

- We have to fit the function $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \omega_o$ to data.
- Simply do a linear regression from $\mathbf{x}$ to y by minimizing the sum-of-squared errors $\sum_n (f(\mathbf{x}_n) - y_n)^2$.
- $\mathbf{w}_{reg} = \left(\sum_n \mathbf{x}_n \mathbf{x}_n^\top\right)^{-1} \sum_n \mathbf{x}_n y_n$
- Q: In what situations might this be a bad idea?



Bishop PRML Figure 4.4

Least square classification

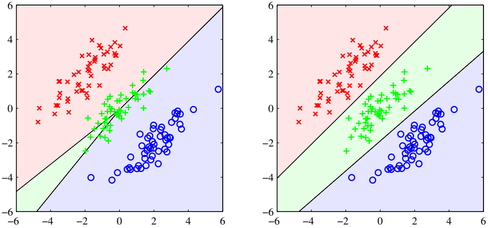


Figure: Left: using a least-squares discriminant; Right: using logistic regression

Bishop PRML Figure 4.5

Classification via projection

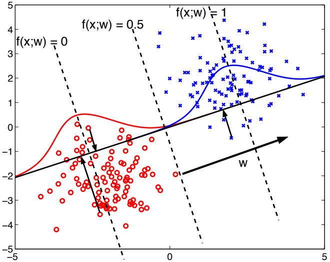
- A linear function: $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \omega_o$ assuming in 2D, projects each point $\mathbf{x} = [x_1, x_2]^T$ to a line parallel to $\mathbf{w}$:

$$
\begin{array}{c|c}
\text{point in } \mathcal{R}^d & \text{projected point in } \mathcal{R} \\
\mathbf{x}_1 & z_1 = \mathbf{w}^T \mathbf{x}_1 \\
\mathbf{x}_2 & z_2 = \mathbf{w}^T \mathbf{x}_2 \\
\ldots & \ldots \\
\mathbf{x}_n & z_n = \mathbf{w}^T \mathbf{x}_n
\end{array}
$$

- We can study how well the projected points $z_1, \ldots, z_n$ are separated across the classes when they are viewed as functions of $\mathbf{w}$.

Classification via projection

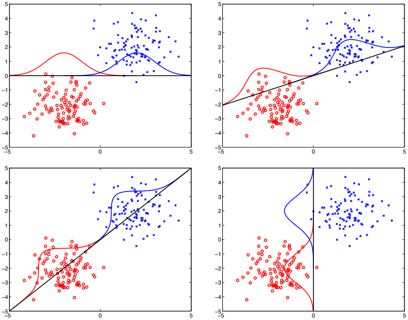
- A linear function: $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \omega_o$ assuming in 2D, projects each point $\mathbf{x} = [x_1, x_2]^T$ to a line parallel to $\mathbf{w}$:



- We can study how well the projected points $z_1, \ldots, z_n$ viewed as functions of $\mathbf{w}$ are separated across the classes.

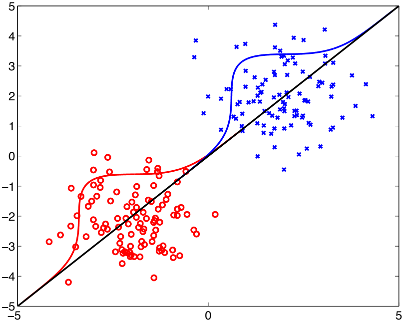
Classification via projection

- By varying **w** we get different levels of separation between the projected points



Optimizing the projection

- We would like to find **w** that somehow maximizes the separation of the projected points across classes.



- We can quantify the separation (overlap) in terms of means and variances of the resulting 1-dimensional class distributions

Fisher's linear discriminant

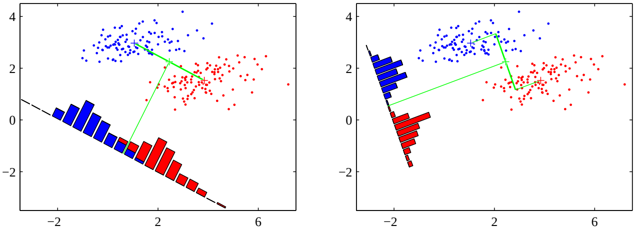
- One way to view a linear classification model is in terms of dimensionality reduction.
- Two class case: suppose we project $\mathbf{x}$ onto one dimension:

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

- Set a threshold t:

$$\begin{array}{ll} \text{if } f \leq t & \text{assign } C_1 \text{ to } \mathbf{x} \\ \text{otherwise} & \text{assign } C_2 \text{ to } \mathbf{x} \end{array}$$

Fisher's linear discriminant



- Find an orientation along which the projected samples are well separated;
- This is exactly the goal of linear discriminant analysis (LDA);
- In other words: we are after the linear projection that best separates the data, i.e. best discriminates data of different classes.

Fisher's linear discriminant

- Two classes: $\{C_+, C_-\}$
- N_+ samples of class C_+
- N_- samples of class C_-
- Consider $\mathbf{w} \in \mathbb{R}^d$ with $||\mathbf{w}|| = 1$
- Then: $\mathbf{w}^T\mathbf{x}$ is the projection of $\mathbf{x}$ along the direction of $\mathbf{w}$.
- We want the projections $\mathbf{w}^T\mathbf{x}$ where $\mathbf{x} \in C_+$ separated from the projections $\mathbf{w}^T\mathbf{x}$ where $\mathbf{x} \in C_-$

Fisher's linear discriminant

- A measure of the separation between the projected points is the difference of the sample means:
  - Sample mean of class C_+:

$$\mathbf{m}_+ = \frac{1}{N_+} \sum_{\mathbf{x} \in C_+} \mathbf{x}$$

  - Sample mean for the projected points:

$$m_+ = \frac{1}{N_+} \sum_{\mathbf{x} \in C_+} \mathbf{w}^T \mathbf{x} = \mathbf{w}^T \mathbf{m}_+$$

$$\Rightarrow |m_+ - m_-| = \mathbf{w}^T (\mathbf{m}_+ - \mathbf{m}_-)$$

- We wish to make the above difference as large as we can. In addition, ...

Fisher's linear discriminant

- To obtain good separation of the projected data, we really want the difference between the means to be large relative to some measure of the standard deviation of each class:
  - Scatter of the projected samples of class C_+:

$$s_+^2 = \sum_{\mathbf{x} \in C_+} (\mathbf{w}^T \mathbf{x} - m_+)^2$$

  - Total within-class scatter of the projected samples:

$$s_+^2 + s_-^2$$

  - Fisher linear discriminant analysis:

$$\arg\max_{\mathbf{w}} \frac{|m_+ - m_-|^2}{s_+^2 + s_-^2}$$

Fisher's linear discriminant

- $J(\mathbf{w}) = \frac{|m_+ - m_-|^2}{s_+^2 + s_-^2}$
- To obtain $J(\mathbf{w})$ as an explicit function of $\mathbf{w}$, we define the following matrices:

$$S_+ = \sum_{\mathbf{x} \in C_+} (\mathbf{x} - \mathbf{m}_+)(\mathbf{x} - \mathbf{m}_+)^T$$

Within-class scatter matrix:

$$S_w = S_+ + S_-$$

- Then:

$$\begin{aligned} s_+^2 &= \sum_{\mathbf{x} \in C_+} (\mathbf{w}^T \mathbf{x} - m_+)^2 = \sum_{\mathbf{x} \in C_+} (\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \mathbf{m}_+)^2 \\ &= \sum_{\mathbf{x} \in C_+} \mathbf{w}^T (\mathbf{x} - \mathbf{m}_+)(\mathbf{x} - \mathbf{m}_+)^T \mathbf{w} = \mathbf{w}^T S_+ \mathbf{w} \end{aligned}$$

Fisher's linear discriminant

- So, $s_+^2 = \mathbf{w}^T S_+ \mathbf{w}$ and $s_-^2 = \mathbf{w}^T S_- \mathbf{w}$
- Thus,

$$
\begin{aligned}
s_+^2 + s_-^2 &= \mathbf{w}^T S_+ \mathbf{w} + \mathbf{w}^T S_- \mathbf{w} \\
&= \mathbf{w}^T (S_+ + S_-) \mathbf{w} \\
&= \mathbf{w}^T S_w \mathbf{w}
\end{aligned}
$$

- Similarly:

$$
\begin{aligned}
(m_+ - m_-)^2 &= (\mathbf{w}^T \mathbf{m}_+ - \mathbf{w}^T \mathbf{m}_-)^2 \\
&= \mathbf{w}^T (\mathbf{m}_+ - \mathbf{m}_-)(\mathbf{m}_+ - \mathbf{m}_-)^T \mathbf{w} \\
&= \mathbf{w}^T S_B \mathbf{w}
\end{aligned}
$$

where $S_B = (\mathbf{m}_+ - \mathbf{m}_-)(\mathbf{m}_+ - \mathbf{m}_-)^T$ (Between-class scatter matrix)

Fisher's linear discriminant

- We have obtained:

$$J(\mathbf{w}) = \frac{|m_+ - m_-|^2}{s_+^2 + s_-^2} = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

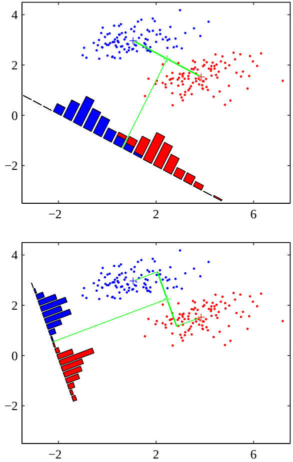
- $J(\mathbf{w})$ is maximized when $(\mathbf{w}^T S_B \mathbf{w}) S_W \mathbf{w} = (\mathbf{w}^T S_W \mathbf{w}) S_B \mathbf{w}$
- We observe that:

$$S_B \mathbf{w} = (\mathbf{m}_+ - \mathbf{m}_-)(\mathbf{m}_+ - \mathbf{m}_-)^T \mathbf{w}$$

  where $(\mathbf{m}_+ - \mathbf{m}_-)^T \mathbf{w}$ is a scalar, always in the direction of $(\mathbf{m}_+ - \mathbf{m}_-)$
- Solution:

$$\mathbf{w} = S_W^{-1}(\mathbf{m}_+ - \mathbf{m}_-)$$

Fisher's linear discriminant: Summary



Bishop PRML Figure 4.6

- $\mathbf{m}_+ = \frac{1}{N_+} \sum_{n \in C_+} \mathbf{x}_n$
  $\mathbf{m}_- = \frac{1}{N_-} \sum_{n \in C_-} \mathbf{x}_n$
- Maximize projection-distance of class means $\mathbf{w}_{simple} \propto \mathbf{m}_+ - \mathbf{m}_-$
- Maximizing distance between means ignores that the projected variances might also be big.
- Fix: Maximize the ratio of between-class variance to within-class variance ('signal to noise'). Fisher criterion

$$J_{\mathbf{w}} = \frac{(m_+ - m_-)^2}{s_+^2 + s_-^2} \tag{1}$$

$$\mathbf{w}_{lda} = S_W^{-1}(\mathbf{m}_+ - \mathbf{m}_-)$$

Fisher's linear discriminant

- Gives the linear function with the maximum ratio of between-class scatter to within-class scatter.
- The problem, e.g. classification, has been reduced from a d-dimensional problem to a more manageable one-dimensional problem.
- Optimal for multivariate normal class conditional densities.

Fisher's linear discriminant: Multi-Class

- The analysis can be extended to multiple classes.
- $S_W = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} (\mathbf{x}_i - \mathbf{m}_k)(\mathbf{x}_i - \mathbf{m}_k)^T$
- $S_B = \sum_{k=1}^{K} m_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$ where $\mathbf{m}$ is the global mean; m_k is the number of examples in class k
- Solve: $S_B \mathbf{v} = \lambda S_W \mathbf{v}$ the generalized eigenvalue problem
- At most K-1 distinct solution eigenvalues
- The optimal projection matrix V to a subspace of dimension k is given by the eigenvectors corresponding to the largest k eigenvalues

Fisher's linear discriminant

- LDA is a linear technique for dimensionality reduction: it projects the data along directions that can be expressed as linear combination of the input features.
- The "appropriate" transformation depends on the data and on the task we want to perform on the data. Note that LDA uses class labels.
- Non-linear extensions of LDA exist (e.g., generalized LDA).

The Perceptron Algorithm (Frank Rosenblatt, 1957)

- First learning algorithm for neural networks.
- Originally introduced for character classification, where each character is represented as an image;
- Total input to output node:

$$\sum_j w_j x_j$$

- Output unit performs the function (activation function):

$$H(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Perceptron: Learning Algorithm

- Goal: compute a mapping from inputs to the outputs.
- Example: two class character recognition problem.
  - Training set: set of images representing either the character 'a' or the character 'b' (supervised learning);
  - Learning task: learn the weights so that when a new unlabelled image comes in, the network can predict its label.
  - Setting: d input units (intensity level of a pixel), 1 output unit.

Perceptron: Learning Algorithm

The algorithm proceeds as follows:

- Initial random setting of weights;
- The input is a random sequence $\{\mathbf{x}_k\}$
- For each element of class C_1, if output $= 1$ (correct), do nothing; otherwise, update weights;
- For each element of class C_2, if output $= 0$ (correct), do nothing; otherwise, update weights;

Perceptron: Learning Algorithm

- More formally: $\mathbf{x} = (x_1, x_2, .., x_d)^T, \mathbf{w} = (w_1, w_2, .., w_d)^T$
- θ: Threshold of the output unit
- Unit output: $\mathbf{w}^T\mathbf{x} = w_1x_1 + w_2x_2 + ... + x_dx_d$
- Output class 1 if $\mathbf{w}^T\mathbf{x} - \theta \geq 0$
- To eliminate the explicit dependence on θ: Output class 1 if: $\mathbf{w}^T\mathbf{x} \geq 0$

Perceptron: Learning Algorithm

- We want to learn values of the weights so that the perceptron correctly discriminate elements of C_1 from elements of C_2
- Given $\mathbf{x}$ in input, if $\mathbf{x}$ is classified correctly, weights are unchanged, otherwise:

$$\mathbf{w} = \begin{cases} \mathbf{w} + \mathbf{x} & \text{if an element of class } C_1 \text{ was classified as in } C_2 \\ \mathbf{w} - \mathbf{x} & \text{if an element of class } C_2 \text{ was classified as in } C_1 \end{cases}$$

Perceptron: Learning Algorithm

- 1^{st} case: $\mathbf{x} \in C_1$ and was classified in C_2. The correct answer is 1, which corresponds to: $\mathbf{w}^T\mathbf{x} \geq 0$, we have $\mathbf{w}^T\mathbf{x} < 0$. We want to get closer to the correct answer: $\mathbf{w}^T\mathbf{x} < \mathbf{w}'^T\mathbf{x}$.

$$\mathbf{w}^T\mathbf{x} < \mathbf{w}'^T\mathbf{x}, \text{ iff } \mathbf{w}^T\mathbf{x} < (\mathbf{w}+\mathbf{x})^T\mathbf{x}$$

$$(\mathbf{w}+\mathbf{x})^T\mathbf{x} = \mathbf{w}^T\mathbf{x} + \mathbf{x}^T\mathbf{x} = \mathbf{w}^T\mathbf{x} + ||\mathbf{x}||^2$$

because $||\mathbf{x}||^2 > 0$, the condition is verified.

- 2^{st} case: $\mathbf{x} \in C_2$ and was classified in C_1. The correct answer is 0, which corresponds to: $\mathbf{w}^T\mathbf{x} < 0$, we have $\mathbf{w}^T\mathbf{x} \geq 0$. We want to get closer to the correct answer: $\mathbf{w}^T\mathbf{x} > \mathbf{w}'^T\mathbf{x}$.

$$\mathbf{w}^T\mathbf{x} > \mathbf{w}'^T\mathbf{x}, \text{ iff } \mathbf{w}^T\mathbf{x} < (\mathbf{w}-\mathbf{x})^T\mathbf{x}$$

$$(\mathbf{w}-\mathbf{x})^T\mathbf{x} = \mathbf{w}^T\mathbf{x} - \mathbf{x}^T\mathbf{x} = \mathbf{w}^T\mathbf{x} - ||\mathbf{x}||^2$$

because $||\mathbf{x}||^2 > 0$, the condition is verified.

Perceptron: Learning Algorithm

In summary:

- A random sequence $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$ is generated such that $x_i \in C_1 \cup C_2$
- If $\mathbf{x}_k$ is correctly classified, then $\mathbf{w}_{k+1} = \mathbf{w}_k$ otherwise:

$$\mathbf{w}_{k+1} = \begin{cases} \mathbf{w}_k + \mathbf{x}_k & \text{if } \mathbf{x}_k \in C_1 \\ \mathbf{w}_k - \mathbf{x}_k & \text{if } \mathbf{x}_k \in C_2 \end{cases}$$

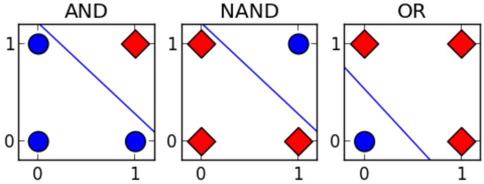
- Convergence theorem: regardless of the initial choice of weights, if the two classes are linearly separable, there exists $\mathbf{w}$ such that:

$$\begin{cases} \mathbf{w}^T\mathbf{x} \geq 0 & \text{if } \mathbf{x}_k \in C_1 \\ \mathbf{w}^T\mathbf{x} < 0 & \text{if } \mathbf{x}_k \in C_2 \end{cases}$$

  then the learning rule will find such solution after a finite number of steps.

Representational Power of Perceptrons

- Marvin Minsky and Seymour Papert, "Perceptrons" 1969:
  The perceptron can solve only problems with linearly separable classes
- Examples of linearly separable Boolean functions:



Representational Power of Perceptrons

- Examples of linearly separable Boolean functions:

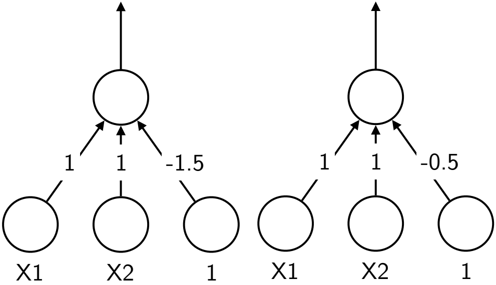
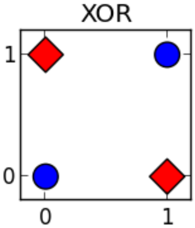


Figure: Left: AND; Right: OR

Representational Power of Perceptrons

- Examples of a non linearly separable Boolean function:



- The EX-OR function cannot be computed by a perceptron.

Naive Bayes: not (necessarily) a Bayesian method;

- A and B are independent iff $p(A, B) = p(A)p(B)$
- A and B are conditionally independent given C iff $p(A, B|C) = p(A|C)p(B|C)$

- Assume dimensions of **x** are conditionally independent given y.
- Example, bag of words:
  $p(\text{“Stevens”}, \text{“Institute”}, \text{“Technology”}|y) =$
  $p(\text{“Stevens”}|y)p(\text{“Institute”}|y)p(\text{“Technology”}|y)$
- Optimizing:

$$
\begin{aligned}
f(x) &= \arg\max_y p(y|x) \\
&= \arg\max_y p(x|y)p(y)/p(x) \\
&= \arg\max_y p(x|y)p(y) \\
&= \arg\max_y p(y)\prod_j p(x_j|y)
\end{aligned}
$$

Naive Bayes: Solution

- $p(y) :\leftarrow \frac{\#\text{ examples where } Y{=}y}{(\#\text{ examples})}$
- $p(X_j = x_j | y) \leftarrow \frac{\#\text{ ex where } Y{=}y \text{ and } X_j{=}x_j}{(\#\text{ ex where } Y{=}y)}$
- Learning by counting!

Gaussian naive Bayes: Continuous data

- $p(y) :\leftarrow \frac{\#\text{ examples where } Y=y}{(\#\text{ examples})}$
- $p(X_j = v|y) \leftarrow \frac{1}{\sqrt{2\pi\sigma_k^2}} exp\{-\frac{(v-\mu_k)^2}{2\sigma_k^2}\}$
- μ_k and σ_k are determined from the training data set.
- Learning by counting!

Gaussian naive Bayes: example (from Wikipedia)

Training data set:

Sex	height	weight	foot size
male	6	180	12
male	5.92	1990	11
male	5.58	170	12
male	5.92	165	10
female	5	100	6
female	5.5	150	8
female	5.42	140	7
female	5.75	150	9

Mean and variance

Sex	mean-height	var-height	mean-weight	var-weight	mean-footsize	var-footsize
male	5.855	$3.5 * 10^{-2}$	176.25	$1.2292 * 10^2$	11.25	$9.1667 * 10^{-1}$
female	5.4175	$9.7225 * 10^{-2}$	132.5	$5.5833 * 10^2$	7.5	1.6667

Gaussian naive Bayes: example (from Wikipedia)

Training mean and variance

Sex	mean-height	var-height	mean-weight	var-weight	mean-footsize	var-footsize
male	5.855	$3.5 * 10^{-2}$	176.25	$1.2292 * 10^{2}$	11.25	$9.1667 * 10^{-1}$
female	5.4175	$9.7225 * 10^{-2}$	132.5	$5.5833 * 10^{2}$	7.5	1.6667

Testing:

Sex	height	weight	foot size
?	6	130	8

$$p(m|x) \approx p(\mathrm{m})p(\text{height}|m)p(\text{weight}|m)p(\text{footsize}|m) = 6.1984 * 10^{-9}$$

$$p(f|x) \approx p(\mathrm{f})p(\text{height}|f)p(\text{weight}|f)p(\text{footsize}|f) = 5.3778 * 10^{-4}$$

What is Model Selection?

Given a set of models $\mathcal{M} = \{M_1, M_2, ..., M_R\}$, choose the model that is expected to do the best on the test data. $\mathcal{M}$ may consist of:

- Same learning model with different complexities or hyperparameters.
  - Nonlinear regression: polynomials with different degrees
  - K-Nearest Neighbors: Different choices of K
  - Decision Trees: Different choices of the number of levels/leaves
  - SVM: Different choices of the misclassification penalty
  - Regularized models: Different choices of the regularization parameter
  - Kernel based methods: Different choices of kernels ...and almost any learning problem
- Different learning models (e.g. SVM, kNN, DT, etc)

Note: usually considered in supervised learning but unsupervised learning faces this issue too.

Held-out Data

- Set aside a fraction (10-20%) of the training data.
- This part becomes our held-out data (validation/development)
- **Remember**: Held-out data is NOT the test data
- Train each model using the remaining training data
- Evaluate error on the held-out data
- Choose the model with the smallest held-out error
- Problems:
  - wastes training data
  - if there was an unfortunate split (can be alleviated by repeated random subsampling)

Cross-Validation

K-fold Cross-Validation on N training examples

- Create K equal sized partitions of the training data
- Each partition has N/K examples
- Train using $K-1$ partitions, validate on the remaining partition
- Repeat the same K times, each with a different validation partition
- Choose the model with the smallest average validation error
- Usually K is chosen as 10

Leave-One-Out (LOO) Cross-Validation

Special case of K-fold Cross-Validation when $K = N$

- Each partition is now an example
- Train using $N - 1$ examples, validate on the remaining example
- Repeat the same N times, each with a different validation example
- Choose the model with the smallest average validation error
- can be expensive for large N. Typically used when N is small

Random Subsampling Cross-Validation

- Randomly subsample a fixed fraction $\alpha N (0 < \alpha < 1)$ of examples; call it the validation set
- Training using the rest of the examples, measure error on the validation set
- Repeat K times, each with a different randomly chosen validation set
- Choose the model with the smallest average validation error
- Usually α is chose as 0.1, K as 10

Bootstrapping

- Given a set of N examples
- Idea: Sample N elements from this set with replacement (already sampled elements can be picked again)
- Use this new set as the training data
- The set of examples not selected as the validation data
- For large N, training data consists of about only **63% unique** examples
- Expected model error:

$$e = 0.632 \times e_{\text{test}} + 0.368 e_{\text{training}}$$

- This can break down if we overfit and $e_{\text{training}} = 0$

Bradley Efron & Robert Tibshirani. *Improvements on Cross-Validation: The 632+ Bootstrap Method*

Information Criteria based methods

- Akaike Information Criteria (AIC)

$$\mathrm{AIC} = 2k - 2\log(\mathcal{L})$$

- Bayesian Information Criteria (BIC)

$$\mathrm{BIC} = k\log(N) - 2\log(\mathcal{L})$$

- k: # of model parameters
- N: # of data examples
- $\mathcal{L}$: maximum value of the model likelihood
- Applicable for probabilistic models
- AIC/BIC penalize model complexity

Feature Selection

Selecting a useful subset from all the features. Why?

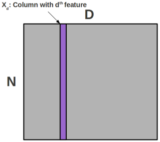
- Some algorithms scale (computationally) poorly with increased dimension
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization
- Removal of features can increase (relative) margin (and generalization)
- Reduces data set and resulting model size
- Note: Feature Selection is different from Feature Extraction
  - The latter transforms original features to get a small set of new features
  - More on feature extraction when we cover **Dimensionality Reduction**

Feature Selection Methods

- Methods agnostic to the learning algorithm
  - Preprocessing based methods
    - E.g., remove a binary feature if it's ON in very few or most examples
  - Filter Feature Selection methods
    - Use some ranking criteria to rank features
    - Select the top ranking features
- Wrapper Methods (keep the learning algorithm in the loop)
  - Requires repeated runs of the learning algorithm with different set of features
  - Can be computationally expensive

Filter Feature Selection

- Uses heuristics but is much faster than wrapper methods



- **Correlation Criteria**: Rank features in order of their correlation with the labels

$$R(X_d, \mathbf{y}) = \frac{cov(X_d, \mathbf{y})}{\sqrt{var(X_d)var(\mathbf{y})}}$$

- **Mutual Information Criteria**:

$$MI(X_d, \mathbf{y}) = \sum_{X_d \in \{0,1\}} \sum_{y \in \{-1,+1\}} P(X_d, \mathbf{y}) \log \frac{P(X_d, \mathbf{y})}{P(X_d)P(y)}$$

  - high mutual information means high relevance of that feature
  - Note: these probabilities can be easily estimated form the data

Wrapper Methods

- Forward Search
  - Start with no features
  - Greedily include the most relevant feature
  - Stop when selected the desired number of features
- Backward Search
  - Start with all features
  - Greedily remove the least relevant feature
  - Stop when selected the desired number of features
- Inclusion/Removal criteria uses cross-validation

Acknowledgements

Slides adapted from Dr. Bert Huang's *Machine Learning* at Virginia Tech, Dr. Tommi S. Jaakkola's *Introduction to Machine Learning* at MIT, and Dr. Piyush Rai's *Machine Learning* at Utah.