CS 559 Machine Learning

Lecture 3: Linear Classification

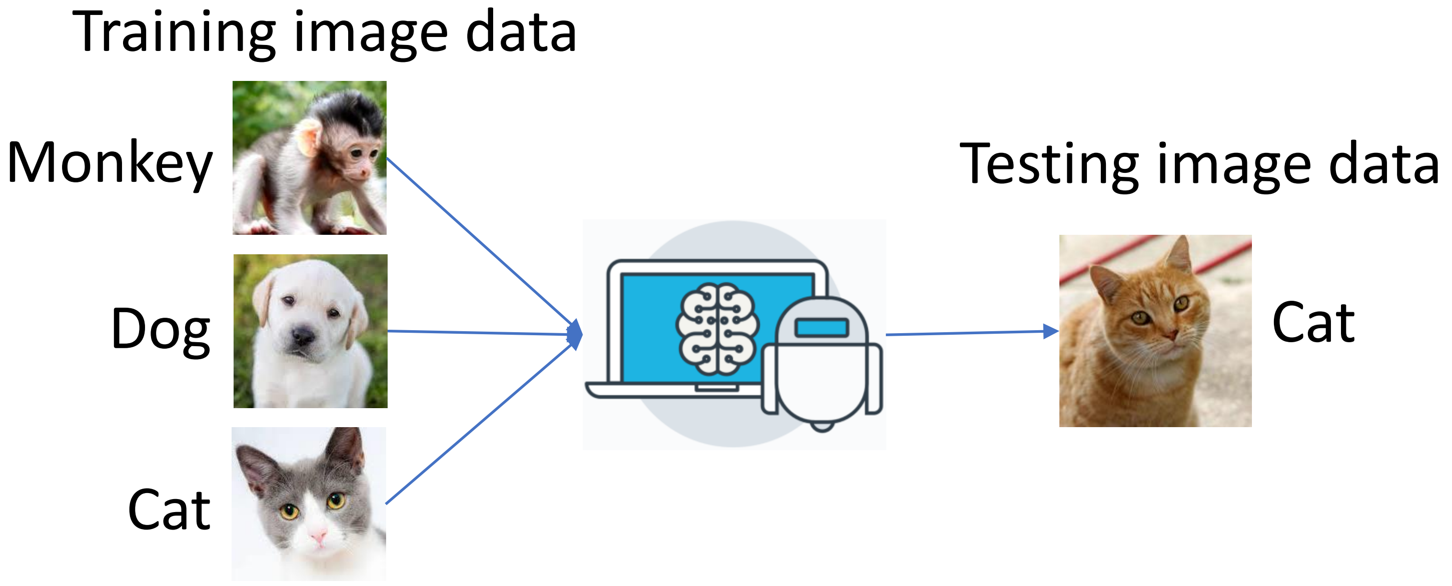
Ping Wang
Department of Computer Science
Stevens Institute of Technology

Today's Lecture

- Generative vs Discriminative Classification
- Linear Discriminant Analysis
- The Perceptron Algorithm
- Model Selection

Classification Task

- Task: find a function f that classifies examples into a given set of discrete classes $\{C_1, C_2, \ldots, C_k\}$.



Features/Attributes (x) → Function f → $y \in \{C_1, C_2, \ldots, C_k\}$

Linear Classification

- Decision boundaries/surfaces: divide the input space into decision regions.
- For linear classification, the decision boundaries are linear functions of the input x.
- Linear separable: datasets that can be exactly separated by linear decision boundaries are linear separable.

Generative and Discriminative Approach

Decision Theory for Classification

- Decision theory, combined with probability theory, allows us to make optimal decisions in situations involving uncertainty.
  - Training data: input values X and target values y
  - Learning stage: use the training data to learn a model for $p(C_k|x)$, where C_k represents the class k.
  - Decision stage: use the learnt posterior probabilities to make optimal class assignments.

Generative Methods

- Solve the inference problem by estimating the class-conditional density $p(x|C_k)$ for each class C_k
- Estimate the class prior probability $p(C_k)$
- Use Bayes' theorem to get the class posterior probability:

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}$$

where $\quad p(x) = \sum_{k=1}^{K} p(x|C_k)p(C_k)$

- Use decision theory to determine class label for each new input x.

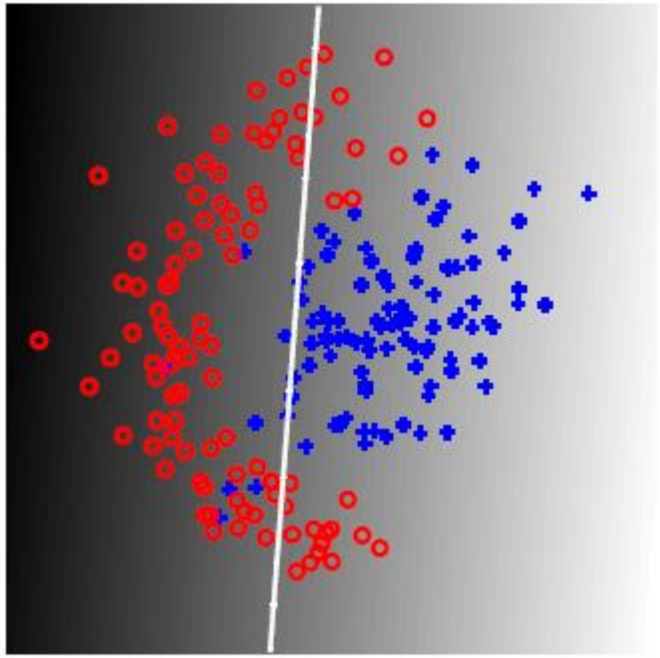
Discriminative Methods

- Directly solve the inference problem of estimating the class posterior probabilities $p(C_k|x)$.
- Discriminative Functions: Find a function $f(x)$ which maps each input directly onto a class label. Probabilities play no role here.
- Use decision theory to determine class label for each new input x.

Linear Discriminant Function

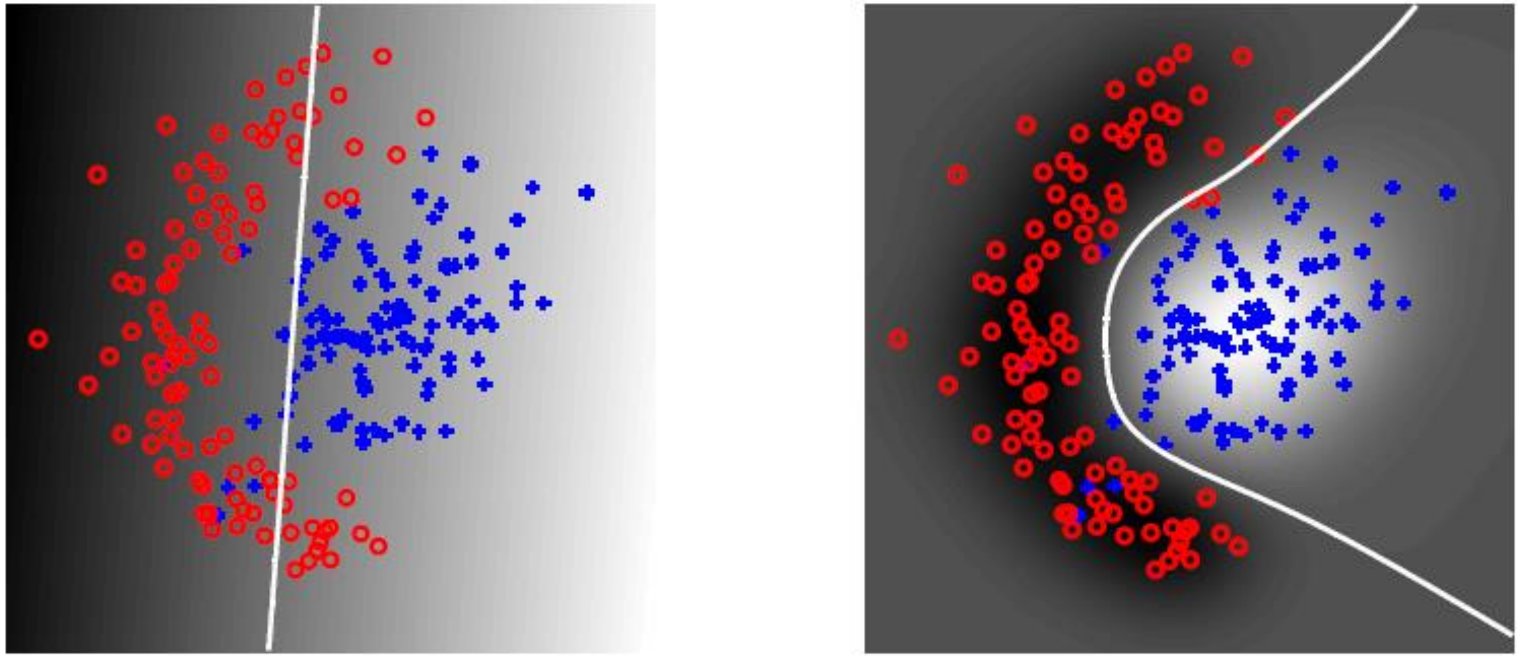
Binary Classification

- Task: Assign each data point to one of two classes (e.g., 0/1, 1/-1).
- Examples:
  - Is there a face in this image?
  - Is this email spam or not?
  - Based on this brain-scan, does this patient have a given disease or not?
  - Will this customer buy this product or not?
  - Will this patient be re-hospitalized or not?



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

Linear Discriminant Function



Linear algorithms can be used together with nonlinear feature spaces or nonlinear basis functions to solve nonlinear classification problems!

Linear Discriminant Function

- Discriminative methods learn the class posterior $p(C_k|x)$.
- The discriminant function here directly assigns the class label for each input vector x.
- The simple way would be the linear discriminant function.

Linear Discriminant Function

- Linear discriminants separate the space by a hyperplane, and the parameters define its normal vector.
- Decision function: $f(x) = w^T x + w_0$, where w represents the weight vector, and w_0 is the bias that determines the location of the decision boundary.
- Classification task*:*

$$\text{If } f(x) \geq 0, x \rightarrow \text{class 1}$$

$$\text{If } f(x) < 0, x \rightarrow \text{class -1}$$

- The decision boundary is defined by equation $f(x) = 0$, which is a hyperplane with dimensionality $D - 1$ (D is the dimension of the input space).

Geometrical Properties

- Decision boundary: $f(x) = w^T x + w_0 = 0$
- Let x_1, x_2 be two points which lie on the decision boundary

$$f(x_1) = w^T x_1 + w_0 = 0$$
$$f(x_2) = w^T x_2 + w_0 = 0$$
$$w^T (x_1 - x_2) = 0$$

- Therefore, w represents the orthogonal direction to the decision boundary. It is the normal vector to the hyperplane, and points into the positive class or negative class.

Geometrical Properties: Dot Product

- The decision boundary is the dot product of two vectors w and x, which can be considered as the projection of x onto w.
  - If $f(x) > 0$, $x \rightarrow$ class 1
  - If $f(x) < 0$, $x \rightarrow$ class -1
  - If $f(x) = 0$, x is on the decision boundary.
- Therefore, the decision boundary is the plane that is perpendicular to the weight vector w.

Sign is Important!

- We can observe that the sign of $f(x)$ is important for classification.
- The scale of the weight w does not matter.
- Therefore, we can use the normalized weight w with length $||w|| = 1$.

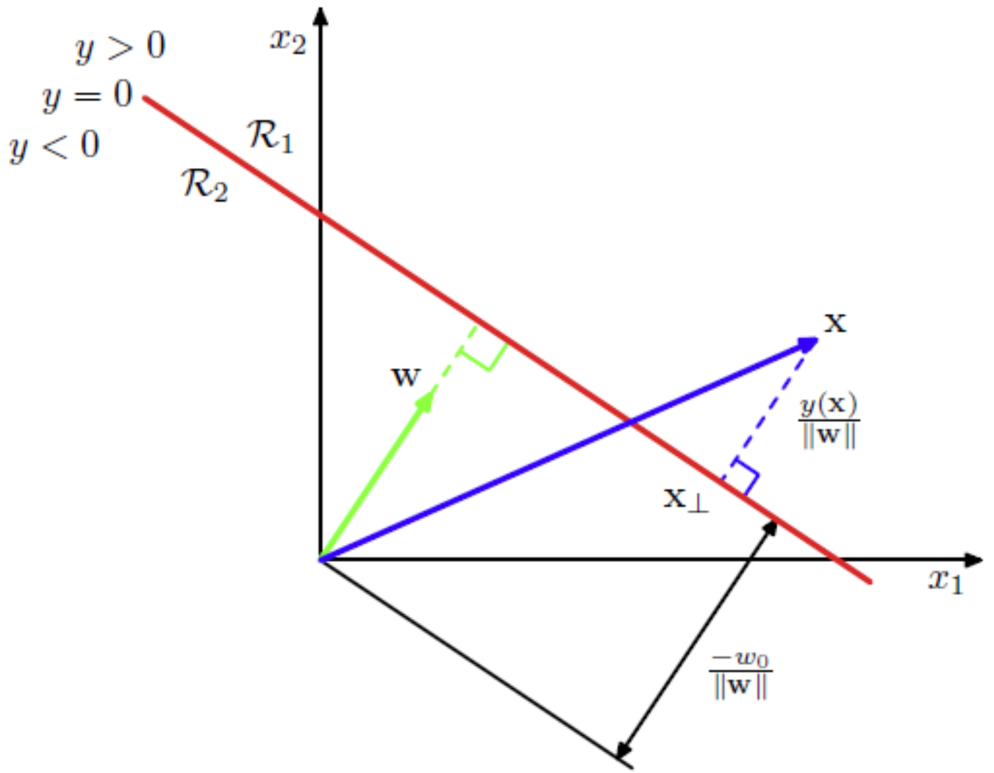
Geometrical Properties Cont.

- $w^{*T} = \frac{w^T}{\|w\|}$
- $w^{*T}(x - x_0)$ is the projection of $(x - x_0)$ onto the direction of w^*; x_0 is a point on the decision boundary.
- Thus

$$\frac{w^T}{\|w\|}(x - x_0) = \frac{1}{\|w\|}(w^T x - w^T x_0)$$

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

- When $x = 0$, $\frac{f(x)}{\|w\|} = \frac{\omega_0}{\|w\|}$

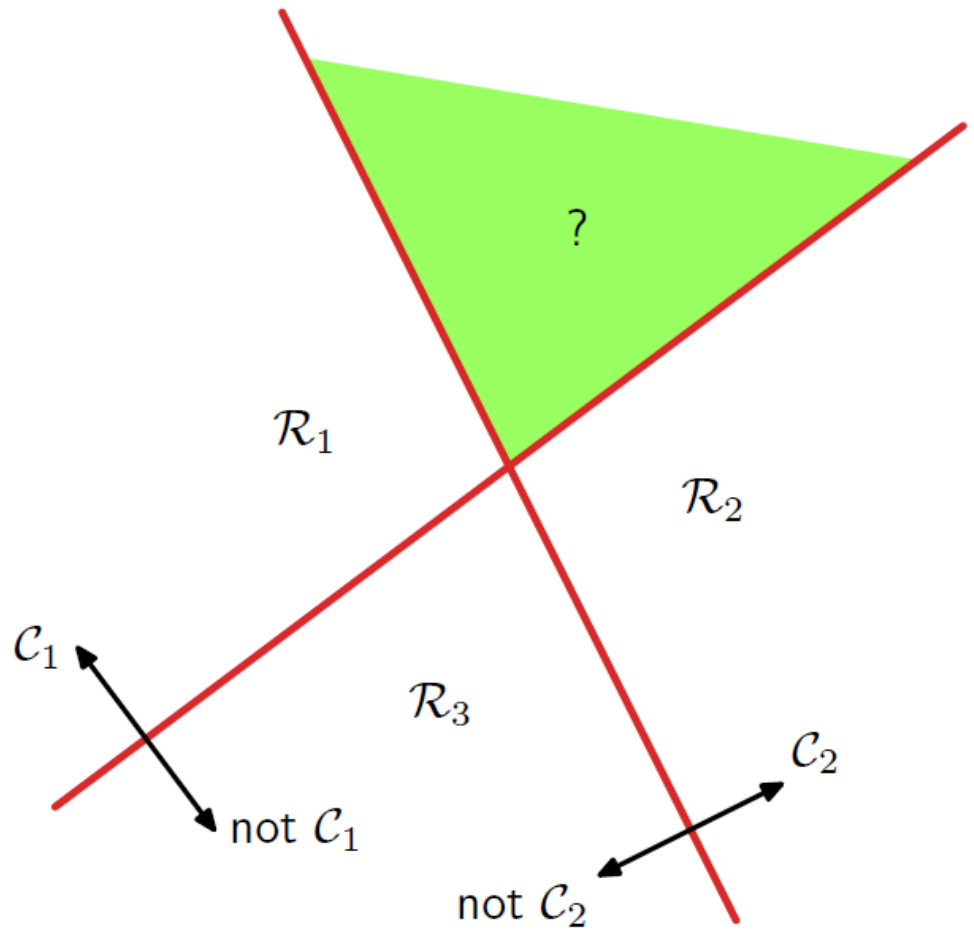


Signed orthogonal distance of the origin from the decision surface

Linear Discriminant Functions: Multiple Classes

one-versus-the-rest:

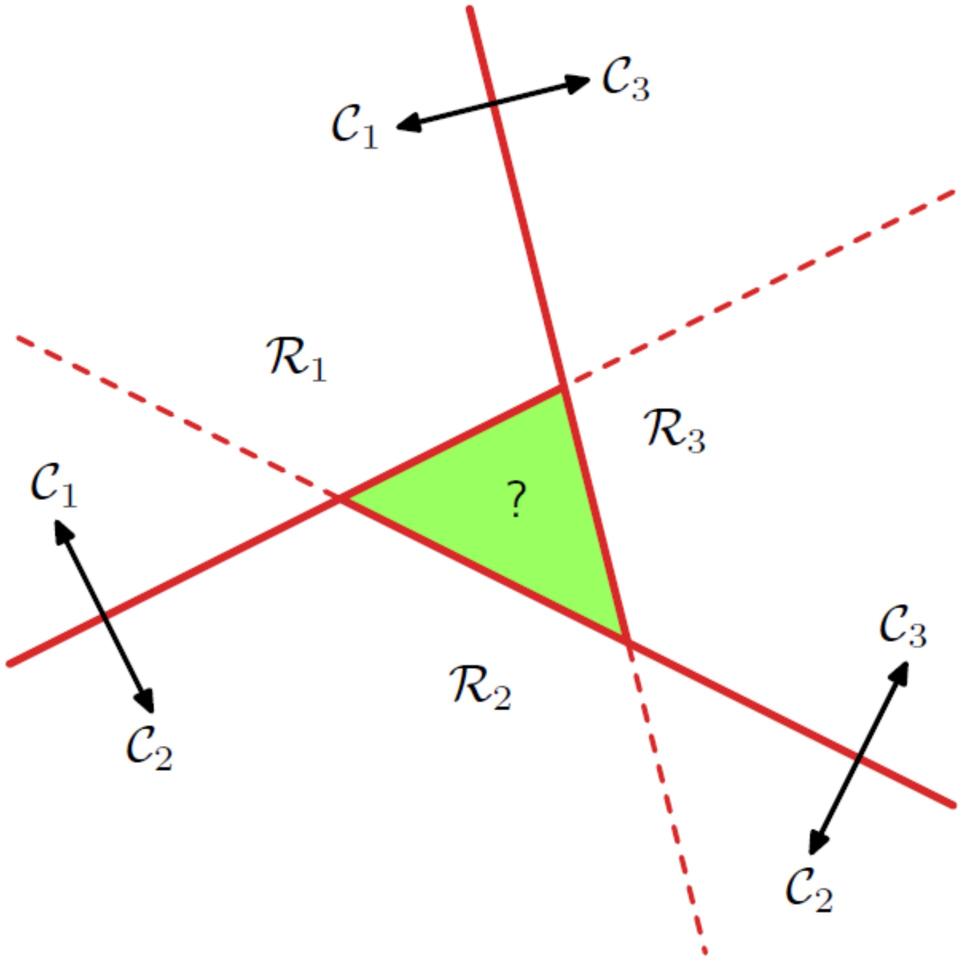
- $K - 1$ classifiers, each of which solves a two-class problem of separating points in a particular class C_k from points not in that class.
- Leads to regions of input space that are ambiguously classified, shown in green.



Linear Discriminant Functions: Multiple classes

one-versus-one:

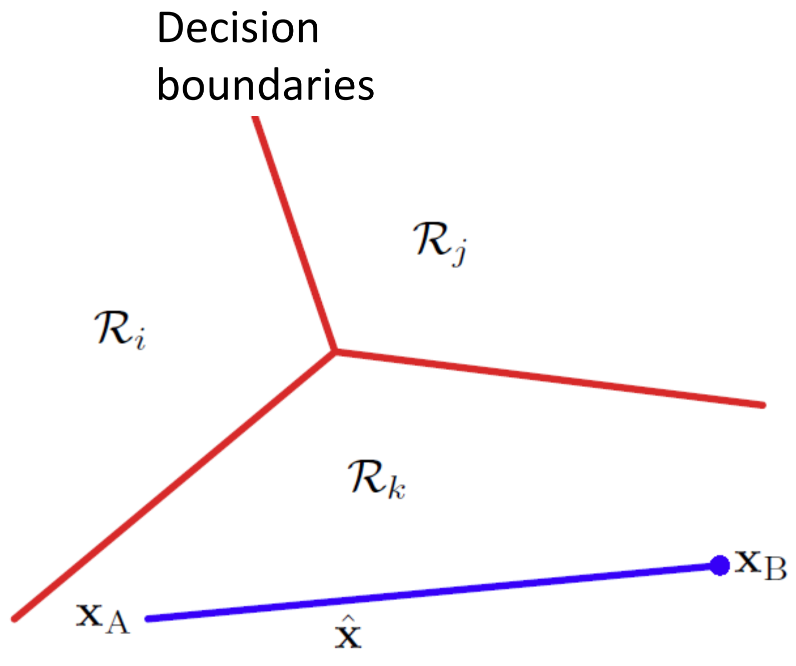
- $\frac{K(K-1)}{2}$ classifiers, one for every possible pair of classes
- Each point is classified by majority voting amongst the discriminant functions.
- Also leads to regions of input space that are ambiguously classified , shown in green.



Linear Discriminant Functions: Multiple classes

- Solution: Consider a single K-class discriminant comprising K linear functions of the form
$$f_k(x) = w_k^T x + w_{k0}$$
- Assign a point x to class C_k if
$$f_k(x) > f_j(x)\ \forall j \neq k.$$
- The decision boundary between class C_k and class C_j is given by:
$$f_k(x) = f_j(x)$$

A hyperplane: $\Rightarrow \left(w_k - w_j\right)^T x + \left(w_{k0} - w_{j0}\right) = 0$



If two points lie in the same decision region, then any point on the line segment must lie in the same region.

Linear Classification Algorithms

- Mis-classification rate $C(w) = \frac{1}{N}\sum_n \delta[f(x_n) = y_n]$ (i.e. average number of errors) difficult to optimize over w, and might have multiple solutions.
- Many algorithms can be derived by replacing C by another cost-function which can be optimized.
  - Least-square Classification
  - Fisher's Linear Discriminant
  - Rosenblatts' Perceptron
  - Logistic Regression
  - Support Vector Machines

Least Square Classification

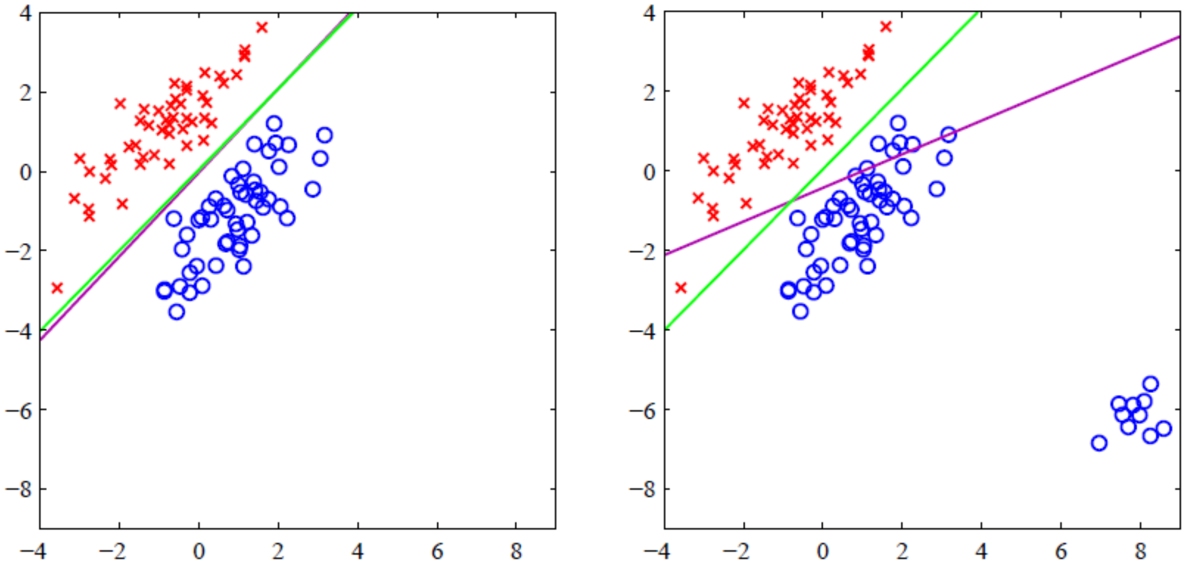
Least Square Classification

- We have to fit the function $f(x) = xw^T + \omega_0$ to data.
- Simply do a linear regression from x to y by minimizing the sum-of-squared errors $\sum_n (f(x_n) - y_n)^2$.
- $w_{reg} = (\sum_n x_n x_n^T)^{-1} \sum_n x_n y_n$

- Questions: In what situations might this be a bad idea?

Least Square Classification

- Least squares solutions lack robustness to outliers.
- The decision boundary changed significantly when extra data points are added.

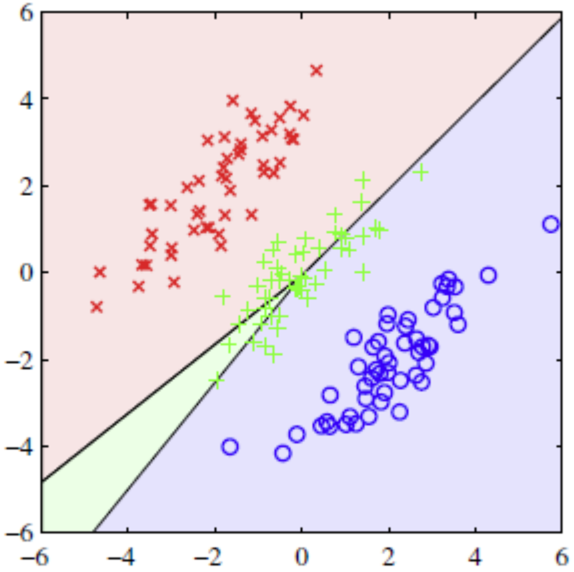


Bishop PRML Figure 4.4.

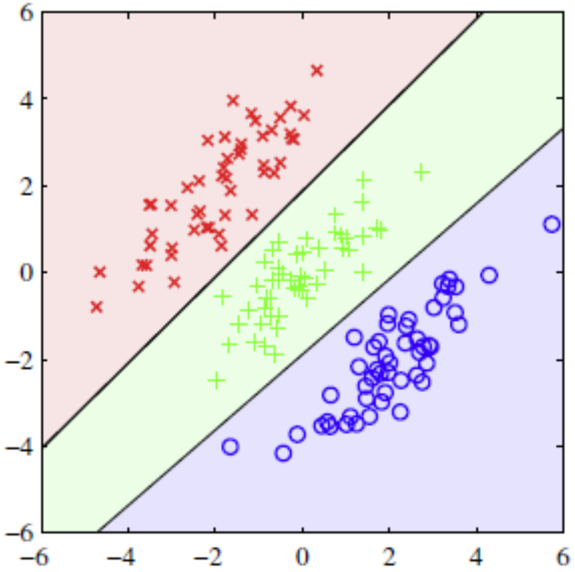
Purple: least squares. Green: logistic regression

Least Square Classification

- Example of a synthetic data set with three classes.
- The background colors denote the respective classes of the decision regions.



Least square discriminant



Logistic regression

Bishop PRML Figure 4.5

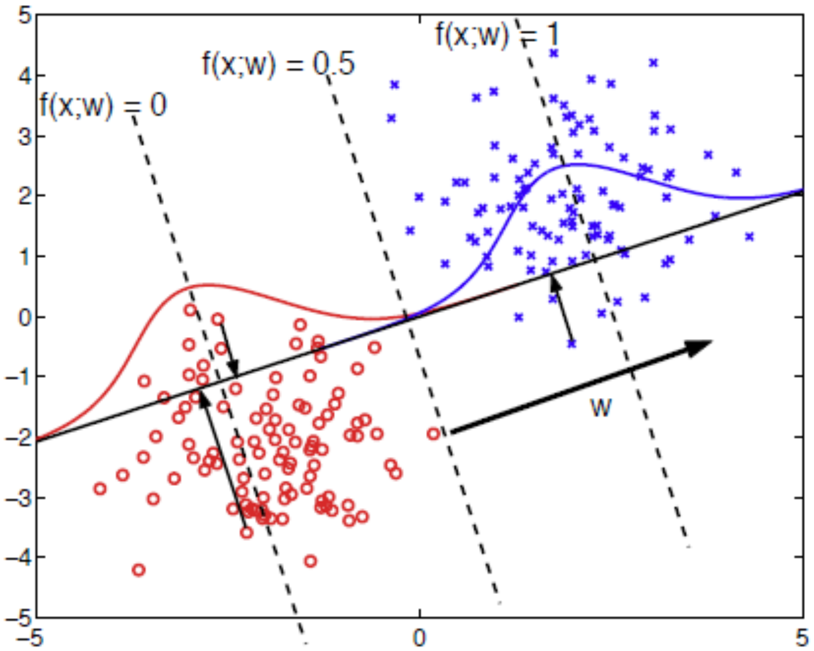
Fisher’s Linear Discriminant

Classification via Projection

- A linear function: $f(x) = w^T x + w_0$ assuming in 2D, projects each point x to a line parallel to w:

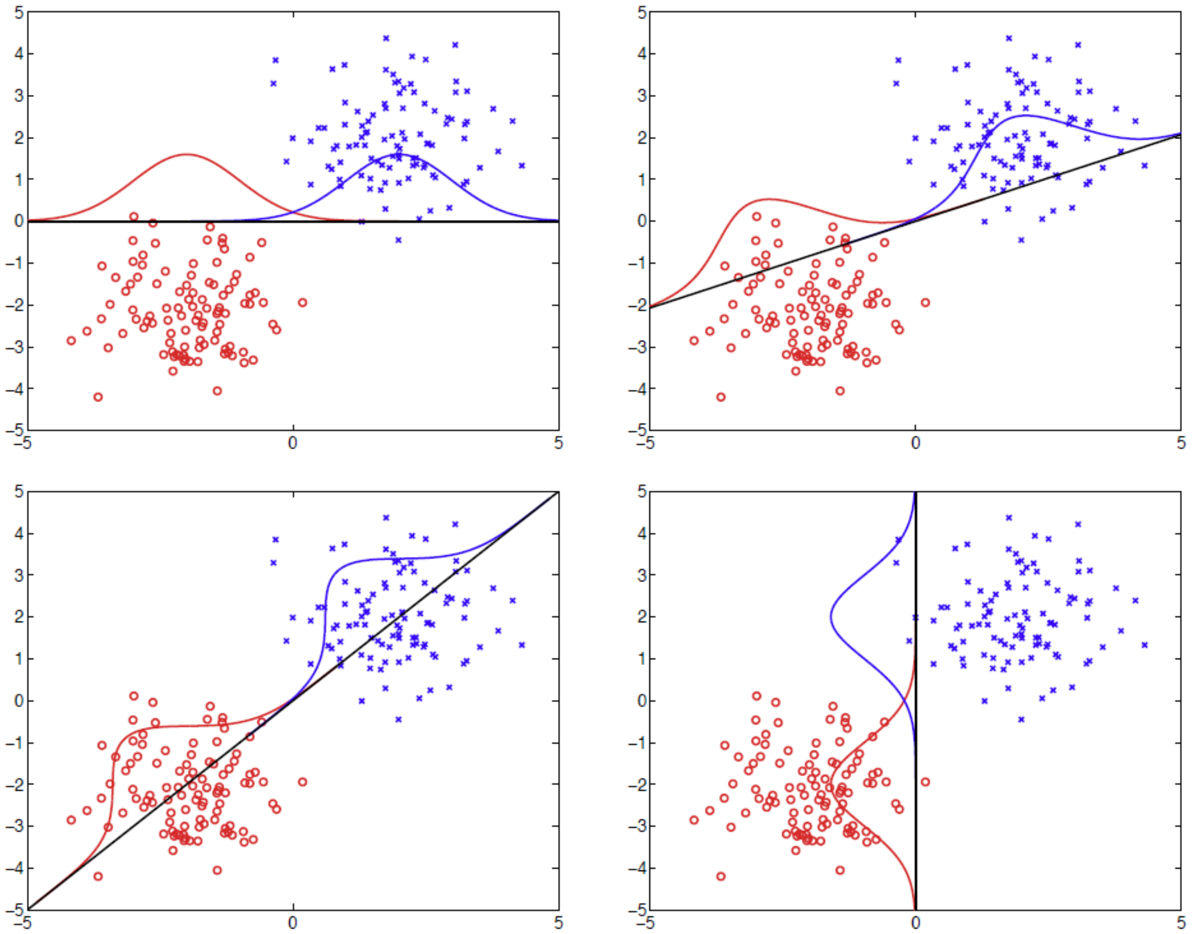
$$x_1 \to z_1 = w^T x_1$$
$$x_2 \to z_2 = w^T x_2$$
$$\dots$$
$$x_N \to z_N = w^T x_N$$

- We can study how well the projected points $z_1, \dots, z_N$ viewed as functions of w are separated across the classes.



Classification via Projection

- By varying w we get different levels of separation between the projected points.
- Find w that maximizes the separation of the projected points across classes.
- Quantify the separation (overlap) in terms of means and variances of the resulting 1-dimensional class distributions.



Optimizing the Projection

- Objective:
  - Find w that maximizes the separation of the projected points across classes.
- Solution:
  - 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.
- For binary classification, suppose we project x onto one dimensional:

$$f = w^T x$$

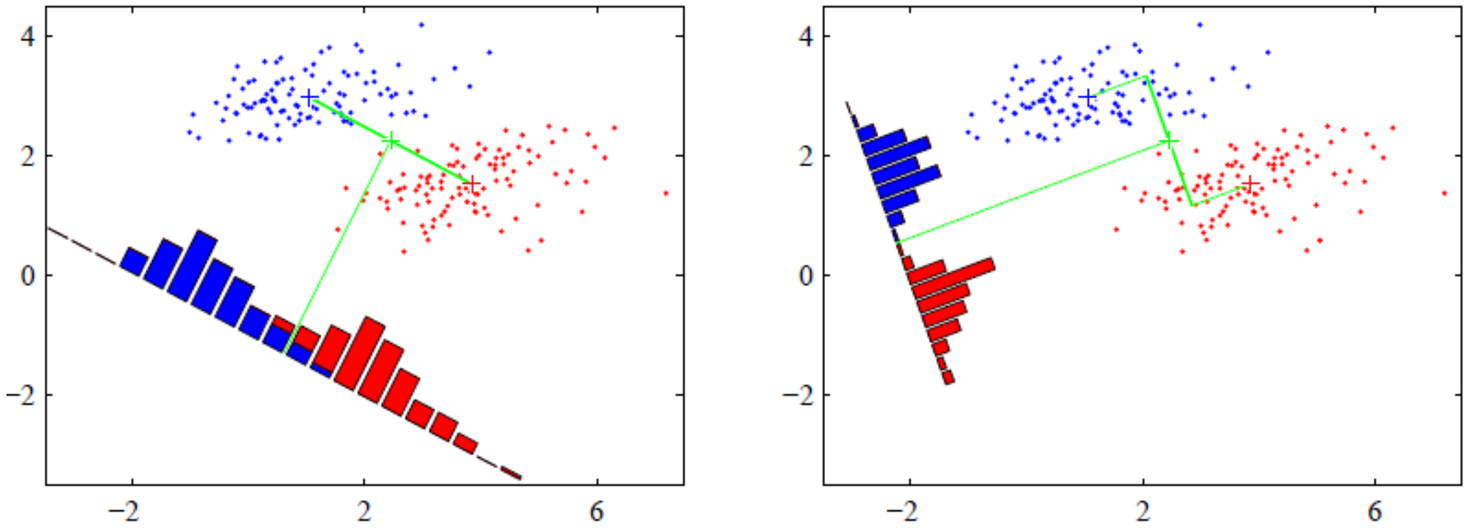
- A threshold t can be set to assign the label for x:

$$\text{If } f \leq t, x \rightarrow C_1$$
$$\text{If } f > t, x \rightarrow C_2$$

- In general, the projection leads to considerable loss of information, and classes well separated in the original space may strongly overlap in one dimension.

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 want to find the linear projection that best separates the data, i.e. best discriminates data of different classes.



Fisher’s Linear Discriminant

- We use N_1 and N_2 to represent the number of samples for class C_1 and C_2, respectively.
- Consider the normalized weight vector w with $||w|| = 1$.
- Then, $w^T x$ is the projection of x onto the direction of w.
- Objective: find the projections of $w^T x$ for $x \in C_1$ and $x \in C_2$ can be well separated (maximize the class separation).

How to Measure the Separation?

A measure of the separation between the projected points is the difference of the sample means.

- Sample mean vector of class C_1:

$$m_1 = \frac{1}{N_1} \sum_{x \in C_1} x$$

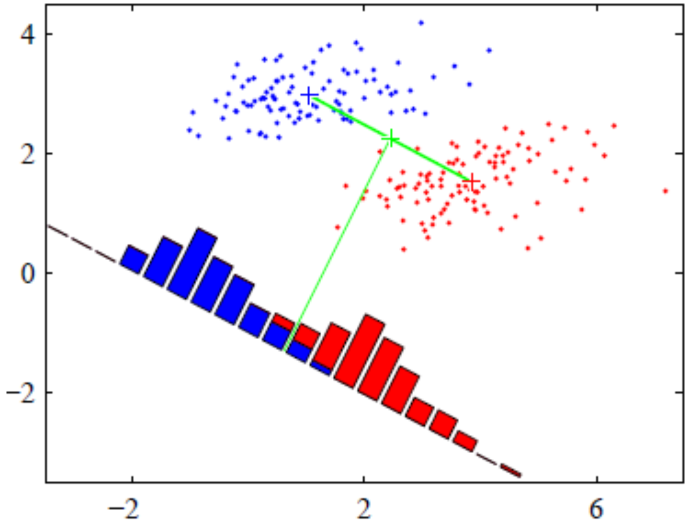
- Sample mean vector of class C_1 after projection:

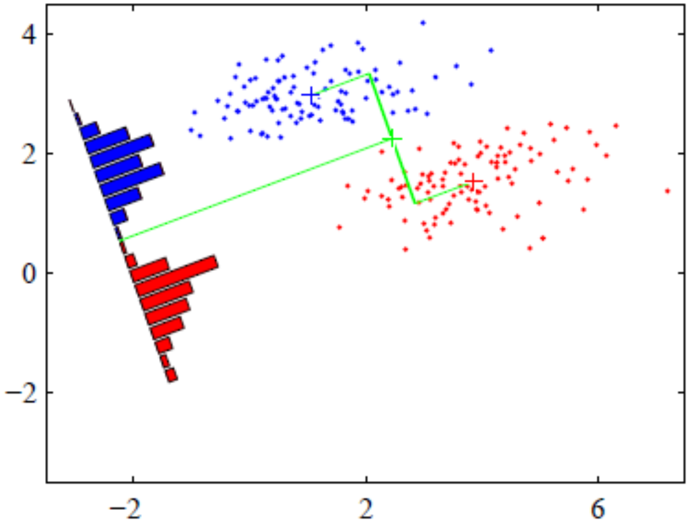
$$m_1' = \frac{1}{N_1} \sum_{x \in C_1} w^T x = w^T m_1$$

- Objective is to choose w that can maximize the separation $|m_1' - m_2'| = w^T |m_1 - m_2|$

How to Measure the Separation?

- Choose w that maximizes projection-distance of class means $|m_1' - m_2'| = w^T|m_1 - m_2|$
- We find that $w \propto m_1 - m_2$
- Limitation: maximizing distance between means, but the projected variances within each class might be large (large class overlap).





How to Measure the Separation?

- 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.
- Variance of the projected samples of class C_1:

$$s_1^2 = \sum_{x \in C_1} (w^T x - m_1')^2$$

- Total within-class variance of the projected samples will be:

$$s_1^2 + s_2^2$$

- Fisher linear discriminant analysis: Fisher criterion defined by the ratio of the between-class variance to the within-class variance.

$$\arg\max_w \frac{|m_1' - m_2'|^2}{s_1^2 + s_2^2}$$

Fisher’s Linear Discriminant

- Define $J(w) = \frac{|m_1' - m_2'|^2}{s_1^2 + s_2^2}$. To obtain $J(w)$ as an explicit function of w, we define the following matrices:

$$S_1 = \sum_{x \in C_1} (x - m_1)(x - m_1)^T$$

- Within-class covariance matrix: $S_W = S_1 + S_2$
- Then

$$s_1^2 = \sum_{x \in C_1} (w^T x - m_1')^2 = \sum_{x \in C_1} (w^T x - w^T m_1)^2$$
$$= \sum_{x \in C_1} w^T (x - m_1)(x - m_1)^T w = w^T S_1 w$$

Fisher's Linear Discriminant

- Therefore, $s_1^2 = w^T S_1 w$ and $s_2^2 = w^T S_2 w$.
- Further:

$$s_1^2 + s_2^2 = w^T S_1 w + w^T S_2 w = w^T (S_1 + S_2) w = w^T S_W w$$

- Similarly:

$$\begin{aligned} |m_1' - m_2'|^2 &= (w^T m_1 - w^T m_2)^2 \\ &= w^T (m_1 - m_2)(m_1 - m_2)^T w \\ &= w^T S_B w \end{aligned}$$

- Where $S_B = (m_1 - m_2)(m_1 - m_2)^T$ is the between-class covariance matrix.

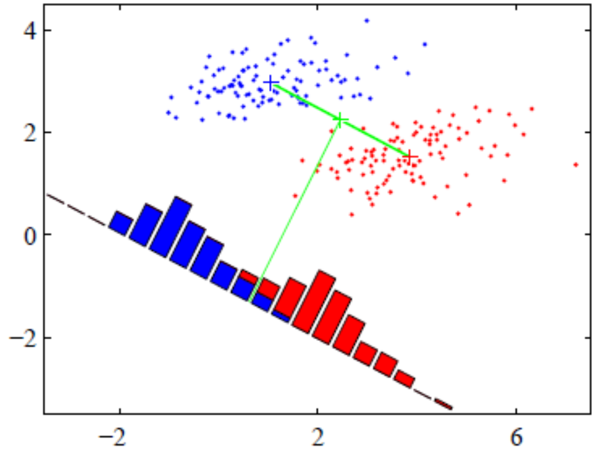
Fisher's Linear Discriminant

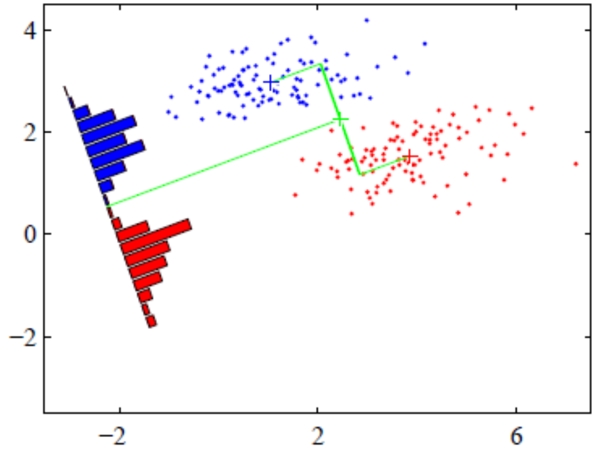
- Therefore, $J(w) = \frac{\left|m_1' - m_2'\right|^2}{s_1^2 + s_2^2} = \frac{w^T S_B w}{w^T S_W w}$
- Differentiating with respect to w, $J(w)$ is maximized when $(w^T S_B w) S_W w = (w^T S_W w) S_B w$
- We can observe that $S_B w = (m_1 - m_2)(m_1 - m_2)^T w$ always in the direction of $(m_1 - m_2)$ since $(m_1 - m_2)^T w$ is a scalar.
- Also, we only care about the direction of w, so we can drop the scaler factors $(w^T S_B w)$ and $(w^T S_W w)$.
- Therefore, we have the solution:

$$w \propto S_W^{-1}(m_1 - m_2)$$

Summary of Fisher's Linear Discriminant

- $m_1 = \frac{1}{N_1}\sum_{x\in C_1} x,\, m_2 = \frac{1}{N_2}\sum_{x\in C_2} x$
- Separation: Maximize projection-distance of class means $w \propto m_1 - m_2$
- Maximizing distance between means, but the projected variances within each class might be large.
- Fix: Maximize the ratio of between-class variance to within-class variance ("signal to noise").
- Fisher criterion $J(w) = \frac{|m_1' - m_2'|^2}{s_1^2 + s_2^2}$
- Updated solution: $w = S_W^{-1}(m_1 - m_2)$





Fisher’s Linear Discriminant

- Gives the linear function with the maximum ratio of between-class scatter to within-class scatter.
- The classification problem 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.
- $S_W = \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_i - m_k)(x_i - m_k)^T$
- $S_B = \sum_{k=1}^{K} N_k (m_k - m)(m_k - m)^T$ where m is the global mean; N_k is the number of examples in class k.
- Solve: $S_B v = \lambda S_W v$ the generalized eigenvalue problem
- At most $K - 1$ distinct eigenvalues
- The optimal projection matrix V to a subspace of dimension k is given by the eigenvectors corresponding to the largest k eigenvalues.

More details can be found in Bishop PRML Section 4.1.6

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

The Perceptron Algorithm

- First learning algorithm for neural networks. (Frank Rosenblatt, 1957)
- Originally introduced for character classification, where each character is represented as an image.
- It is a type of linear classifier that makes predictions based on a linear predictor function and a set of weights with the input feature vector.

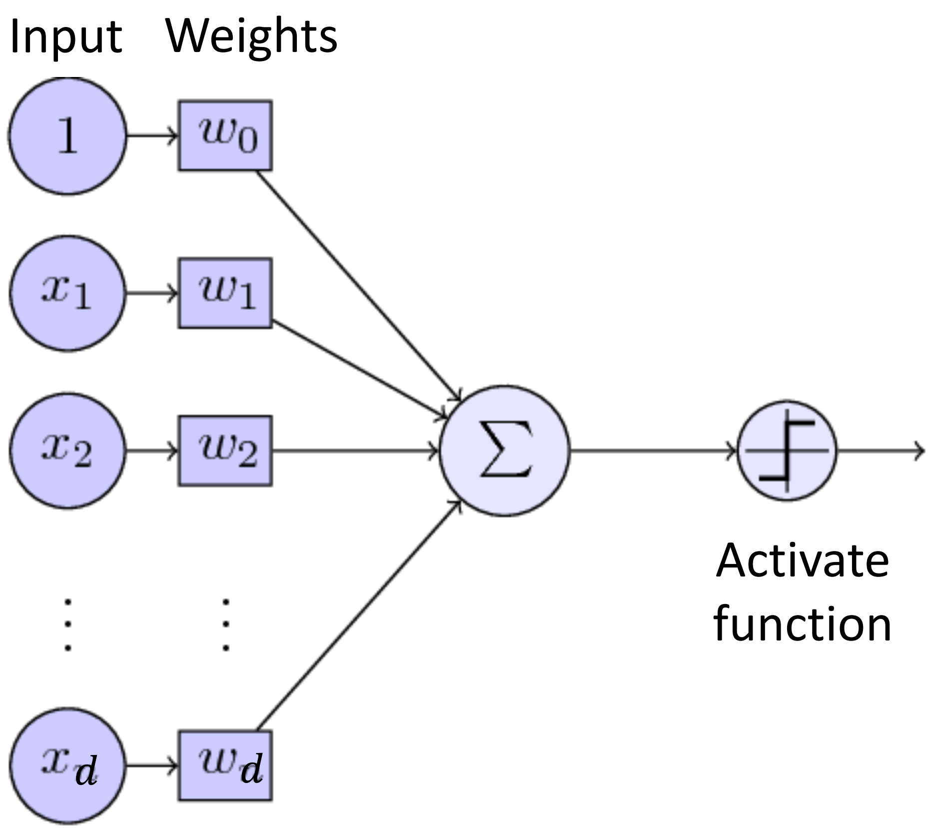
The Perceptron Algorithm

- Total input to output node:

$$\sum_j w_j x_j$$

- Output unit performs the function (activation function):

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



Perceptron: Learning Task

- Goal: compute a mapping from inputs to the outputs.
- Example: two-class character recognition problem.
  - Training set: a set of images representing either the character ‘a’ or the character ‘b' (supervised learning);
  - Learning task: learn the weights so that when a new unlabeled 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 $\{x_k\}$
  - 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: $x = (x_1, x_2, \ldots, x_d)^T, w = (w_1, w_2, \ldots, w_d)^T$
- θ: represents the threshold of the output unit
- Unit output: $w^T x = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$
- Output class 1 if $w^T x - \theta \geq 0$
- To eliminate the explicit dependence on θ, output class 1 if $w^T x \geq 0$.

Perceptron: Learning Algorithm

- The objective is to learn the weights so that the perceptron can correctly discriminate elements of C_1 from elements of C_2
- Given x in input, if x is classified correctly, weights are unchanged, otherwise:

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

Perceptron: Learning Algorithm

- It is online:
  - Only process one example at a time, instead of considering the entire dataset at the same time.
- Error-Driven Updating:
  - If it is doing well, it doesn't update the parameters.
  - Only when the prediction is incorrect, it updates the parameters.
  - The parameters are updated in a way that it would do better on this example next time around.
- Simple, but works well!

Perceptron: Learning Algorithm

- **1^{st} case:** $x \in C_1$, but was classified in C_2. In other words, the correct answer is 1, which corresponds to $w^T x \geq 0$, but the model provides $w^T x < 0$. In the next round after updating the parameters to w'^T, we want to get closer to the correct answer (be greater):
$$w^T x < w'^T x$$
- Verify if it would do better after updating w as $w' = w + x$.
$$\begin{aligned} w'^T x &= (w + x)^T x \\ &= w^T x + x^T x \\ &= w^T x + ||x||^2 \end{aligned}$$
- Since $||x||^2 > 0$, the condition is verified.

Perceptron: Learning Algorithm

- **2^{nd} case:** $x \in C_2$, but was classified in C_1. In other words, the correct answer is 0, which corresponds to $w^T x < 0$, but the model provides $w^T x \geq 0$. In the next round after updating the parameters to w'^T, we want to get closer to the correct answer (be smaller):

$$w^T x > w'^T x$$

- Verify if it would do better after updating w as $w' = w - x$.

$$\begin{aligned} w'^T x &= (w - x)^T x \\ &= w^T x - x^T x \\ &= w^T x - ||x||^2 \end{aligned}$$

- Since $||x||^2 > 0$, the condition is verified.

The Perceptron Algorithm: Example

Example:

$(-1,2)\ -$ ✗

$(1,0)\ +$ ✓

$(1,1)\ +$ ✗

$(-1,0)\ -$ ✓

$(-1,-2)\ -$ ✗

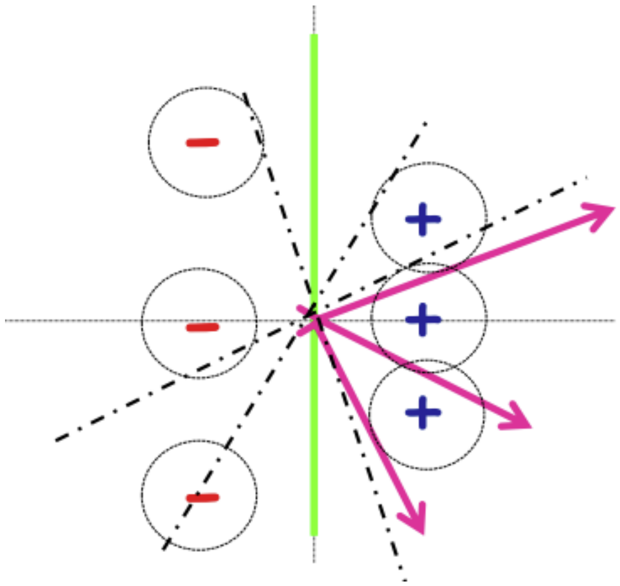
$(1,-1)\ +$ ✓

$$w_1 = (0,0)$$

$$w_2 = w_1 - (-1,2) = (1,-2)$$

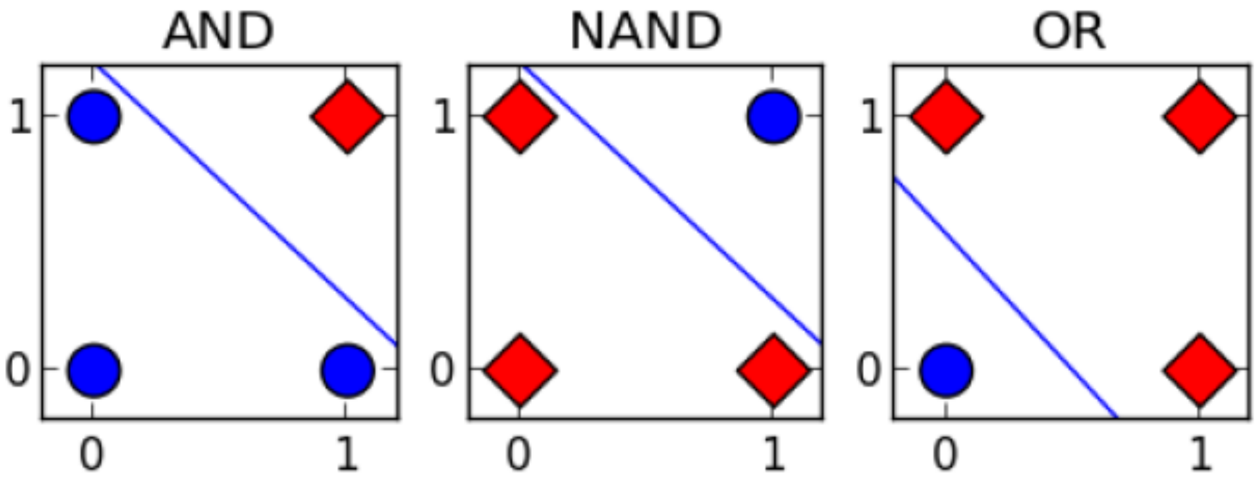
$$w_3 = w_2 + (1,1) = (2,-1)$$

$$w_4 = w_3 - (-1,-2) = (3,1)$$



Representational Power of Perceptrons

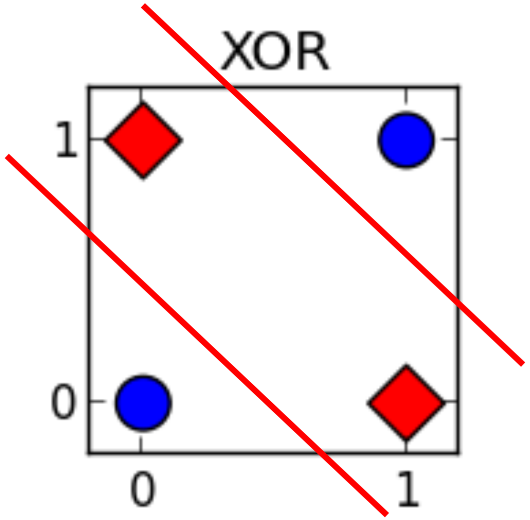
- Marvin Minsky and Seymour Papert, “Perceptrons" 1969: The perceptron can only solve problems with linearly separable classes.
  - The functions can be drawn in 2-dim graph and a single straight line separates values in two parts.
- Examples of linearly separable Boolean functions:



Representational Power of Perceptrons

- The perceptron cannot model the non-linearly separable functions: Logical XOR function (computes the logical exclusive).
  - Input: Two input arguments with values in {0,1}.
  - Output: 1 if and only if two inputs have different values.
  - For such functions, we have to use multi-layer feed-forward network.

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0



Summary of The Perceptron Algorithm

- For a random sequence $x_1, x_2, \dots, x_k$, with $x_i (i = 1, \dots, k) \in C_1 \; or \; C_2$
- For each x, if it is correctly classified, then $w_{k+1} = w_k$, otherwise,

$$w_{k+1} = \begin{cases} w_k + x \;\; if \; x \in C_1 \\ w_k - x \;\; if \; x \in C_2 \end{cases}$$

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

$$\begin{cases} w^T x \geq 0 \;\; if \; x \in C_1 \\ w^T x < 0 \;\; if \; x \in C_2 \end{cases}$$

- The learning rule will find such solution after a finite number of steps.

Model Selection

What is Model Selection?

Given a set of models $M = \{M_1, M_2, \dots, M_R\}$, choose the model that is expected to do the best on the test data. 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 5 or 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:

$$e = 0.632e_{test} + 0.368e_{training}$$

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

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

Information Criteria Based Methods

- Akaike Information Criteria (AIC)

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

- Bayesian Information Criteria (BIC)

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

where

- 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; the smaller the better.

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 (we will discuss in 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

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

- Mutual Information Criteria

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

  - High mutual information means high relevance of that feature
  - 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

Summary of Today's Lecture

- Generative vs Discriminative Classification
- Linear Discriminant Analysis
- The Perceptron Algorithm
- Model Selection