Linear Models for Classification

K Sri Rama Murty

IIT Hyderabad

ksrm@ee.iith.ac.in

March 22, 2022

Classification

Classification

- Given an observation vector $\mathbf{x}$, assign it to one of the K classes

Classification

- Given an observation vector $\mathbf{x}$, assign it to one of the K classes
- The target can take one of the K discrete labels $\mathcal{C}_k$, $k = 1, 2, \cdots K$

Classification

- Given an observation vector $\mathbf{x}$, assign it to one of the K classes
- The target can take one of the K discrete labels $\mathcal{C}_k$, $k = 1, 2, \cdots K$
  - Classify $\mathbf{x}$ =(45Kg, 4.8ft) as adult or kid

Classification

- Given an observation vector $\mathbf{x}$, assign it to one of the K classes
- The target can take one of the K discrete labels $\mathcal{C}_k$, $k = 1, 2, \cdots K$
  - Classify $\mathbf{x}$ =(45Kg, 4.8ft) as adult or kid
  - Classify a given image as face or nonface

Classification

- Given an observation vector $\mathbf{x}$, assign it to one of the K classes
- The target can take one of the K discrete labels $\mathcal{C}_k$, $k = 1, 2, \cdots K$
  - Classify $\mathbf{x}$ =(45Kg, 4.8ft) as adult or kid
  - Classify a given image as face or nonface
  - Classify a speech waveform as one of the 40 phonemes

Classification

- Given an observation vector $\mathbf{x}$, assign it to one of the K classes
- The target can take one of the K discrete labels $\mathcal{C}_k$, $k = 1, 2, \cdots K$
  - Classify $\mathbf{x} =$(45Kg, 4.8ft) as adult or kid
  - Classify a given image as face or nonface
  - Classify a speech waveform as one of the 40 phonemes
- Discriminant function to create separating hyperplane between classes
  Eg. Least squares, Fisher discriminant, perceptron, SVM

Classification

- Given an observation vector $\mathbf{x}$, assign it to one of the K classes
- The target can take one of the K discrete labels $\mathcal{C}_k$, $k = 1, 2, \cdots K$
  - Classify $\mathbf{x}$ =(45Kg, 4.8ft) as adult or kid
  - Classify a given image as face or nonface
  - Classify a speech waveform as one of the 40 phonemes
- Discriminant function to create separating hyperplane between classes
  Eg. Least squares, Fisher discriminant, perceptron, SVM
- Probabilistic approaches to estimate posterior probabilities $P[\mathcal{C}_k/\mathbf{x}]$

Classification

- Given an observation vector $\mathbf{x}$, assign it to one of the K classes
- The target can take one of the K discrete labels $\mathcal{C}_k$, $k = 1, 2, \cdots K$
  - Classify $\mathbf{x}$ =(45Kg, 4.8ft) as adult or kid
  - Classify a given image as face or nonface
  - Classify a speech waveform as one of the 40 phonemes
- Discriminant function to create separating hyperplane between classes
  Eg. Least squares, Fisher discriminant, perceptron, SVM
- Probabilistic approaches to estimate posterior probabilities $P[\mathcal{C}_k/\mathbf{x}]$
  - Discriminative models: Directly estimate $P[\mathcal{C}_k/\mathbf{x}] = f(\mathbf{x}, \mathbf{w})$
    Eg. Logistic regression, DNN classifiers

Classification

- Given an observation vector $\mathbf{x}$, assign it to one of the K classes
- The target can take one of the K discrete labels $\mathcal{C}_k$, $k = 1, 2, \cdots K$
  - Classify $\mathbf{x}$ =(45Kg, 4.8ft) as adult or kid
  - Classify a given image as face or nonface
  - Classify a speech waveform as one of the 40 phonemes
- Discriminant function to create separating hyperplane between classes
  Eg. Least squares, Fisher discriminant, perceptron, SVM
- Probabilistic approaches to estimate posterior probabilities $P[\mathcal{C}_k/\mathbf{x}]$
  - Discriminative models: Directly estimate $P[\mathcal{C}_k/\mathbf{x}] = f(\mathbf{x}, \mathbf{w})$
    Eg. Logistic regression, DNN classifiers
  - Generative models: Arrive at the posterior from joint density $p(\mathbf{x}, \mathcal{C}_k)$
    Eg. PDF Estimation: GMM (RV), HMM (RP)

Discriminant Functions (Binary Classification)

Discriminant Functions (Binary Classification)

- For an input vector $\mathbf{x} \in \mathbb{R}^D$, Linear Discriminant Function is given by

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

Discriminant Functions (Binary Classification)

- For an input vector $\mathbf{x} \in \mathbb{R}^D$, Linear Discriminant Function is given by

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

- Decision Rule: Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$, otherwise assign $\mathbf{x}$ to $\mathcal{C}_2$

Discriminant Functions (Binary Classification)

- For an input vector $\mathbf{x} \in \mathbb{R}^D$, Linear Discriminant Function is given by

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

- Decision Rule: Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$, otherwise assign $\mathbf{x}$ to $\mathcal{C}_2$
- Decision boundary, $y(\mathbf{x}) = 0$, corresponds to a $D - 1$ dim. hyperplane

Discriminant Functions (Binary Classification)

- For an input vector $\mathbf{x} \in \mathbb{R}^D$, Linear Discriminant Function is given by

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

- Decision Rule: Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$, otherwise assign $\mathbf{x}$ to $\mathcal{C}_2$
- Decision boundary, $y(\mathbf{x}) = 0$, corresponds to a $D - 1$ dim. hyperplane
- Weight vector $\mathbf{w}$ is orthogonal to every vector on the decision surface

Discriminant Functions (Binary Classification)

- For an input vector $\mathbf{x} \in \mathbb{R}^D$, Linear Discriminant Function is given by

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

- Decision Rule: Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$, otherwise assign $\mathbf{x}$ to $\mathcal{C}_2$
- Decision boundary, $y(\mathbf{x}) = 0$, corresponds to a $D - 1$ dim. hyperplane
- Weight vector $\mathbf{w}$ is orthogonal to every vector on the decision surface
- Weight vector $\mathbf{w}$ determines the orientation of the decision surface

Discriminant Functions (Binary Classification)

- For an input vector $\mathbf{x} \in \mathbb{R}^D$, Linear Discriminant Function is given by

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

- Decision Rule: Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$, otherwise assign $\mathbf{x}$ to $\mathcal{C}_2$
- Decision boundary, $y(\mathbf{x}) = 0$, corresponds to a $D-1$ dim. hyperplane
- Weight vector $\mathbf{w}$ is orthogonal to every vector on the decision surface
- Weight vector $\mathbf{w}$ determines the orientation of the decision surface
- Bias parameter w_0 determines the location of the decision surface

Discriminant Functions (Binary Classification)

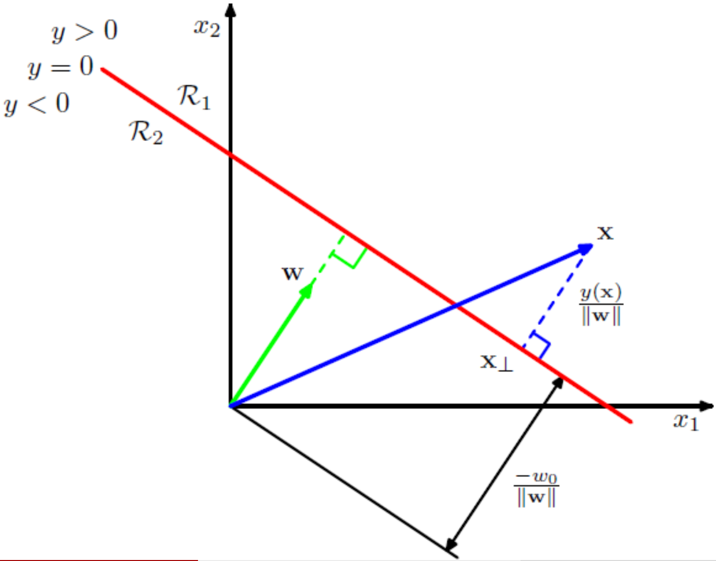
- For an input vector $\mathbf{x} \in \mathbb{R}^D$, Linear Discriminant Function is given by

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

- Decision Rule: Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$, otherwise assign $\mathbf{x}$ to $\mathcal{C}_2$
- Decision boundary, $y(\mathbf{x}) = 0$, corresponds to a $D-1$ dim. hyperplane
- Weight vector $\mathbf{w}$ is orthogonal to every vector on the decision surface
- Weight vector $\mathbf{w}$ determines the orientation of the decision surface
- Bias parameter w_0 determines the location of the decision surface
- Normal distance from origin to the decision surface is given by

$$\frac{\mathbf{w}^\mathsf{T}\mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$

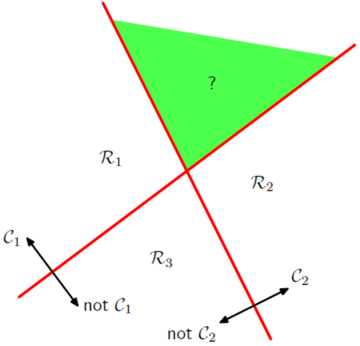
Geometry of Decision Boundary in 2D



Extending to Multiple Classes

Extending to Multiple Classes

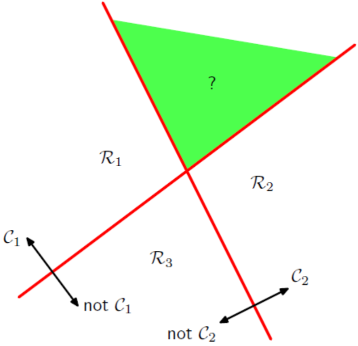
One Against Rest



K-1 Decision Surfaces

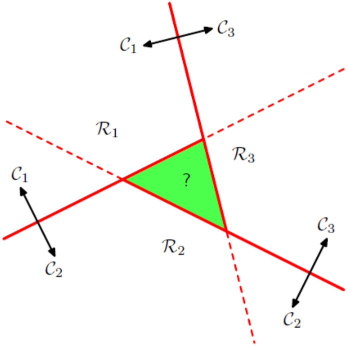
Extending to Multiple Classes

One Against Rest



K-1 Decision Surfaces

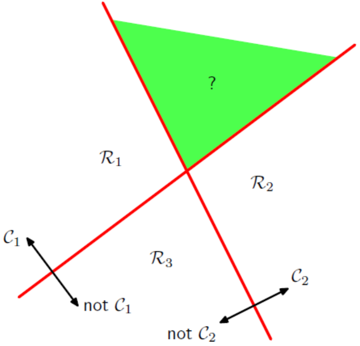
One Against One



$K(K-1)/2$ Decision Surfaces

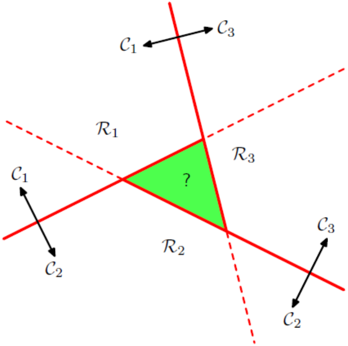
Extending to Multiple Classes

One Against Rest



K-1 Decision Surfaces

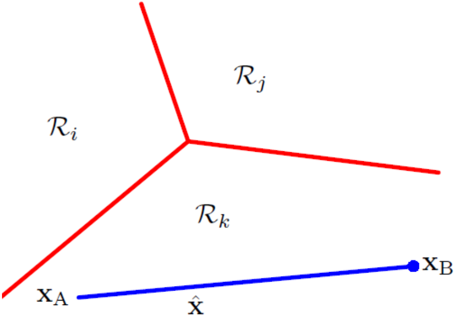
One Against One



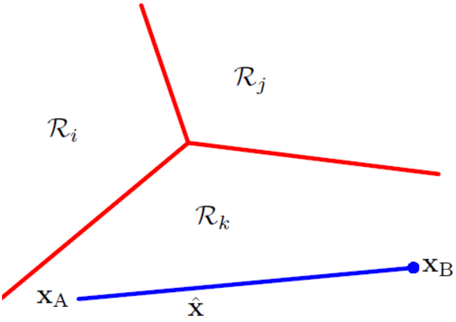
$K(K-1)/2$ Decision Surfaces

- Both the approaches lead to ambiguous regions!
- Decision regions should be singly connected and convex.

Multiclass Linear Discriminant



Multiclass Linear Discriminant



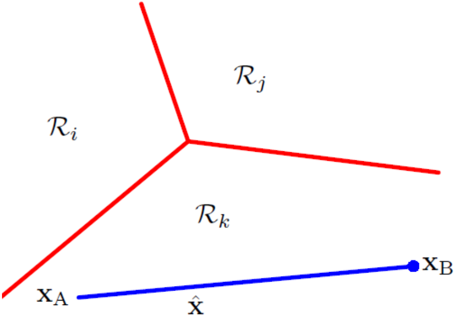
- K-Linear discriminant functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0} \quad k = 1, 2, \cdots, K$$

- Decision rule: Assign $\mathbf{x}$ to $\mathcal{C}_k$ if

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

Multiclass Linear Discriminant



- Decision boundary $\mathcal{C}_k$ & $\mathcal{C}_j$:

$$y_k(\mathbf{x}) = y_j(\mathbf{x})$$

$$(\mathbf{w}_k - \mathbf{w}_j)^T x + (w_{k0} - w_{j0}) = 0$$

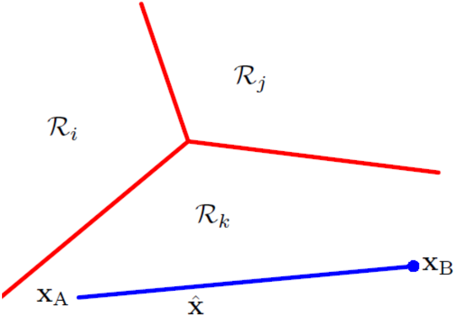
- K-Linear discriminant functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0} \quad k = 1, 2, \cdots, K$$

- Decision rule: Assign $\mathbf{x}$ to $\mathcal{C}_k$ if

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

Multiclass Linear Discriminant



- Decision boundary $\mathcal{C}_k$ & $\mathcal{C}_j$:

$$y_k(\mathbf{x}) = y_j(\mathbf{x})$$

$$(\mathbf{w}_k - \mathbf{w}_j)^{\mathsf{T}} x + (w_{k0} - w_{j0}) = 0$$

- Decision regions are convex

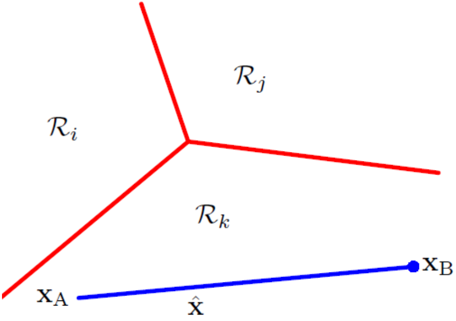
- K-Linear discriminant functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathsf{T}}\mathbf{x} + w_{k0} \quad k = 1, 2, \cdots, K$$

- Decision rule: Assign $\mathbf{x}$ to $\mathcal{C}_k$ if

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

Multiclass Linear Discriminant



- K-Linear discriminant functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^\mathsf{T}\mathbf{x} + w_{k0} \quad k = 1, 2, \cdots, K$$

- Decision rule: Assign $\mathbf{x}$ to $\mathcal{C}_k$ if

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

- Decision boundary $\mathcal{C}_k$ & $\mathcal{C}_j$:

$$y_k(\mathbf{x}) = y_j(\mathbf{x})$$

$$(\mathbf{w}_k - \mathbf{w}_j)^\mathsf{T} x + (w_{k0} - w_{j0}) = 0$$

- Decision regions are convex
- For two classes, we can either employ a single discriminant or two discriminants

Least Squares for Classification

Least Squares for Classification

- Let us apply Least Squares Regression approach to classification

Least Squares for Classification

- Let us apply Least Squares Regression approach to classification
- Training data: $\{(\mathbf{x}_1, \mathbf{t}_1), (\mathbf{x}_2, \mathbf{t}_2), \cdots (\mathbf{x}_N, \mathbf{t}_N)\}$

Least Squares for Classification

- Let us apply Least Squares Regression approach to classification
- Training data: $\{(\mathbf{x}_1, \mathbf{t}_1), (\mathbf{x}_2, \mathbf{t}_2), \cdots (\mathbf{x}_N, \mathbf{t}_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ denotes input observation vectors

Least Squares for Classification

- Let us apply Least Squares Regression approach to classification
- Training data: $\{(\mathbf{x}_1, \mathbf{t}_1), (\mathbf{x}_2, \mathbf{t}_2), \cdots (\mathbf{x}_N, \mathbf{t}_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ denotes input observation vectors
  - $\mathbf{t}_n$, with 1-of-K binary coding, denotes the class label (K - Classes)

Least Squares for Classification

- Let us apply Least Squares Regression approach to classification
- Training data: $\{(\mathbf{x}_1, \mathbf{t}_1), (\mathbf{x}_2, \mathbf{t}_2), \cdots (\mathbf{x}_N, \mathbf{t}_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ denotes input observation vectors
  - $\mathbf{t}_n$, with 1-of-K binary coding, denotes the class label (K - Classes)
- Each class $\mathcal{C}_k$ is described by its own linear discriminant

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathsf{T}}\mathbf{x} + w_{k0} \quad k = 1, 2, \cdots, K$$

Least Squares for Classification

- Let us apply Least Squares Regression approach to classification
- Training data: $\{(\mathbf{x}_1, \mathbf{t}_1), (\mathbf{x}_2, \mathbf{t}_2), \cdots (\mathbf{x}_N, \mathbf{t}_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ denotes input observation vectors
  - $\mathbf{t}_n$, with 1-of-K binary coding, denotes the class label (K - Classes)
- Each class $\mathcal{C}_k$ is described by its own linear discriminant

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathsf{T}}\mathbf{x} + w_{k0} \quad k = 1, 2, \cdots, K$$

- Regressed target for $\mathbf{x}_n$: $\hat{\mathbf{t}}_n = \mathbf{y}(\mathbf{x}_n) = \mathbf{W}^{\mathsf{T}}\mathbf{x}_n$ (Augmented)

Least Squares for Classification

- Let us apply Least Squares Regression approach to classification
- Training data: $\{(\mathbf{x}_1, \mathbf{t}_1), (\mathbf{x}_2, \mathbf{t}_2), \cdots (\mathbf{x}_N, \mathbf{t}_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ denotes input observation vectors
  - $\mathbf{t}_n$, with 1-of-K binary coding, denotes the class label (K - Classes)
- Each class $\mathcal{C}_k$ is described by its own linear discriminant

$$y_k(\mathbf{x}) = \mathbf{w}_k^\mathsf{T}\mathbf{x} + w_{k0} \quad k = 1, 2, \cdots, K$$

- Regressed target for $\mathbf{x}_n$: $\hat{\mathbf{t}}_n = \mathbf{y}(\mathbf{x}_n) = \mathbf{W}^\mathsf{T}\mathbf{x}_n$ (Augmented)
- Estimate $\mathbf{W}$ to minimize sum of squares error

$$J(\mathbf{W}) = \frac{1}{2}\,\mathrm{Tr}\left\{(\mathbf{X}\mathbf{W} - \mathbf{T})^\mathsf{T}(\mathbf{X}\mathbf{W} - \mathbf{T})\right\}$$

Least Squares for Classification

- Let us apply Least Squares Regression approach to classification
- Training data: $\{(\mathbf{x}_1, \mathbf{t}_1), (\mathbf{x}_2, \mathbf{t}_2), \cdots (\mathbf{x}_N, \mathbf{t}_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ denotes input observation vectors
  - $\mathbf{t}_n$, with 1-of-K binary coding, denotes the class label (K - Classes)
- Each class $\mathcal{C}_k$ is described by its own linear discriminant

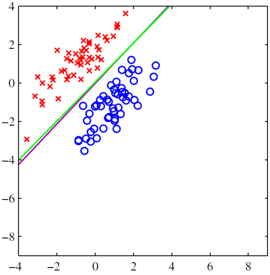
$$y_k(\mathbf{x}) = \mathbf{w}_k^\mathsf{T}\mathbf{x} + w_{k0} \quad k = 1, 2, \cdots, K$$

- Regressed target for $\mathbf{x}_n$: $\hat{\mathbf{t}}_n = \mathbf{y}(\mathbf{x}_n) = \mathbf{W}^\mathsf{T}\mathbf{x}_n$ (Augmented)
- Estimate $\mathbf{W}$ to minimize sum of squares error

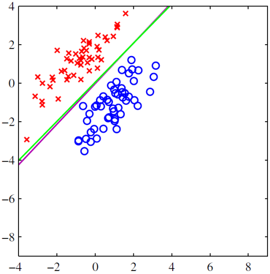
$$J(\mathbf{W}) = \frac{1}{2}\,\mathrm{Tr}\left\{(\mathbf{XW} - \mathbf{T})^\mathsf{T}(\mathbf{XW} - \mathbf{T})\right\}$$

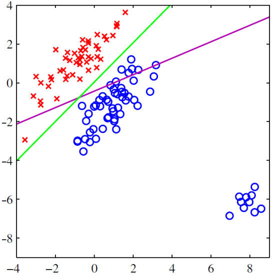
- Setting $\nabla_{\mathbf{W}} J(\mathbf{W}) = \mathbf{0}$, we get $\boxed{\mathbf{W}_* = \left(\mathbf{X}^\mathsf{T}\mathbf{X}\right)^{-1}\mathbf{X}^\mathsf{T}\mathbf{T}}$

Issues with LS

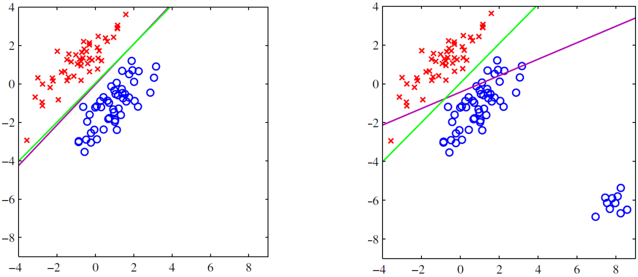


Issues with LS



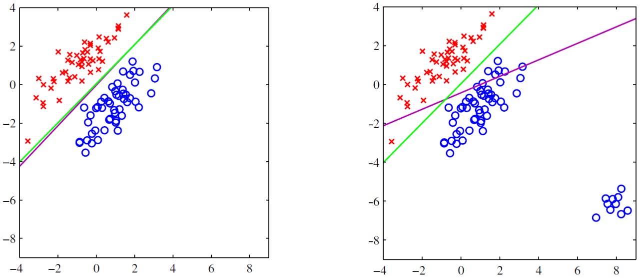


Issues with LS



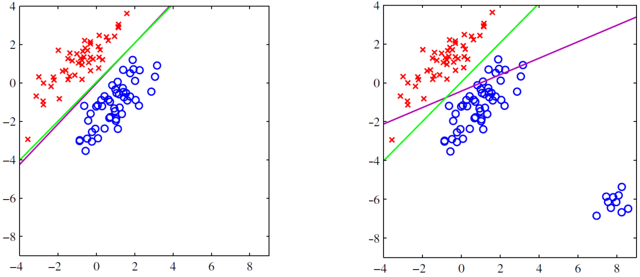
- Least squares solutions lack robustness to outliers

Issues with LS



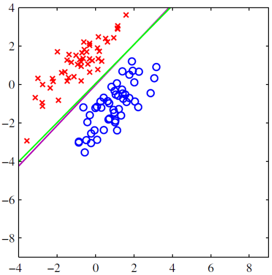
- Least squares solutions lack robustness to outliers
- Additional data-points resulted in significant change in boundary

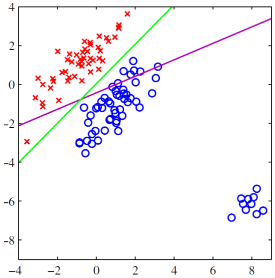
Issues with LS



- Least squares solutions lack robustness to outliers
- Additional data-points resulted in significant change in boundary
- Sum of squares error penalizes predictions that are "too correct"

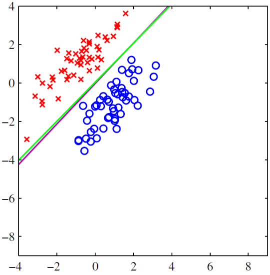
Issues with LS

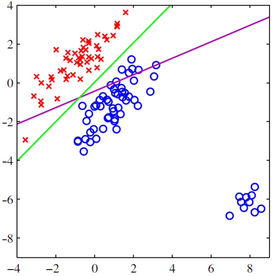




- Least squares solutions lack robustness to outliers
- Additional data-points resulted in significant change in boundary
- Sum of squares error penalizes predictions that are "too correct"
- Attempts to achieve "many-to-one" mapping through linearity!

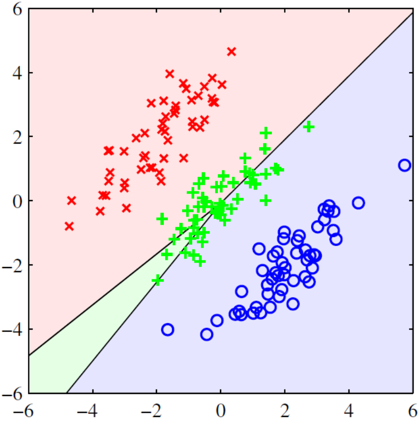
Issues with LS



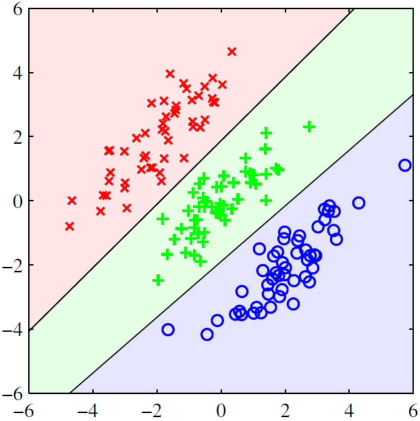


- Least squares solutions lack robustness to outliers
- Additional data-points resulted in significant change in boundary
- Sum of squares error penalizes predictions that are "too correct"
- Attempts to achieve "many-to-one" mapping through linearity!
- LS approach failed even for linearly separable classes

Issues with LS



Least Squares Classifier



Logistic Regression

Homework

Homework

- **Property of LS:** If every target in the training set satisfies some linear constraint

$$\mathbf{a}^{\mathsf{T}}\mathbf{t}_n + b = 0, \qquad \forall n$$

for some arbitrary constants $\mathbf{a}$ and b, then the model prediction for any value of $\mathbf{x}$ satisfies the same constraint.

$$\mathbf{a}^{\mathsf{T}}\mathbf{y}(\mathbf{x}) + b = 0$$

Homework

- **Property of LS:** If every target in the training set satisfies some linear constraint

$$\mathbf{a}^{\mathsf{T}}\mathbf{t}_n + b = 0, \qquad \forall n$$

for some arbitrary constants $\mathbf{a}$ and b, then the model prediction for any value of $\mathbf{x}$ satisfies the same constraint.

$$\mathbf{a}^{\mathsf{T}}\mathbf{y}(\mathbf{x}) + b = 0$$

- If we use 1-of-K coding for targets, then the predictions sum to 1.

$$\sum_{k=1}^{K} y_k(\mathbf{x}) = 1$$

Homework

- **Property of LS:** If every target in the training set satisfies some linear constraint

$$\mathbf{a}^{\mathsf{T}}\mathbf{t}_n + b = 0, \qquad \forall n$$

for some arbitrary constants $\mathbf{a}$ and b, then the model prediction for any value of $\mathbf{x}$ satisfies the same constraint.

$$\mathbf{a}^{\mathsf{T}}\mathbf{y}(\mathbf{x}) + b = 0$$

- If we use 1-of-K coding for targets, then the predictions sum to 1.

$$\sum_{k=1}^{K} y_k(\mathbf{x}) = 1$$

- However, $y_k(\mathbf{x})$ cannot be interpreted as posterior probability. They can be negative!

Linear Discriminant Analysis

Linear Discriminant Analysis

- Dimensionality reduction interpretation to LS classifier (2 classes)

Linear Discriminant Analysis

- Dimensionality reduction interpretation to LS classifier (2 classes)
  - Project the $\mathbf{x} \in \mathbb{R}^D$ on to real line (1-D): $y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$

Linear Discriminant Analysis

- Dimensionality reduction interpretation to LS classifier (2 classes)
  - Project the $\mathbf{x} \in \mathbb{R}^D$ on to real line (1-D): $y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$
  - Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) > -w_0$ and to $\mathcal{C}_2$ otherwise

Linear Discriminant Analysis

- Dimensionality reduction interpretation to LS classifier (2 classes)
  - Project the $\mathbf{x} \in \mathbb{R}^D$ on to real line (1-D): $y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$
  - Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) > -w_0$ and to $\mathcal{C}_2$ otherwise
  - Projecting to 1-D, in general, leads to loss of information

Linear Discriminant Analysis

- Dimensionality reduction interpretation to LS classifier (2 classes)
  - Project the $\mathbf{x} \in \mathbb{R}^D$ on to real line (1-D): $y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$
  - Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) > -w_0$ and to $\mathcal{C}_2$ otherwise
  - Projecting to 1-D, in general, leads to loss of information
  - Adjust $\mathbf{w}$ to select a projection that maximizes the class separation.

Linear Discriminant Analysis

- Dimensionality reduction interpretation to LS classifier (2 classes)
  - Project the $\mathbf{x} \in \mathbb{R}^D$ on to real line (1-D): $y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$
  - Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) > -w_0$ and to $\mathcal{C}_2$ otherwise
  - Projecting to 1-D, in general, leads to loss of information
  - Adjust $\mathbf{w}$ to select a projection that maximizes the class separation.
- Consider N_1 points from $\mathcal{C}_1$ and N_2 points from $\mathcal{C}_2$ in D-dim space

Linear Discriminant Analysis

- Dimensionality reduction interpretation to LS classifier (2 classes)
  - Project the $\mathbf{x} \in \mathbb{R}^D$ on to real line (1-D): $y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$
  - Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) > -w_0$ and to $\mathcal{C}_2$ otherwise
  - Projecting to 1-D, in general, leads to loss of information
  - Adjust $\mathbf{w}$ to select a projection that maximizes the class separation.
- Consider N_1 points from $\mathcal{C}_1$ and N_2 points from $\mathcal{C}_2$ in D-dim space
  - Mean vectors in original space: $\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$

Linear Discriminant Analysis

- Dimensionality reduction interpretation to LS classifier (2 classes)
  - Project the $\mathbf{x} \in \mathbb{R}^D$ on to real line (1-D): $y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$
  - Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) > -w_0$ and to $\mathcal{C}_2$ otherwise
  - Projecting to 1-D, in general, leads to loss of information
  - Adjust $\mathbf{w}$ to select a projection that maximizes the class separation.
- Consider N_1 points from $\mathcal{C}_1$ and N_2 points from $\mathcal{C}_2$ in D-dim space
  - Mean vectors in original space: $\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$
  - Let the projected means be $\mu_1 = \mathbf{w}^\mathsf{T}\mathbf{m}_1$ and $\mu_2 = \mathbf{w}^\mathsf{T}\mathbf{m}_2$

Linear Discriminant Analysis

- Dimensionality reduction interpretation to LS classifier (2 classes)
  - Project the $\mathbf{x} \in \mathbb{R}^D$ on to real line (1-D): $y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$
  - Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) > -w_0$ and to $\mathcal{C}_2$ otherwise
  - Projecting to 1-D, in general, leads to loss of information
  - Adjust $\mathbf{w}$ to select a projection that maximizes the class separation.
- Consider N_1 points from $\mathcal{C}_1$ and N_2 points from $\mathcal{C}_2$ in D-dim space
  - Mean vectors in original space: $\mathbf{m}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}\mathbf{x}_n \quad \mathbf{m}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}\mathbf{x}_n$
  - Let the projected means be $\mu_1 = \mathbf{w}^\mathsf{T}\mathbf{m}_1$ and $\mu_2 = \mathbf{w}^\mathsf{T}\mathbf{m}_2$
  - Choose $\mathbf{w}$ to maximize the separation between projected means

$$\mu_2 - \mu_1 = \mathbf{w}^\mathsf{T}(\mathbf{m}_2 - \mathbf{m}_1) \quad \textbf{st.} \quad \mathbf{w}^\mathsf{T}\mathbf{w} = 1$$

Linear Discriminant Analysis

- Dimensionality reduction interpretation to LS classifier (2 classes)
  - Project the $\mathbf{x} \in \mathbb{R}^D$ on to real line (1-D): $y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$
  - Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) > -w_0$ and to $\mathcal{C}_2$ otherwise
  - Projecting to 1-D, in general, leads to loss of information
  - Adjust $\mathbf{w}$ to select a projection that maximizes the class separation.
- Consider N_1 points from $\mathcal{C}_1$ and N_2 points from $\mathcal{C}_2$ in D-dim space
  - Mean vectors in original space: $\mathbf{m}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}\mathbf{x}_n \quad \mathbf{m}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}\mathbf{x}_n$
  - Let the projected means be $\mu_1 = \mathbf{w}^\mathsf{T}\mathbf{m}_1$ and $\mu_2 = \mathbf{w}^\mathsf{T}\mathbf{m}_2$
  - Choose $\mathbf{w}$ to maximize the separation between projected means

$$\mu_2 - \mu_1 = \mathbf{w}^\mathsf{T}(\mathbf{m}_2 - \mathbf{m}_1) \quad \textbf{st.} \quad \mathbf{w}^\mathsf{T}\mathbf{w} = 1$$

  - The direction for $\mathbf{w}$ can be shown to be $\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)$

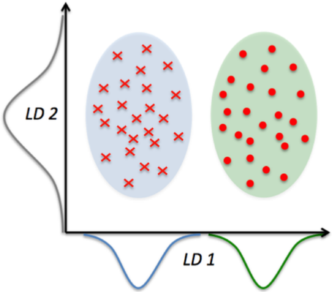
Linear Discriminant Analysis

- Dimensionality reduction interpretation to LS classifier (2 classes)
  - Project the $\mathbf{x} \in \mathbb{R}^D$ on to real line (1-D): $y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$
  - Assign $\mathbf{x}$ to $\mathcal{C}_1$ if $y(\mathbf{x}) > -w_0$ and to $\mathcal{C}_2$ otherwise
  - Projecting to 1-D, in general, leads to loss of information
  - Adjust $\mathbf{w}$ to select a projection that maximizes the class separation.
- Consider N_1 points from $\mathcal{C}_1$ and N_2 points from $\mathcal{C}_2$ in D-dim space
  - Mean vectors in original space: $\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$
  - Let the projected means be $\mu_1 = \mathbf{w}^\mathsf{T}\mathbf{m}_1$ and $\mu_2 = \mathbf{w}^\mathsf{T}\mathbf{m}_2$
  - Choose $\mathbf{w}$ to maximize the separation between projected means

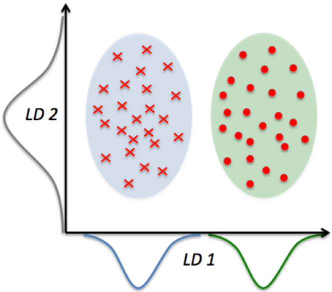
$$\mu_2 - \mu_1 = \mathbf{w}^\mathsf{T}(\mathbf{m}_2 - \mathbf{m}_1) \quad \textbf{st.} \quad \mathbf{w}^\mathsf{T}\mathbf{w} = 1$$

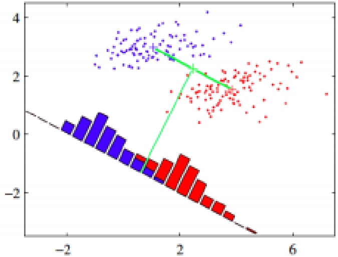
  - The direction for $\mathbf{w}$ can be shown to be $\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)$
- This approach is not optimal for nondiagonal covariances

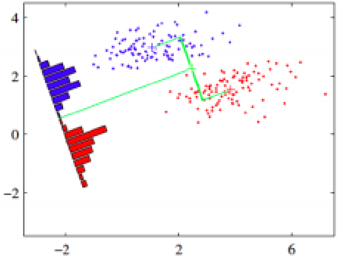
LDA - Illustration



LDA - Illustration







Fisher Discriminant Analysis

- For nondiagonal covariances, spread of data should also be considered
- Project the data in a direction that
  - maximizes seperation between the means of the projected classes
  - minimizes the variance within each projected class
- The objective function is given by

$$J(\mathbf{w}) = \frac{(\mu_2 - \mu_1)^2}{\sigma_1^2 + \sigma_2^2}$$

Fisher Discriminant Analysis

- For nondiagonal covariances, spread of data should also be considered
- Project the data in a direction that
  - maximizes seperation between the means of the projected classes
  - minimizes the variance within each projected class
- The objective function is given by

$$J(\mathbf{w}) = \frac{(\mu_2 - \mu_1)^2}{\sigma_1^2 + \sigma_2^2} = \frac{\mathbf{w}^\mathsf{T}\mathbf{S}_B\mathbf{w}}{\mathbf{w}^\mathsf{T}\mathbf{S}_W\mathbf{w}}$$

  - $\mathbf{S}_B$ is *between-class* covariance: $\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\mathsf{T}$
  - $\mathbf{S}_W$ is *within-class* covariance: $\mathbf{S}_W = \sum_{k=1}^{2} \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^\mathsf{T}$
- Optimal direction of $\mathbf{w}$ can be obtained by maximizing $J(\mathbf{w})$

Solution to Fisher Criterion

Solution to Fisher Criterion

- Equating $\nabla J(\mathbf{w}) = \mathbf{0}$, we obtain

$$(\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

Solution to Fisher Criterion

- Equating $\nabla J(\mathbf{w}) = \mathbf{0}$, we obtain

$$(\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

  - $\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w}$ and $\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w}$ contribute to only scalar multiples

Solution to Fisher Criterion

- Equating $\nabla J(\mathbf{w}) = \mathbf{0}$, we obtain

$$(\mathbf{w}^\mathsf{T}\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^\mathsf{T}\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

  - $\mathbf{w}^\mathsf{T}\mathbf{S}_B\mathbf{w}$ and $\mathbf{w}^\mathsf{T}\mathbf{S}_W\mathbf{w}$ contribute to only scalar multiples
  - The direction of $\mathbf{w}$ is given by $\mathbf{S}_W\mathbf{w} \propto \mathbf{S}_B\mathbf{w}$

Solution to Fisher Criterion

- Equating $\nabla J(\mathbf{w}) = \mathbf{0}$, we obtain

$$(\mathbf{w}^\mathsf{T}\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^\mathsf{T}\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

  - $\mathbf{w}^\mathsf{T}\mathbf{S}_B\mathbf{w}$ and $\mathbf{w}^\mathsf{T}\mathbf{S}_W\mathbf{w}$ contribute to only scalar multiples
  - The direction of $\mathbf{w}$ is given by $\mathbf{S}_W\mathbf{w} \propto \mathbf{S}_B\mathbf{w}$
  - $\mathbf{S}_B\mathbf{w}$ is always in the direction of $\mathbf{m}_2 - \mathbf{m}_1$

Solution to Fisher Criterion

- Equating $\nabla J(\mathbf{w}) = \mathbf{0}$, we obtain

$$(\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

  - $\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w}$ and $\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w}$ contribute to only scalar multiples
  - The direction of $\mathbf{w}$ is given by $\mathbf{S}_W\mathbf{w} \propto \mathbf{S}_B\mathbf{w}$
  - $\mathbf{S}_B\mathbf{w}$ is always in the direction of $\mathbf{m}_2 - \mathbf{m}_1$
  - The direction of the fisher discriminant: $\boxed{\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)}$

Solution to Fisher Criterion

- Equating $\nabla J(\mathbf{w}) = \mathbf{0}$, we obtain

$$(\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

  - $\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w}$ and $\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w}$ contribute to only scalar multiples
  - The direction of $\mathbf{w}$ is given by $\mathbf{S}_W\mathbf{w} \propto \mathbf{S}_B\mathbf{w}$
  - $\mathbf{S}_B\mathbf{w}$ is always in the direction of $\mathbf{m}_2 - \mathbf{m}_1$
  - The direction of the fisher discriminant: $\boxed{\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)}$
- Multiplication by $\mathbf{S}_W^{-1}$ can be interpreted as data whitening.

Solution to Fisher Criterion

- Equating $\nabla J(\mathbf{w}) = \mathbf{0}$, we obtain

$$(\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

  - $\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w}$ and $\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w}$ contribute to only scalar multiples
  - The direction of $\mathbf{w}$ is given by $\mathbf{S}_W\mathbf{w} \propto \mathbf{S}_B\mathbf{w}$
  - $\mathbf{S}_B\mathbf{w}$ is always in the direction of $\mathbf{m}_2 - \mathbf{m}_1$
  - The direction of the fisher discriminant: $\boxed{\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)}$
- Multiplication by $\mathbf{S}_W^{-1}$ can be interpreted as data whitening.
- FDA also is sensitive to outliers even if they are "too correct"

Solution to Fisher Criterion

- Equating $\nabla J(\mathbf{w}) = \mathbf{0}$, we obtain

$$(\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

  - $\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w}$ and $\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w}$ contribute to only scalar multiples
  - The direction of $\mathbf{w}$ is given by $\mathbf{S}_W\mathbf{w} \propto \mathbf{S}_B\mathbf{w}$
  - $\mathbf{S}_B\mathbf{w}$ is always in the direction of $\mathbf{m}_2 - \mathbf{m}_1$
  - The direction of the fisher discriminant: $\boxed{\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)}$
- Multiplication by $\mathbf{S}_W^{-1}$ can be interpreted as data whitening.
- FDA also is sensitive to outliers even if they are "too correct"
- FD is strictly not a "discriminant" in true sense.

Solution to Fisher Criterion

- Equating $\nabla J(\mathbf{w}) = \mathbf{0}$, we obtain

$$(\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

  - $\mathbf{w}^{\mathsf{T}}\mathbf{S}_B\mathbf{w}$ and $\mathbf{w}^{\mathsf{T}}\mathbf{S}_W\mathbf{w}$ contribute to only scalar multiples
  - The direction of $\mathbf{w}$ is given by $\mathbf{S}_W\mathbf{w} \propto \mathbf{S}_B\mathbf{w}$
  - $\mathbf{S}_B\mathbf{w}$ is always in the direction of $\mathbf{m}_2 - \mathbf{m}_1$
  - The direction of the fisher discriminant: $\boxed{\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)}$
- Multiplication by $\mathbf{S}_W^{-1}$ can be interpreted as data whitening.
- FDA also is sensitive to outliers even if they are "too correct"
- FD is strictly not a "discriminant" in true sense.
  - It just projects binary class data to 1-D

Solution to Fisher Criterion

- Equating $\nabla J(\mathbf{w}) = \mathbf{0}$, we obtain

$$(\mathbf{w}^\mathsf{T}\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^\mathsf{T}\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

  - $\mathbf{w}^\mathsf{T}\mathbf{S}_B\mathbf{w}$ and $\mathbf{w}^\mathsf{T}\mathbf{S}_W\mathbf{w}$ contribute to only scalar multiples
  - The direction of $\mathbf{w}$ is given by $\mathbf{S}_W\mathbf{w} \propto \mathbf{S}_B\mathbf{w}$
  - $\mathbf{S}_B\mathbf{w}$ is always in the direction of $\mathbf{m}_2 - \mathbf{m}_1$
  - The direction of the fisher discriminant: $\boxed{\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)}$
- Multiplication by $\mathbf{S}_W^{-1}$ can be interpreted as data whitening.
- FDA also is sensitive to outliers even if they are "too correct"
- FD is strictly not a "discriminant" in true sense.
  - It just projects binary class data to 1-D
  - We need to arrive at a threshold w_0 to perform classification

Homework: Relation to Least Squares

- In LS approach, linear discriminant is determined to make model predictions as close as possible to target values
- In FDA, the discriminant is derived to achieve maximum class separation in the projected space
- If we take targets for $\mathcal{C}_1$ and $\mathcal{C}_2$ as $\frac{N}{N_1}$ and $-\frac{N}{N_2}$, respectively, where $N = N_1 + N_2$, show that LS approach yields the same solution as FD.

Extending FDA to Multiple Classes

Extending FDA to Multiple Classes

- Assumption: Data dimensionality $D > K$ Number of classes

Extending FDA to Multiple Classes

- Assumption: Data dimensionality $D > K$ Number of classes
- For multi-class case, it is not enough to project data to 1-D

Extending FDA to Multiple Classes

- Assumption: Data dimensionality $D > K$ Number of classes
- For multi-class case, it is not enough to project data to 1-D
- Project the data to a lower dimensional space $D' < D$

Extending FDA to Multiple Classes

- Assumption: Data dimensionality $D > K$ Number of classes
- For multi-class case, it is not enough to project data to 1-D
- Project the data to a lower dimensional space $D' < D$
  - Let $\mathbf{W} : \mathbb{R}^{D \times D'}$ be the linear map to achieve dimensionality reduction

Extending FDA to Multiple Classes

- Assumption: Data dimensionality $D > K$ Number of classes
- For multi-class case, it is not enough to project data to 1-D
- Project the data to a lower dimensional space $D' < D$
  - Let $\mathbf{W} : \mathbb{R}^{D \times D'}$ be the linear map to achieve dimensionality reduction
  - The lower dimensional representation of $\mathbf{x}_n \in \mathbb{R}^D$ is $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n \in \mathbb{R}^{D'}$

Extending FDA to Multiple Classes

- Assumption: Data dimensionality $D > K$ Number of classes
- For multi-class case, it is not enough to project data to 1-D
- Project the data to a lower dimensional space $D' < D$
  - Let $\mathbf{W} : \mathbb{R}^{D \times D'}$ be the linear map to achieve dimensionality reduction
  - The lower dimensional representation of $\mathbf{x}_n \in \mathbb{R}^D$ is $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n \in \mathbb{R}^{D'}$
  - Maximize between-class spread and minimize within-class spread in the projected space

Extending FDA to Multiple Classes

- Assumption: Data dimensionality $D > K$ Number of classes
- For multi-class case, it is not enough to project data to 1-D
- Project the data to a lower dimensional space $D' < D$
  - Let $\mathbf{W} : \mathbb{R}^{D \times D'}$ be the linear map to achieve dimensionality reduction
  - The lower dimensional representation of $\mathbf{x}_n \in \mathbb{R}^D$ is $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n \in \mathbb{R}^{D'}$
  - Maximize between-class spread and minimize within-class spread in the projected space
  - Let $\mathbf{S}_W$ and $\mathbf{S}_B$ denote within-class and between-class covariance in original D-dim space

Extending FDA to Multiple Classes

- Assumption: Data dimensionality $D > K$ Number of classes
- For multi-class case, it is not enough to project data to 1-D
- Project the data to a lower dimensional space $D' < D$
  - Let $\mathbf{W} : \mathbb{R}^{D \times D'}$ be the linear map to achieve dimensionality reduction
  - The lower dimensional representation of $\mathbf{x}_n \in \mathbb{R}^D$ is $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n \in \mathbb{R}^{D'}$
  - Maximize between-class spread and minimize within-class spread in the projected space
  - Let $\mathbf{S}_W$ and $\mathbf{S}_B$ denote within-class and between-class covariance in original D-dim space
  - Let $\mathbf{\Sigma}_W$ and $\mathbf{\Sigma}_B$ be their counterparts in projected space

Extending FDA to Multiple Classes

- Assumption: Data dimensionality $D > K$ Number of classes
- For multi-class case, it is not enough to project data to 1-D
- Project the data to a lower dimensional space $D' < D$
  - Let $\mathbf{W} : \mathbb{R}^{D \times D'}$ be the linear map to achieve dimensionality reduction
  - The lower dimensional representation of $\mathbf{x}_n \in \mathbb{R}^D$ is $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n \in \mathbb{R}^{D'}$
  - Maximize between-class spread and minimize within-class spread in the projected space
  - Let $\mathbf{S}_W$ and $\mathbf{S}_B$ denote within-class and between-class covariance in original D-dim space
  - Let $\mathbf{\Sigma}_W$ and $\mathbf{\Sigma}_B$ be their counterparts in projected space
  - Estimate $\mathbf{W}$ to maximize $J(\mathbf{W}) = \mathrm{Tr}\left\{\mathbf{\Sigma}_W^{-1}\mathbf{\Sigma}_B\right\}$

Extending FDA to Multiple Classes

- Assumption: Data dimensionality $D > K$ Number of classes
- For multi-class case, it is not enough to project data to 1-D
- Project the data to a lower dimensional space $D' < D$
  - Let $\mathbf{W} : \mathbb{R}^{D \times D'}$ be the linear map to achieve dimensionality reduction
  - The lower dimensional representation of $\mathbf{x}_n \in \mathbb{R}^D$ is $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n \in \mathbb{R}^{D'}$
  - Maximize between-class spread and minimize within-class spread in the projected space
  - Let $\mathbf{S}_W$ and $\mathbf{S}_B$ denote within-class and between-class covariance in original D-dim space
  - Let $\mathbf{\Sigma}_W$ and $\mathbf{\Sigma}_B$ be their counterparts in projected space
  - Estimate $\mathbf{W}$ to maximize $J(\mathbf{W}) = \text{Tr}\left\{\mathbf{\Sigma}_W^{-1}\mathbf{\Sigma}_B\right\}$
- The columns of $\mathbf{W}$ are given by eigenvectors corresponding to the D' largest eigenvalues of $\mathbf{S}_W^{-1}\mathbf{S}_B$

Statistics in Original D-dim Space

Statistics in Original D-dim Space

- Class specific statistics in the original D-dim space

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$$

Statistics in Original D-dim Space

- Class specific statistics in the original D-dim space

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n \qquad \mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^\mathsf{T}$$

Statistics in Original D-dim Space

- Class specific statistics in the original D-dim space

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n \qquad \mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^{\mathsf{T}}$$

- Define *within-class* covariance as $\mathbf{S}_W = \sum_{k=1}^{K} \mathbf{S}_k$

Statistics in Original D-dim Space

- Class specific statistics in the original D-dim space

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n \qquad \mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^{\mathsf{T}}$$

- Define *within-class* covariance as $\mathbf{S}_W = \sum_{k=1}^{K} \mathbf{S}_k$
- Overall data statistics (without considering class labels)

$$\mathbf{m}_T = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n$$

Statistics in Original D-dim Space

- Class specific statistics in the original D-dim space

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n \qquad \mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^\mathsf{T}$$

- Define *within-class* covariance as $\mathbf{S}_W = \sum_{k=1}^{K} \mathbf{S}_k$
- Overall data statistics (without considering class labels)

$$\mathbf{m}_T = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n = \frac{1}{N} \sum_{k=1}^{K} N_k \mathbf{m}_k$$

Statistics in Original D-dim Space

- Class specific statistics in the original D-dim space

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n \qquad \mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^{\mathsf{T}}$$

- Define *within-class* covariance as $\mathbf{S}_W = \sum_{k=1}^{K} \mathbf{S}_k$
- Overall data statistics (without considering class labels)

$$\mathbf{m}_T = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n = \frac{1}{N} \sum_{k=1}^{K} N_k \mathbf{m}_k \qquad \mathbf{S}_T = \sum_{n=1}^{N} (\mathbf{x}_n - \mathbf{m}_T)(\mathbf{x}_n - \mathbf{m}_T)^{\mathsf{T}}$$

Statistics in Original D-dim Space

- Class specific statistics in the original D-dim space

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n \qquad \mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^{\mathsf{T}}$$

- Define *within-class* covariance as $\mathbf{S}_W = \sum_{k=1}^{K} \mathbf{S}_k$
- Overall data statistics (without considering class labels)

$$\mathbf{m}_T = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n = \frac{1}{N} \sum_{k=1}^{K} N_k \mathbf{m}_k \qquad \mathbf{S}_T = \sum_{n=1}^{N} (\mathbf{x}_n - \mathbf{m}_T)(\mathbf{x}_n - \mathbf{m}_T)^{\mathsf{T}}$$

- Assumption: Define *between-class* covariance using $\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B$

Statistics in Original D-dim Space

- Class specific statistics in the original D-dim space

$$\mathbf{m}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{x}_n \qquad \mathbf{S}_k = \sum_{n\in\mathcal{C}_k}(\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^{\mathsf{T}}$$

- Define *within-class* covariance as $\mathbf{S}_W = \sum_{k=1}^{K}\mathbf{S}_k$
- Overall data statistics (without considering class labels)

$$\mathbf{m}_T = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n = \frac{1}{N}\sum_{k=1}^{K}N_k\mathbf{m}_k \qquad \mathbf{S}_T = \sum_{n=1}^{N}(\mathbf{x}_n - \mathbf{m}_T)(\mathbf{x}_n - \mathbf{m}_T)^{\mathsf{T}}$$

- Assumption: Define *between-class* covariance using $\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B$

$$\mathbf{S}_B = \sum_{k=1}^{K}N_k(\mathbf{m}_k - \mathbf{m}_T)(\mathbf{m}_k - \mathbf{m}_T)^{\mathsf{T}}$$

Statistics in Projected D'-dim Space

Statistics in Projected D'-dim Space

- The projected data-points in $\mathbb{R}^{D'}$ are given by $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n$

Statistics in Projected D'-dim Space

- The projected data-points in $\mathbb{R}^{D'}$ are given by $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n$
- The class-specific statistics in the projected space are

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n \in \mathcal{C}_k} \mathbf{y}_n$$

Statistics in Projected D'-dim Space

- The projected data-points in $\mathbb{R}^{D'}$ are given by $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n$
- The class-specific statistics in the projected space are

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n \in \mathcal{C}_k} \mathbf{y}_n \qquad \boldsymbol{\Sigma}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^\mathsf{T}$$

Statistics in Projected D'-dim Space

- The projected data-points in $\mathbb{R}^{D'}$ are given by $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n$
- The class-specific statistics in the projected space are

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{y}_n \qquad \boldsymbol{\Sigma}_k = \sum_{n\in\mathcal{C}_k}(\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^\mathsf{T}$$

- Relation between the statistics of original and projected space

Statistics in Projected D'-dim Space

- The projected data-points in $\mathbb{R}^{D'}$ are given by $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n$
- The class-specific statistics in the projected space are

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{y}_n \qquad \boldsymbol{\Sigma}_k = \sum_{n\in\mathcal{C}_k}(\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^\mathsf{T}$$

- Relation between the statistics of original and projected space

$$\boldsymbol{\mu}_k = \mathbf{W}^\mathsf{T}\mathbf{m}_k$$

Statistics in Projected D'-dim Space

- The projected data-points in $\mathbb{R}^{D'}$ are given by $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n$
- The class-specific statistics in the projected space are

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{y}_n \qquad \boldsymbol{\Sigma}_k = \sum_{n\in\mathcal{C}_k}(\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^\mathsf{T}$$

- Relation between the statistics of original and projected space

$$\boldsymbol{\mu}_k = \mathbf{W}^\mathsf{T}\mathbf{m}_k \quad \boldsymbol{\Sigma}_k = \mathbf{W}^\mathsf{T}\mathbf{S}_k\mathbf{W}$$

Statistics in Projected D'-dim Space

- The projected data-points in $\mathbb{R}^{D'}$ are given by $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n$
- The class-specific statistics in the projected space are

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{y}_n \qquad \boldsymbol{\Sigma}_k = \sum_{n\in\mathcal{C}_k}(\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^\mathsf{T}$$

- Relation between the statistics of original and projected space

$$\boldsymbol{\mu}_k = \mathbf{W}^\mathsf{T}\mathbf{m}_k \quad \boldsymbol{\Sigma}_k = \mathbf{W}^\mathsf{T}\mathbf{S}_k\mathbf{W} \quad \boldsymbol{\Sigma}_W = \mathbf{W}^\mathsf{T}\mathbf{S}_W\mathbf{W} \quad \boldsymbol{\Sigma}_B = \mathbf{W}^\mathsf{T}\mathbf{S}_B\mathbf{W}$$

Statistics in Projected D'-dim Space

- The projected data-points in $\mathbb{R}^{D'}$ are given by $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n$
- The class-specific statistics in the projected space are

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{y}_n \qquad \boldsymbol{\Sigma}_k = \sum_{n\in\mathcal{C}_k}(\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^\mathsf{T}$$

- Relation between the statistics of original and projected space

$$\boldsymbol{\mu}_k = \mathbf{W}^\mathsf{T}\mathbf{m}_k \quad \boldsymbol{\Sigma}_k = \mathbf{W}^\mathsf{T}\mathbf{S}_k\mathbf{W} \quad \boldsymbol{\Sigma}_W = \mathbf{W}^\mathsf{T}\mathbf{S}_W\mathbf{W} \quad \boldsymbol{\Sigma}_B = \mathbf{W}^\mathsf{T}\mathbf{S}_B\mathbf{W}$$

- Estimate $\mathbf{W}$ to maximize $J(\mathbf{W}) = \mathrm{Tr}\left\{\left(\mathbf{W}^\mathsf{T}\mathbf{S}_W\mathbf{W}\right)^{-1}\left(\mathbf{W}^\mathsf{T}\mathbf{S}_B\mathbf{W}\right)\right\}$

Statistics in Projected D'-dim Space

- The projected data-points in $\mathbb{R}^{D'}$ are given by $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n$
- The class-specific statistics in the projected space are

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{y}_n \qquad \boldsymbol{\Sigma}_k = \sum_{n\in\mathcal{C}_k}(\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^\mathsf{T}$$

- Relation between the statistics of original and projected space

$$\boldsymbol{\mu}_k = \mathbf{W}^\mathsf{T}\mathbf{m}_k \quad \boldsymbol{\Sigma}_k = \mathbf{W}^\mathsf{T}\mathbf{S}_k\mathbf{W} \quad \boldsymbol{\Sigma}_W = \mathbf{W}^\mathsf{T}\mathbf{S}_W\mathbf{W} \quad \boldsymbol{\Sigma}_B = \mathbf{W}^\mathsf{T}\mathbf{S}_B\mathbf{W}$$

- Estimate $\mathbf{W}$ to maximize $J(\mathbf{W}) = \mathrm{Tr}\left\{\left(\mathbf{W}^\mathsf{T}\mathbf{S}_W\mathbf{W}\right)^{-1}\left(\mathbf{W}^\mathsf{T}\mathbf{S}_B\mathbf{W}\right)\right\}$
  - Solution leads to the famous eigenvalue problem: $\mathbf{S}_W\mathbf{W} \propto \mathbf{S}_B\mathbf{W}$

Statistics in Projected D'-dim Space

- The projected data-points in $\mathbb{R}^{D'}$ are given by $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n$
- The class-specific statistics in the projected space are

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{y}_n \qquad \boldsymbol{\Sigma}_k = \sum_{n\in\mathcal{C}_k}(\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^\mathsf{T}$$

- Relation between the statistics of original and projected space

$$\boldsymbol{\mu}_k = \mathbf{W}^\mathsf{T}\mathbf{m}_k \quad \boldsymbol{\Sigma}_k = \mathbf{W}^\mathsf{T}\mathbf{S}_k\mathbf{W} \quad \boldsymbol{\Sigma}_W = \mathbf{W}^\mathsf{T}\mathbf{S}_W\mathbf{W} \quad \boldsymbol{\Sigma}_B = \mathbf{W}^\mathsf{T}\mathbf{S}_B\mathbf{W}$$

- Estimate $\mathbf{W}$ to maximize $J(\mathbf{W}) = \mathrm{Tr}\left\{\left(\mathbf{W}^\mathsf{T}\mathbf{S}_W\mathbf{W}\right)^{-1}\left(\mathbf{W}^\mathsf{T}\mathbf{S}_B\mathbf{W}\right)\right\}$
  - Solution leads to the famous eigenvalue problem: $\mathbf{S}_W\mathbf{W} \propto \mathbf{S}_B\mathbf{W}$
  - Eigenvectors corresponding to D' largest eigenvalues forms $\mathbf{W}$

Statistics in Projected D'-dim Space

- The projected data-points in $\mathbb{R}^{D'}$ are given by $\mathbf{y}_n = \mathbf{W}^\mathsf{T}\mathbf{x}_n$
- The class-specific statistics in the projected space are

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{y}_n \qquad \boldsymbol{\Sigma}_k = \sum_{n\in\mathcal{C}_k}(\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^\mathsf{T}$$

- Relation between the statistics of original and projected space

$$\boldsymbol{\mu}_k = \mathbf{W}^\mathsf{T}\mathbf{m}_k \quad \boldsymbol{\Sigma}_k = \mathbf{W}^\mathsf{T}\mathbf{S}_k\mathbf{W} \quad \boldsymbol{\Sigma}_W = \mathbf{W}^\mathsf{T}\mathbf{S}_W\mathbf{W} \quad \boldsymbol{\Sigma}_B = \mathbf{W}^\mathsf{T}\mathbf{S}_B\mathbf{W}$$

- Estimate $\mathbf{W}$ to maximize $J(\mathbf{W}) = \mathrm{Tr}\left\{\left(\mathbf{W}^\mathsf{T}\mathbf{S}_W\mathbf{W}\right)^{-1}\left(\mathbf{W}^\mathsf{T}\mathbf{S}_B\mathbf{W}\right)\right\}$
    - Solution leads to the famous eigenvalue problem: $\mathbf{S}_W\mathbf{W} \propto \mathbf{S}_B\mathbf{W}$
    - Eigenvectors corresponding to D' largest eigenvalues forms $\mathbf{W}$
- D' is bounded by rank of $\mathbf{S}_B$, and can be at most $K-1$

The Perceptron

The Perceptron

- Issues with least squares and linear discriminants

The Perceptron

- Issues with least squares and linear discriminants
  - Linear relation cannot achieve *many-to-one* mapping

The Perceptron

- Issues with least squares and linear discriminants
  - Linear relation cannot achieve *many-to-one* mapping
  - No inbuilt mechanism to ignore *too-correct* outliers

The Perceptron

- Issues with least squares and linear discriminants
  - Linear relation cannot achieve *many-to-one* mapping
  - No inbuilt mechanism to ignore *too-correct* outliers
- Perceptron employs a *nonlinearity* to estimate target: $y(\mathbf{x}) = f(\mathbf{w}^\mathsf{T}\mathbf{x})$

The Perceptron

- Issues with least squares and linear discriminants
  - Linear relation cannot achieve *many-to-one* mapping
  - No inbuilt mechanism to ignore *too-correct* outliers
- Perceptron employs a *nonlinearity* to estimate target: $y(\mathbf{x}) = f(\mathbf{w}^\mathsf{T}\mathbf{x})$
- $f(.)$ is a hard-limiting nonlinear function given by

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

The Perceptron

- Issues with least squares and linear discriminants
  - Linear relation cannot achieve *many-to-one* mapping
  - No inbuilt mechanism to ignore *too-correct* outliers
- Perceptron employs a *nonlinearity* to estimate target: $y(\mathbf{x}) = f(\mathbf{w}^\mathsf{T}\mathbf{x})$
- $f(.)$ is a hard-limiting nonlinear function given by

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

- Desired targets are encoded as $+1$ for $\mathcal{C}_1$ and -1 for $\mathcal{C}_2$.

The Perceptron

- Issues with least squares and linear discriminants
  - Linear relation cannot achieve *many-to-one* mapping
  - No inbuilt mechanism to ignore *too-correct* outliers
- Perceptron employs a *nonlinearity* to estimate target: $y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x})$
- $f(.)$ is a hard-limiting nonlinear function given by

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

- Desired targets are encoded as $+1$ for $\mathcal{C}_1$ and -1 for $\mathcal{C}_2$.
- Estimate $\mathbf{w}$ s.t. $\mathbf{w}^T\mathbf{x}_n \geq 0$ for $x_n \in \mathcal{C}_1$, and $\mathbf{w}^T\mathbf{x}_n < 0$ for $\mathbf{x}_n \in \mathcal{C}_2$

The Perceptron

- Issues with least squares and linear discriminants
  - Linear relation cannot achieve *many-to-one* mapping
  - No inbuilt mechanism to ignore *too-correct* outliers
- Perceptron employs a *nonlinearity* to estimate target: $y(\mathbf{x}) = f(\mathbf{w}^\mathsf{T}\mathbf{x})$
- $f(.)$ is a hard-limiting nonlinear function given by

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

- Desired targets are encoded as $+1$ for $\mathcal{C}_1$ and -1 for $\mathcal{C}_2$.
- Estimate $\mathbf{w}$ s.t. $\mathbf{w}^\mathsf{T}\mathbf{x}_n \geq 0$ for $x_n \in \mathcal{C}_1$, and $\mathbf{w}^\mathsf{T}\mathbf{x}_n < 0$ for $\mathbf{x}_n \in \mathcal{C}_2$
- Both the targets and estimates are discrete

The Perceptron

- Issues with least squares and linear discriminants
  - Linear relation cannot achieve *many-to-one* mapping
  - No inbuilt mechanism to ignore *too-correct* outliers
- Perceptron employs a *nonlinearity* to estimate target: $y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x})$
- $f(.)$ is a hard-limiting nonlinear function given by

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

- Desired targets are encoded as $+1$ for $\mathcal{C}_1$ and -1 for $\mathcal{C}_2$.
- Estimate $\mathbf{w}$ s.t. $\mathbf{w}^T\mathbf{x}_n \geq 0$ for $x_n \in \mathcal{C}_1$, and $\mathbf{w}^T\mathbf{x}_n < 0$ for $\mathbf{x}_n \in \mathcal{C}_2$
- Both the targets and estimates are discrete
- Number of misclassified examples is piece-wise constant

The Perceptron Learning

The Perceptron Learning

- Perceptron criterion: defined over set of misclassified examples $\mathcal{M}$

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

The Perceptron Learning

- Perceptron criterion: defined over set of misclassified examples $\mathcal{M}$

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

- SGD can be employed to update the weight vector

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J_p(\mathbf{w})$$

The Perceptron Learning

- Perceptron criterion: defined over set of misclassified examples $\mathcal{M}$

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

- SGD can be employed to update the weight vector

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J_p(\mathbf{w}) = \mathbf{w}^{old} + \eta \sum_{n \in \mathcal{M}} \mathbf{x}_n t_n$$

The Perceptron Learning

- Perceptron criterion: defined over set of misclassified examples $\mathcal{M}$

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

- SGD can be employed to update the weight vector

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J_p(\mathbf{w}) = \mathbf{w}^{old} + \eta \sum_{n \in \mathcal{M}} \mathbf{x}_n t_n$$

  - If x_n is correctly classified, do not update the weight vector

The Perceptron Learning

- Perceptron criterion: defined over set of misclassified examples $\mathcal{M}$

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

- SGD can be employed to update the weight vector

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J_p(\mathbf{w}) = \mathbf{w}^{old} + \eta \sum_{n \in \mathcal{M}} \mathbf{x}_n t_n$$

  - If x_n is correctly classified, do not update the weight vector
  - If $\mathbf{x}_n \in \mathcal{C}_1$ is misclassified: $\mathbf{w}^{new} = \mathbf{w}^{old} + \mathbf{x}_n$

The Perceptron Learning

- Perceptron criterion: defined over set of misclassified examples $\mathcal{M}$

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

- SGD can be employed to update the weight vector

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J_p(\mathbf{w}) = \mathbf{w}^{old} + \eta \sum_{n \in \mathcal{M}} \mathbf{x}_n t_n$$

  - If x_n is correctly classified, do not update the weight vector
  - If $\mathbf{x}_n \in \mathcal{C}_1$ is misclassified: $\mathbf{w}^{new} = \mathbf{w}^{old} + \mathbf{x}_n$
  - If $\mathbf{x}_n \in \mathcal{C}_2$ is misclassified: $\mathbf{w}^{new} = \mathbf{w}^{old} - \mathbf{x}_n$

The Perceptron Learning

- Perceptron criterion: defined over set of misclassified examples $\mathcal{M}$

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

- SGD can be employed to update the weight vector

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J_p(\mathbf{w}) = \mathbf{w}^{old} + \eta \sum_{n \in \mathcal{M}} \mathbf{x}_n t_n$$

  - If x_n is correctly classified, do not update the weight vector
  - If $\mathbf{x}_n \in \mathcal{C}_1$ is misclassified: $\mathbf{w}^{new} = \mathbf{w}^{old} + \mathbf{x}_n$
  - If $\mathbf{x}_n \in \mathcal{C}_2$ is misclassified: $\mathbf{w}^{new} = \mathbf{w}^{old} - \mathbf{x}_n$

- Perceptron algorithm is guaranteed to converge in finite steps, for linearly separable classes. [Demo]

The Perceptron Learning

- Perceptron criterion: defined over set of misclassified examples $\mathcal{M}$

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

- SGD can be employed to update the weight vector

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta\nabla J_p(\mathbf{w}) = \mathbf{w}^{old} + \eta\sum_{n\in\mathcal{M}} \mathbf{x}_n t_n$$

  - If x_n is correctly classified, do not update the weight vector
  - If $\mathbf{x}_n \in \mathcal{C}_1$ is misclassified: $\mathbf{w}^{new} = \mathbf{w}^{old} + \mathbf{x}_n$
  - If $\mathbf{x}_n \in \mathcal{C}_2$ is misclassified: $\mathbf{w}^{new} = \mathbf{w}^{old} - \mathbf{x}_n$

- Perceptron algorithm is guaranteed to converge in finite steps, for linearly separable classes. [Demo]
- Perceptron algorithm is not vulnerable to *too-correct* outliers

Perceptron Convergence Theorem

Perceptron Convergence Theorem

- Starting from a random initial guess $\mathbf{w}_0$, perform K updates:

$$\mathbf{w}_K = \mathbf{w}_{K-1} + \mathbf{x}_K t_K = \mathbf{w}_0 + \sum_{k=1}^{K} \mathbf{x}_k t_k$$

where $\mathbf{x}_k$ is the randomly selected misclassified point at k^{th} iteration

Perceptron Convergence Theorem

- Starting from a random initial guess $\mathbf{w}_0$, perform K updates:

$$\mathbf{w}_K = \mathbf{w}_{K-1} + \mathbf{x}_K t_K = \mathbf{w}_0 + \sum_{k=1}^{K} \mathbf{x}_k t_k$$

  where $\mathbf{x}_k$ is the randomly selected misclassified point at k^{th} iteration

- Letting $\mathbf{w}_0 = \mathbf{0}$, the norm of $\mathbf{w}_K$ is bounded by

$$\|\mathbf{w}_K\| = \left\| \sum_{k=1}^{K} \mathbf{x}_k t_k \right\| \leq \sum_{k=1}^{K} \|\mathbf{x}_k\|$$

Perceptron Convergence Theorem

- Starting from a random initial guess $\mathbf{w}_0$, perform K updates:

$$\mathbf{w}_K = \mathbf{w}_{K-1} + \mathbf{x}_K t_K = \mathbf{w}_0 + \sum_{k=1}^{K} \mathbf{x}_k t_k$$

  where $\mathbf{x}_k$ is the randomly selected misclassified point at k^{th} iteration

- Letting $\mathbf{w}_0 = \mathbf{0}$, the norm of $\mathbf{w}_K$ is bounded by

$$\|\mathbf{w}_K\| = \left\| \sum_{k=1}^{K} \mathbf{x}_k t_k \right\| \leq \sum_{k=1}^{K} \|\mathbf{x}_k\|$$

- Let α be the maximum norm in the training data: $\alpha = \max_n \|\mathbf{x}_n\|$

Perceptron Convergence Theorem

- Starting from a random initial guess $\mathbf{w}_0$, perform K updates:

$$\mathbf{w}_K = \mathbf{w}_{K-1} + \mathbf{x}_K t_K = \mathbf{w}_0 + \sum_{k=1}^{K} \mathbf{x}_k t_k$$

  where $\mathbf{x}_k$ is the randomly selected misclassified point at k^{th} iteration

- Letting $\mathbf{w}_0 = \mathbf{0}$, the norm of $\mathbf{w}_K$ is bounded by

$$\|\mathbf{w}_K\| = \left\| \sum_{k=1}^{K} \mathbf{x}_k t_k \right\| \leq \sum_{k=1}^{K} \|\mathbf{x}_k\|$$

- Let α be the maximum norm in the training data: $\alpha = \max_n \|\mathbf{x}_n\|$
- The norm of the weight vector at K^{th} iteration is upper bounded by

$$\|\mathbf{w}_K\| \leq K\alpha$$

Perceptron Convergence Theorem

Perceptron Convergence Theorem

- Let $\mathbf{w}_*$ be one of the solutions that separates classes exactly

$$\mathbf{w}_*^\mathsf{T}\mathbf{w}_K = \sum_{k=1}^{K} \mathbf{w}_*^\mathsf{T}\mathbf{x}_k t_k$$

Perceptron Convergence Theorem

- Let $\mathbf{w}_*$ be one of the solutions that separates classes exactly

$$\mathbf{w}_*^\mathsf{T}\mathbf{w}_K = \sum_{k=1}^{K} \mathbf{w}_*^\mathsf{T}\mathbf{x}_k t_k$$

- Since $\mathbf{w}_*$ is a solution, $\mathbf{w}_*\mathbf{x}_n t_n \geq 0$ for all n in the dataset

Perceptron Convergence Theorem

- Let $\mathbf{w}_*$ be one of the solutions that separates classes exactly

$$\mathbf{w}_*^\mathsf{T}\mathbf{w}_K = \sum_{k=1}^{K} \mathbf{w}_*^\mathsf{T}\mathbf{x}_k t_k$$

- Since $\mathbf{w}_*$ is a solution, $\mathbf{w}_*\mathbf{x}_n t_n \geq 0$ for all n in the dataset
- Let β be the minimum projection onto $\mathbf{w}_*$: $\beta = \min_n \mathbf{w}_*\mathbf{x}_n t_n$

Perceptron Convergence Theorem

- Let $\mathbf{w}_*$ be one of the solutions that separates classes exactly

$$\mathbf{w}_*^\mathsf{T}\mathbf{w}_K = \sum_{k=1}^{K} \mathbf{w}_*^\mathsf{T}\mathbf{x}_k t_k$$

- Since $\mathbf{w}_*$ is a solution, $\mathbf{w}_*\mathbf{x}_n t_n \geq 0$ for all n in the dataset
- Let β be the minimum projection onto $\mathbf{w}_*$: $\beta = \min_n \mathbf{w}_*\mathbf{x}_n t_n$
- Norm of the weight vector $\mathbf{w}_K$ is lower bounded by

$$\mathbf{w}_*^\mathsf{T}\mathbf{w}_K \geq K\beta \qquad \|\mathbf{w}_K\| \geq \frac{1}{\|\mathbf{w}_*\|}K^2\beta^2$$

Perceptron Convergence Theorem

- Let $\mathbf{w}_*$ be one of the solutions that separates classes exactly

$$\mathbf{w}_*^{\mathsf{T}}\mathbf{w}_K = \sum_{k=1}^{K} \mathbf{w}_*^{\mathsf{T}}\mathbf{x}_k t_k$$

- Since $\mathbf{w}_*$ is a solution, $\mathbf{w}_*\mathbf{x}_n t_n \geq 0$ for all n in the dataset
- Let β be the minimum projection onto $\mathbf{w}_*$: $\beta = \min_n \mathbf{w}_*\mathbf{x}_n t_n$
- Norm of the weight vector $\mathbf{w}_K$ is lower bounded by

$$\mathbf{w}_*^{\mathsf{T}}\mathbf{w}_K \geq K\beta \qquad \|\mathbf{w}_K\| \geq \frac{1}{\|\mathbf{w}_*\|}K^2\beta^2$$

- The range of the norm of the weight vector is given by

$$K^2\beta' \leq \|\mathbf{w}_K\| \leq K\alpha$$

Perceptron Convergence Theorem

- Let $\mathbf{w}_*$ be one of the solutions that separates classes exactly

$$\mathbf{w}_*^\mathsf{T}\mathbf{w}_K = \sum_{k=1}^{K} \mathbf{w}_*^\mathsf{T}\mathbf{x}_k t_k$$

- Since $\mathbf{w}_*$ is a solution, $\mathbf{w}_*\mathbf{x}_n t_n \geq 0$ for all n in the dataset
- Let β be the minimum projection onto $\mathbf{w}_*$: $\beta = \min_n \mathbf{w}_*\mathbf{x}_n t_n$
- Norm of the weight vector $\mathbf{w}_K$ is lower bounded by

$$\mathbf{w}_*^\mathsf{T}\mathbf{w}_K \geq K\beta \qquad \|\mathbf{w}_K\| \geq \frac{1}{\|\mathbf{w}_*\|}K^2\beta^2$$

- The range of the norm of the weight vector is given by

$$K^2\beta' \leq \|\mathbf{w}_K\| \leq K\alpha$$

- Lower bound is quadratic in K, upper bound is linear in K.

Limitations of Perceptron

Limitations of Perceptron

- Perceptron algorithm converges only for linearly separable data

Limitations of Perceptron

- Perceptron algorithm converges only for linearly separable data
  - Number of iterations K could be very large

Limitations of Perceptron

- Perceptron algorithm converges only for linearly separable data
  - Number of iterations K could be very large
  - Error does not decrease monotonically $J_P(\mathbf{w}_{k+1}) \lesseqgtr J_P(\mathbf{w}_k)$

Limitations of Perceptron

- Perceptron algorithm converges only for linearly separable data
  - Number of iterations K could be very large
  - Error does not decrease monotonically $J_P(\mathbf{w}_{k+1}) \lesseqgtr J_P(\mathbf{w}_k)$
  - The final solution need not be optimal- one of the decision boundaries.

Limitations of Perceptron

- Perceptron algorithm converges only for linearly separable data
  - Number of iterations K could be very large
  - Error does not decrease monotonically $J_P(\mathbf{w}_{k+1}) \lesseqgtr J_P(\mathbf{w}_k)$
  - The final solution need not be optimal- one of the decision boundaries.
- Does not converge if the classes are not linearly separable - XOR

Limitations of Perceptron

- Perceptron algorithm converges only for linearly separable data
  - Number of iterations K could be very large
  - Error does not decrease monotonically $J_P(\mathbf{w}_{k+1}) \lesseqgtr J_P(\mathbf{w}_k)$
  - The final solution need not be optimal- one of the decision boundaries.
- Does not converge if the classes are not linearly separable - XOR
- Does not offer probabilistic interpretation or confidence intervals.

Limitations of Perceptron

- Perceptron algorithm converges only for linearly separable data
  - Number of iterations K could be very large
  - Error does not decrease monotonically $J_P(\mathbf{w}_{k+1}) \lesseqgtr J_P(\mathbf{w}_k)$
  - The final solution need not be optimal- one of the decision boundaries.
- Does not converge if the classes are not linearly separable - XOR
- Does not offer probabilistic interpretation or confidence intervals.
- Cannot be readily extended to multiclass problems

Decision Theory

Decision Theory

- Probability theory offers a mathematical tool to deal with uncertainty.

Decision Theory

- Probability theory offers a mathematical tool to deal with uncertainty.
- Supervised learning: Predict target $\mathbf{t}$ from observed input $\mathbf{x}$

Decision Theory

- Probability theory offers a mathematical tool to deal with uncertainty.
- Supervised learning: Predict target $\mathbf{t}$ from observed input $\mathbf{x}$
- In such a case, $p(\mathbf{x}, \mathbf{t})$ provides a complete summary of uncertainty

Decision Theory

- Probability theory offers a mathematical tool to deal with uncertainty.
- Supervised learning: Predict target $\mathbf{t}$ from observed input $\mathbf{x}$
- In such a case, $p(\mathbf{x}, \mathbf{t})$ provides a complete summary of uncertainty
- In classification task, $\mathbf{t}$ can take discrete labels $\mathcal{C}_k \quad k = 1, 2, \cdots, K$

Decision Theory

- Probability theory offers a mathematical tool to deal with uncertainty.
- Supervised learning: Predict target $\mathbf{t}$ from observed input $\mathbf{x}$
- In such a case, $p(\mathbf{x}, \mathbf{t})$ provides a complete summary of uncertainty
- In classification task, $\mathbf{t}$ can take discrete labels $\mathcal{C}_k \quad k = 1, 2, \cdots, K$
  - Estimating $p(\mathbf{x}, \mathcal{C}_k)$ from training set is referred to as *Inference* step

Decision Theory

- Probability theory offers a mathematical tool to deal with uncertainty.
- Supervised learning: Predict target $\mathbf{t}$ from observed input $\mathbf{x}$
- In such a case, $p(\mathbf{x}, \mathbf{t})$ provides a complete summary of uncertainty
- In classification task, $\mathbf{t}$ can take discrete labels $\mathcal{C}_k \quad k = 1, 2, \cdots, K$
  - Estimating $p(\mathbf{x}, \mathcal{C}_k)$ from training set is referred to as *Inference* step
  - Assigning a test-point $\mathbf{x}_t$ to one of the classes is *Decision* step

Decision Theory

- Probability theory offers a mathematical tool to deal with uncertainty.
- Supervised learning: Predict target $\mathbf{t}$ from observed input $\mathbf{x}$
- In such a case, $p(\mathbf{x}, \mathbf{t})$ provides a complete summary of uncertainty
- In classification task, $\mathbf{t}$ can take discrete labels $\mathcal{C}_k \quad k = 1, 2, \cdots, K$
  - Estimating $p(\mathbf{x}, \mathcal{C}_k)$ from training set is referred to as *Inference* step
  - Assigning a test-point $\mathbf{x}_t$ to one of the classes is *Decision* step
- Decision rule divides the input space into K decision regions $\mathcal{R}_k$

Decision Theory

- Probability theory offers a mathematical tool to deal with uncertainty.
- Supervised learning: Predict target $\mathbf{t}$ from observed input $\mathbf{x}$
- In such a case, $p(\mathbf{x}, \mathbf{t})$ provides a complete summary of uncertainty
- In classification task, $\mathbf{t}$ can take discrete labels $\mathcal{C}_k \quad k = 1, 2, \cdots, K$
  - Estimating $p(\mathbf{x}, \mathcal{C}_k)$ from training set is referred to as *Inference* step
  - Assigning a test-point $\mathbf{x}_t$ to one of the classes is *Decision* step
- Decision rule divides the input space into K decision regions $\mathcal{R}_k$
- Choose *decision boundaries* to minimize the probability of error

Decision Theory

- Probability theory offers a mathematical tool to deal with uncertainty.
- Supervised learning: Predict target $\mathbf{t}$ from observed input $\mathbf{x}$
- In such a case, $p(\mathbf{x}, \mathbf{t})$ provides a complete summary of uncertainty
- In classification task, $\mathbf{t}$ can take discrete labels $\mathcal{C}_k \quad k = 1, 2, \cdots, K$
  - Estimating $p(\mathbf{x}, \mathcal{C}_k)$ from training set is referred to as *Inference* step
  - Assigning a test-point $\mathbf{x}_t$ to one of the classes is *Decision* step
- Decision rule divides the input space into K decision regions $\mathcal{R}_k$
- Choose *decision boundaries* to minimize the probability of error
- The probability of error for binary classification is given by

$$p(\text{mistake}) = \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}$$

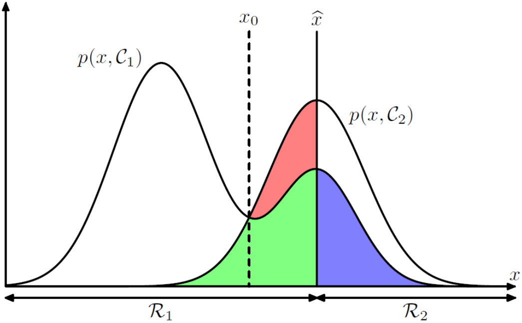
Decision Theory

- Probability theory offers a mathematical tool to deal with uncertainty.
- Supervised learning: Predict target $\mathbf{t}$ from observed input $\mathbf{x}$
- In such a case, $p(\mathbf{x}, \mathbf{t})$ provides a complete summary of uncertainty
- In classification task, $\mathbf{t}$ can take discrete labels $\mathcal{C}_k \quad k = 1, 2, \cdots, K$
  - Estimating $p(\mathbf{x}, \mathcal{C}_k)$ from training set is referred to as *Inference* step
  - Assigning a test-point $\mathbf{x}_t$ to one of the classes is *Decision* step
- Decision rule divides the input space into K decision regions $\mathcal{R}_k$
- Choose *decision boundaries* to minimize the probability of error
- The probability of error for binary classification is given by

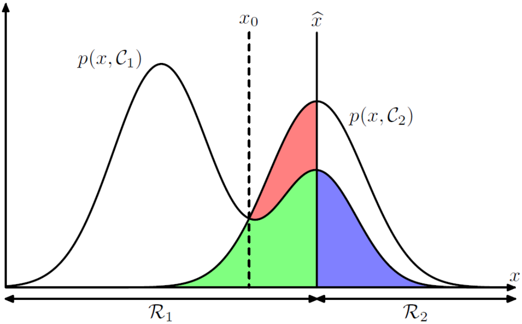
$$p(\text{mistake}) = \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}$$

- Divide input space into $\mathcal{R}_1$ and $\mathcal{R}_2$ s.t. $p(\text{mistake})$ is minimized

Choice of Decision Regions (Boundaries)

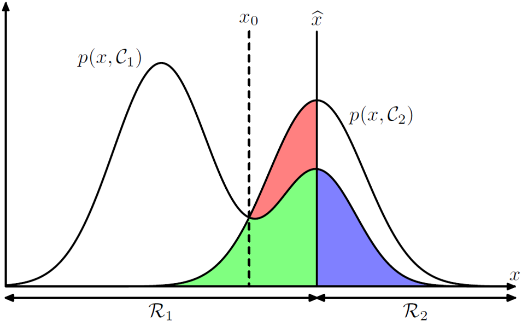


Choice of Decision Regions (Boundaries)



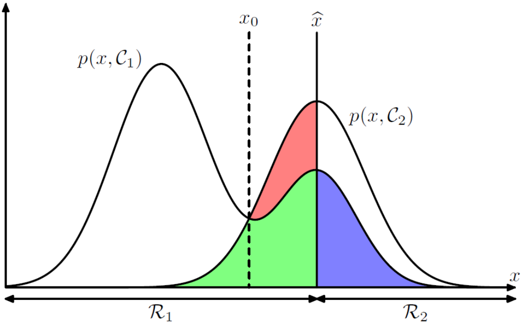
- Let $x = \hat{x}$ be decision boundary. $\mathcal{C}_1 : x \leq \hat{x}$ and $\mathcal{C}_2 : x > \hat{x}$

Choice of Decision Regions (Boundaries)



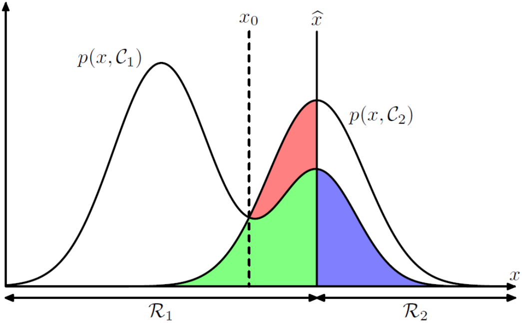
- Let $x = \hat{x}$ be decision boundary. $\mathcal{C}_1 : x \leq \hat{x}$ and $\mathcal{C}_2 : x > \hat{x}$
- Errors arise from area under the blue, red and green regions

Choice of Decision Regions (Boundaries)



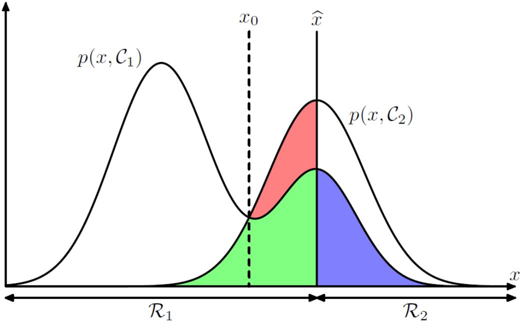
- Let $x = \hat{x}$ be decision boundary. $\mathcal{C}_1 : x \leq \hat{x}$ and $\mathcal{C}_2 : x > \hat{x}$
- Errors arise from area under the blue, red and green regions
    - Blue: Points from $\mathcal{C}_1$ but misclassified as $\mathcal{C}_2$

Choice of Decision Regions (Boundaries)



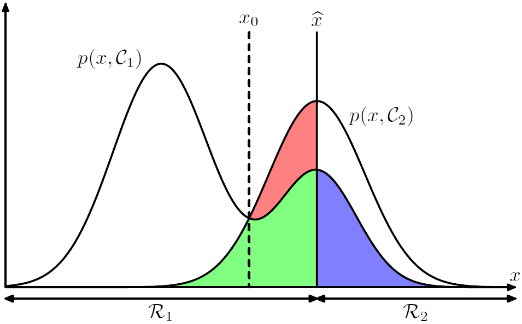
- Let $x = \hat{x}$ be decision boundary. $\mathcal{C}_1 : x \leq \hat{x}$ and $\mathcal{C}_2 : x > \hat{x}$
- Errors arise from area under the blue, red and green regions
  - Blue: Points from $\mathcal{C}_1$ but misclassified as $\mathcal{C}_2$
  - Green + Red: Points from $\mathcal{C}_2$, but misclassified as $\mathcal{C}_1$

Choice of Decision Regions (Boundaries)



- Let $x = \hat{x}$ be decision boundary. $\mathcal{C}_1 : x \leq \hat{x}$ and $\mathcal{C}_2 : x > \hat{x}$
- Errors arise from area under the blue, red and green regions
  - Blue: Points from $\mathcal{C}_1$ but misclassified as $\mathcal{C}_2$
  - Green + Red: Points from $\mathcal{C}_2$, but misclassified as $\mathcal{C}_1$
  - Adjust the boundary $\hat{x}$ to minimize the area under blue+green+red

Choice of Decision Regions (Boundaries)



- Let $x = \hat{x}$ be decision boundary. $\mathcal{C}_1 : x \leq \hat{x}$ and $\mathcal{C}_2 : x > \hat{x}$
- Errors arise from area under the blue, red and green regions
  - Blue: Points from $\mathcal{C}_1$ but misclassified as $\mathcal{C}_2$
  - Green + Red: Points from $\mathcal{C}_2$, but misclassified as $\mathcal{C}_1$
  - Adjust the boundary $\hat{x}$ to minimize the area under blue+green+red
  - Area under blue+green remains constant irrespective of choice of $\hat{x}$

Minimizing Classification Error

$$p(\text{mistake}) = \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}$$

Minimizing Classification Error

$$
\begin{aligned}
p(\text{mistake}) &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \\
&= 1 + \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} - \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}
\end{aligned}
$$

Minimizing Classification Error

$$\begin{aligned} p(\text{mistake}) &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \\ &= 1 + \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} - \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \\ &= 1 - \int_{\mathcal{R}_1} \Big(p(\mathbf{x}, \mathcal{C}_1) - p(\mathbf{x}, \mathcal{C}_2) \Big) d\mathbf{x} \end{aligned}$$

Minimizing Classification Error

$$
\begin{aligned}
p(\text{mistake}) &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \\
&= 1 + \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} - \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \\
&= 1 - \int_{\mathcal{R}_1} \Big(p(\mathbf{x}, \mathcal{C}_1) - p(\mathbf{x}, \mathcal{C}_2) \Big) d\mathbf{x}
\end{aligned}
$$

- Choose region $\mathcal{R}_1$ such that $p(\mathbf{x}, \mathcal{C}_1) > p(\mathbf{x}, \mathcal{C}_2)$

Minimizing Classification Error

$$
\begin{aligned}
p(\text{mistake}) &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \\
&= 1 + \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} - \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \\
&= 1 - \int_{\mathcal{R}_1} \Big(p(\mathbf{x}, \mathcal{C}_1) - p(\mathbf{x}, \mathcal{C}_2) \Big) d\mathbf{x}
\end{aligned}
$$

- Choose region $\mathcal{R}_1$ such that $p(\mathbf{x}, \mathcal{C}_1) > p(\mathbf{x}, \mathcal{C}_2)$
- Decision rule: Assign x $\mathcal{C}_1$ if $p[\mathcal{C}_1/\mathbf{x}] > p[\mathcal{C}_2/\mathbf{x}]$

Minimizing Classification Error

$$\begin{aligned} p(\text{mistake}) &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \\ &= 1 + \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} - \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \\ &= 1 - \int_{\mathcal{R}_1} \Big(p(\mathbf{x}, \mathcal{C}_1) - p(\mathbf{x}, \mathcal{C}_2) \Big) d\mathbf{x} \end{aligned}$$

- Choose region $\mathcal{R}_1$ such that $p(\mathbf{x}, \mathcal{C}_1) > p(\mathbf{x}, \mathcal{C}_2)$
- Decision rule: Assign x $\mathcal{C}_1$ if $p[\mathcal{C}_1/\mathbf{x}] > p[\mathcal{C}_2/\mathbf{x}]$
- For general case of K classes, it is easier to maximize $p(\text{correct})$

$$p(\text{correct}) = \sum_{k=1}^{K} \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}$$

Minimizing Classification Error

$$\begin{aligned} p(\text{mistake}) &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \\ &= 1 + \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} - \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \\ &= 1 - \int_{\mathcal{R}_1} \Big(p(\mathbf{x}, \mathcal{C}_1) - p(\mathbf{x}, \mathcal{C}_2) \Big) d\mathbf{x} \end{aligned}$$

- Choose region $\mathcal{R}_1$ such that $p(\mathbf{x}, \mathcal{C}_1) > p(\mathbf{x}, \mathcal{C}_2)$
- Decision rule: Assign x $\mathcal{C}_1$ if $p[\mathcal{C}_1/\mathbf{x}] > p[\mathcal{C}_2/\mathbf{x}]$
- For general case of K classes, it is easier to maximize $p(\text{correct})$

$$p(\text{correct}) = \sum_{k=1}^{K} \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}$$

- Decision Rule: Assign x to the class with highest posterior $p[\mathcal{C}_k/\mathbf{x}]$

Expected Loss

Expected Loss

- In reality, the errors may have varying degrees of consequences

Expected Loss

- In reality, the errors may have varying degrees of consequences
- Penalty for Healthy vs Cancer classification could be $\begin{array}{c c} & \begin{array}{cc} H & C \end{array} \\ \begin{array}{c} H \\ C \end{array} & \begin{pmatrix} 0 & 1 \\ 100 & 0 \end{pmatrix} \end{array}$

Expected Loss

- In reality, the errors may have varying degrees of consequences
- Penalty for Healthy vs Cancer classification could be $\begin{array}{c c} & \begin{array}{cc} H & C \end{array} \\ \begin{array}{c} H \\ C \end{array} & \begin{pmatrix} 0 & 1 \\ 100 & 0 \end{pmatrix} \end{array}$
- Expected loss can be obtained by weighing error with penalty

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}$$

Expected Loss

- In reality, the errors may have varying degrees of consequences
- Penalty for Healthy vs Cancer classification could be $\begin{array}{cc} & \begin{array}{cc} H & C \end{array} \\ \begin{array}{c} H \\ C \end{array} & \begin{pmatrix} 0 & 1 \\ 100 & 0 \end{pmatrix} \end{array}$
- Expected loss can be obtained by weighing error with penalty

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}$$

- Choose region $\mathcal{R}_j$ to minimize $\sum_k L_{kj} p(\mathbf{x}, \mathcal{C}_k)$

Expected Loss

- In reality, the errors may have varying degrees of consequences
- Penalty for Healthy vs Cancer classification could be $\begin{array}{cc} & \begin{array}{cc} H & C \end{array} \\ \begin{array}{c} H \\ C \end{array} & \begin{pmatrix} 0 & 1 \\ 100 & 0 \end{pmatrix} \end{array}$
- Expected loss can be obtained by weighing error with penalty

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}$$

- Choose region $\mathcal{R}_j$ to minimize $\sum_k L_{kj} p(\mathbf{x}, \mathcal{C}_k)$
- Decision Rule: Assign x to $\mathcal{C}_j$ that minimizes $\sum_k L_{kj} p[\mathcal{C}_k / \mathbf{x}]$

Expected Loss

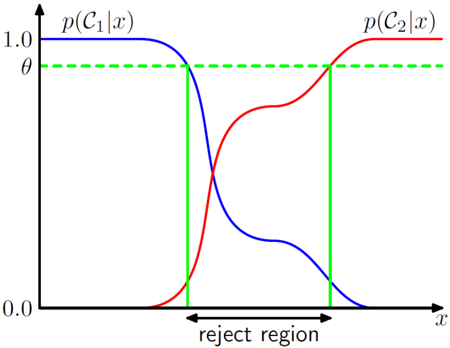
- In reality, the errors may have varying degrees of consequences
- Penalty for Healthy vs Cancer classification could be $\begin{array}{c c} & \begin{array}{cc} H & C \end{array} \\ \begin{array}{c} H \\ C \end{array} & \begin{pmatrix} 0 & 1 \\ 100 & 0 \end{pmatrix} \end{array}$
- Expected loss can be obtained by weighing error with penalty

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}$$

- Choose region $\mathcal{R}_j$ to minimize $\sum_k L_{kj} p(\mathbf{x}, \mathcal{C}_k)$
- Decision Rule: Assign x to $\mathcal{C}_j$ that minimizes $\sum_k L_{kj} p[\mathcal{C}_k/\mathbf{x}]$
- This is a trivial assignment once posterior probabilities are estimated

Reject Option

- Errors arises from regions where $\max_k p[\mathcal{C}_k/\mathbf{x}] << 1$
  - That means, all the posteriors are in similar range
  - In those regions the classifier is relatively uncertain
- In such cases, it is better to avoid decision making
  - Reject the test samples $\mathbf{x}$ for which $\max_k p[\mathcal{C}_k/\mathbf{x}] < \theta$



Homework: Expected Loss with Reject Option

- Consider a classification problem in which the loss incurred when an input vector from class $\mathcal{C}_k$ is classified as belonging to class $\mathcal{C}_j$ is given by the loss matrix L_{kj}, and for which the loss incurred in selecting the reject option is λ. Find the decision criterion that will give the minimum expected loss. Verify that this reduces to the reject criterion discussed earlier when the loss matrix is given by $L_{kj} = 1 - I_{kj}$. What is the relationship between λ and the rejection threshold θ?

Inference & Decision Stages

- Classification problem can be broken into inference and decision stages

Inference & Decision Stages

- Classification problem can be broken into inference and decision stages
- Generative models
  - Inference: Estimate posterior $p[\mathcal{C}_k/\mathbf{x}]$ from the joint density $p(\mathbf{x}, \mathcal{C}_k)$

$$p[\mathcal{C}_k/\mathbf{x}) = \frac{p[\mathcal{C}_k]p(\mathbf{x}/\mathcal{C}_k)}{p(\mathbf{x})}$$

  - Such a model can generate synthetic examples of a class from $p(\mathbf{x}, \mathcal{C}_k)$

Inference & Decision Stages

- Classification problem can be broken into inference and decision stages
- Generative models
  - Inference: Estimate posterior $p[\mathcal{C}_k/\mathbf{x}]$ from the joint density $p(\mathbf{x}, \mathcal{C}_k)$

$$p[\mathcal{C}_k/\mathbf{x}) = \frac{p[\mathcal{C}_k]p(\mathbf{x}/\mathcal{C}_k)}{p(\mathbf{x})}$$

  - Such a model can generate synthetic examples of a class from $p(\mathbf{x}, \mathcal{C}_k)$
- Discriminative models
  - Inference: Estimate posterior $p[\mathcal{C}_k/\mathbf{x}]$ as a parametric function $y(\mathbf{x}, \mathbf{w})$
  - Decision: $p[\mathcal{C}_k/\mathbf{x}]$ can be used for decision with loss and reject option

Inference & Decision Stages

- Classification problem can be broken into inference and decision stages
- Generative models
  - Inference: Estimate posterior $p[\mathcal{C}_k/\mathbf{x}]$ from the joint density $p(\mathbf{x}, \mathcal{C}_k)$

$$p[\mathcal{C}_k/\mathbf{x}) = \frac{p[\mathcal{C}_k]p(\mathbf{x}/\mathcal{C}_k)}{p(\mathbf{x})}$$

  - Such a model can generate synthetic examples of a class from $p(\mathbf{x}, \mathcal{C}_k)$
- Discriminative models
  - Inference: Estimate posterior $p[\mathcal{C}_k/\mathbf{x}]$ as a parametric function $y(\mathbf{x}, \mathbf{w})$
  - Decision: $p[\mathcal{C}_k/\mathbf{x}]$ can be used for decision with loss and reject option
- Discriminant functions
  - Find a function $y(\mathbf{x}, \mathbf{w})$ that maps input $\mathbf{x}$ to a class label
  - Inference and decision stages cannot be separated

Pros & Cons

Feature	Generate	Discriminative	Discriminant
Computation	High	Moderate	Low

Pros & Cons

Feature	Generate	Discriminative	Discriminant
Computation	High	Moderate	Low
Data Req.	Very high	Moderate	Low

Pros & Cons

Feature	Generate	Discriminative	Discriminant
Computation	High	Moderate	Low
Data Req.	Very high	Moderate	Low
Outlier detection	Yes $p(\mathbf{x})$	No	No

Pros & Cons

Feature	Generate	Discriminative	Discriminant
Computation	High	Moderate	Low
Data Req.	Very high	Moderate	Low
Outlier detection	Yes $p(\mathbf{x})$	No	No
Accuracy	Reasonable	Higher	Low

Pros & Cons

Feature	Generate	Discriminative	Discriminant
Computation	High	Moderate	Low
Data Req.	Very high	Moderate	Low
Outlier detection	Yes $p(\mathbf{x})$	No	No
Accuracy	Reasonable	Higher	Low
Minimizing Risk	Easy	Easy	Not SF

Pros & Cons

Feature	Generate	Discriminative	Discriminant
Computation	High	Moderate	Low
Data Req.	Very high	Moderate	Low
Outlier detection	Yes $p(\mathbf{x})$	No	No
Accuracy	Reasonable	Higher	Low
Minimizing Risk	Easy	Easy	Not SF
Reject option	Easy	Easy	Not SF

Pros & Cons

Feature	Generate	Discriminative	Discriminant
Computation	High	Moderate	Low
Data Req.	Very high	Moderate	Low
Outlier detection	Yes $p(\mathbf{x})$	No	No
Accuracy	Reasonable	Higher	Low
Minimizing Risk	Easy	Easy	Not SF
Reject option	Easy	Easy	Not SF
Modifying Priors	Easy	Not SF	No

Pros & Cons

Feature	Generate	Discriminative	Discriminant
Computation	High	Moderate	Low
Data Req.	Very high	Moderate	Low
Outlier detection	Yes $p(\mathbf{x})$	No	No
Accuracy	Reasonable	Higher	Low
Minimizing Risk	Easy	Easy	Not SF
Reject option	Easy	Easy	Not SF
Modifying Priors	Easy	Not SF	No
Model Fusion	Easy	Easy	Not SF

Pros & Cons

Feature	Generate	Discriminative	Discriminant
Computation	High	Moderate	Low
Data Req.	Very high	Moderate	Low
Outlier detection	Yes $p(\mathbf{x})$	No	No
Accuracy	Reasonable	Higher	Low
Minimizing Risk	Easy	Easy	Not SF
Reject option	Easy	Easy	Not SF
Modifying Priors	Easy	Not SF	No
Model Fusion	Easy	Easy	Not SF

$$P[\mathcal{C}_k/\mathbf{x}_A, \mathbf{x}_B] \propto p[\mathcal{C}_k]p(\mathbf{x}_A, \mathbf{x}_B/\mathcal{C}_k)$$

Pros & Cons

Feature	Generate	Discriminative	Discriminant
Computation	High	Moderate	Low
Data Req.	Very high	Moderate	Low
Outlier detection	Yes $p(\mathbf{x})$	No	No
Accuracy	Reasonable	Higher	Low
Minimizing Risk	Easy	Easy	Not SF
Reject option	Easy	Easy	Not SF
Modifying Priors	Easy	Not SF	No
Model Fusion	Easy	Easy	Not SF

$$\begin{aligned} P[\mathcal{C}_k/\mathbf{x}_A, \mathbf{x}_B] &\propto p[\mathcal{C}_k]p(\mathbf{x}_A, \mathbf{x}_B/\mathcal{C}_k) \\ &\propto p[\mathcal{C}_k]p(\mathbf{x}_A/\mathcal{C}_k)p(\mathbf{x}_B/\mathcal{C}_k) \qquad \text{Cond. Ind.} \end{aligned}$$

Pros & Cons

Feature	Generate	Discriminative	Discriminant
Computation	High	Moderate	Low
Data Req.	Very high	Moderate	Low
Outlier detection	Yes $p(\mathbf{x})$	No	No
Accuracy	Reasonable	Higher	Low
Minimizing Risk	Easy	Easy	Not SF
Reject option	Easy	Easy	Not SF
Modifying Priors	Easy	Not SF	No
Model Fusion	Easy	Easy	Not SF

$$
\begin{aligned}
P[\mathcal{C}_k/\mathbf{x}_A, \mathbf{x}_B] &\propto p[\mathcal{C}_k]p(\mathbf{x}_A, \mathbf{x}_B/\mathcal{C}_k) \\
&\propto p[\mathcal{C}_k]p(\mathbf{x}_A/\mathcal{C}_k)p(\mathbf{x}_B/\mathcal{C}_k) \qquad \text{Cond. Ind.} \\
&\propto \frac{p[\mathcal{C}_k/\mathbf{x}_A]p[\mathcal{C}_k/\mathbf{x}_B]}{p[\mathcal{C}_k]}
\end{aligned}
$$

Maximum Likelihood Density Estimation

Maximum Likelihood Density Estimation

- Consider data $X = \{x_1, x_2, \cdots x_N\}$ drawn from unknown distribution

Maximum Likelihood Density Estimation

- Consider data $X = \{x_1, x_2, \cdots x_N\}$ drawn from unknown distribution
- Underlying distribution $p(x_n)$ be approximated by a parametric form

Maximum Likelihood Density Estimation

- Consider data $X = \{x_1, x_2, \cdots x_N\}$ drawn from unknown distribution
- Underlying distribution $p(x_n)$ be approximated by a parametric form
  - Gaussian assumption: $p(x_n) \sim \mathcal{N}(x_n/\mu, \sigma^2)$

Maximum Likelihood Density Estimation

- Consider data $X = \{x_1, x_2, \cdots x_N\}$ drawn from unknown distribution
- Underlying distribution $p(x_n)$ be approximated by a parametric form
  - Gaussian assumption: $p(x_n) \sim \mathcal{N}(x_n/\mu, \sigma^2)$
  - Laplacian assumption: $p(x_n) \sim \mathcal{L}(x_n/\mu, b)$

Maximum Likelihood Density Estimation

- Consider data $X = \{x_1, x_2, \cdots x_N\}$ drawn from unknown distribution
- Underlying distribution $p(x_n)$ be approximated by a parametric form
  - Gaussian assumption: $p(x_n) \sim \mathcal{N}(x_n/\mu, \sigma^2)$
  - Laplacian assumption: $p(x_n) \sim \mathcal{L}(x_n/\mu, b)$
- Assuming i.i.d, the likelihood function can be written as

$$\begin{aligned} p(X/\mu, \sigma^2) &= \prod_{n=1}^{N} p(x_n/\mu, \sigma^2) \\ &= \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_n-\mu)^2}{2\sigma^2}\right) \end{aligned}$$

Maximum Likelihood Density Estimation

- Consider data $X = \{x_1, x_2, \cdots x_N\}$ drawn from unknown distribution
- Underlying distribution $p(x_n)$ be approximated by a parametric form
  - Gaussian assumption: $p(x_n) \sim \mathcal{N}(x_n/\mu, \sigma^2)$
  - Laplacian assumption: $p(x_n) \sim \mathcal{L}(x_n/\mu, b)$
- Assuming i.i.d, the likelihood function can be written as

$$
\begin{aligned}
p(X/\mu, \sigma^2) &= \prod_{n=1}^{N} p(x_n/\mu, \sigma^2) \\
&= \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_n - \mu)^2}{2\sigma^2}\right)
\end{aligned}
$$

- Estimate μ and σ to maximize the likelihood function, or eqv.

$$J(\mu, \sigma^2) = \log p(X/\mu, \sigma^2)$$

Parameter Estimation

Parameter Estimation

- μ and σ can be determined by equating the derivatives to zero

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad \hat{\sigma^2} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \hat{\mu})^2$$

Parameter Estimation

- μ and σ can be determined by equating the derivatives to zero

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad \hat{\sigma^2} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \hat{\mu})^2$$

- Checking for *bias* in estimation: Taking expectation over estimates

$$\mathbb{E}[\hat{\mu}] = \mathbb{E}\left[\frac{1}{N}\sum_{n=1}^{N} x_n\right] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[x_n] = \mu \qquad \text{unbiased}$$

$$\mathbb{E}[\hat{\sigma^2}] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[(x_n - \hat{\mu})^2] = \frac{N-1}{N}\sigma^2 \qquad \text{biased}$$

Parameter Estimation

- μ and σ can be determined by equating the derivatives to zero

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad \hat{\sigma^2} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \hat{\mu})^2$$

- Checking for *bias* in estimation: Taking expectation over estimates

$$\mathbb{E}[\hat{\mu}] = \mathbb{E}\left[\frac{1}{N}\sum_{n=1}^{N} x_n\right] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[x_n] = \mu \qquad \text{unbiased}$$

$$\mathbb{E}[\hat{\sigma^2}] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[(x_n - \hat{\mu})^2] = \frac{N-1}{N}\sigma^2 \qquad \text{biased}$$

- Variance is underestimated in ML approach. $\hat{\sigma^2} < \sigma^2$

Parameter Estimation

- μ and σ can be determined by equating the derivatives to zero

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad \hat{\sigma^2} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \hat{\mu})^2$$

- Checking for *bias* in estimation: Taking expectation over estimates

$$\mathbb{E}[\hat{\mu}] = \mathbb{E}\left[\frac{1}{N}\sum_{n=1}^{N} x_n\right] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[x_n] = \mu \qquad \text{unbiased}$$

$$\mathbb{E}[\hat{\sigma^2}] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[(x_n - \hat{\mu})^2] = \frac{N-1}{N}\sigma^2 \qquad \text{biased}$$

- Variance is underestimated in ML approach. $\hat{\sigma^2} < \sigma^2$
- The variance estimation can be corrected as $\tilde{\sigma^2} = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \hat{\mu})^2$

ML approach to Generative Models

ML approach to Generative Models

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$

ML approach to Generative Models

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ are input variables, $t_n \in \{0, 1\}$ are targets

ML approach to Generative Models

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ are input variables, $t_n \in \{0, 1\}$ are targets
  - Let N_1 points are from $\mathcal{C}_1$ and N_2 points are from $\mathcal{C}_2$: $N = N_1 + N_2$

ML approach to Generative Models

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ are input variables, $t_n \in \{0, 1\}$ are targets
  - Let N_1 points are from $\mathcal{C}_1$ and N_2 points are from $\mathcal{C}_2$: $N = N_1 + N_2$
- Generative models require estimation of joint density $p(\mathbf{x}_n, \mathcal{C}_k)$

$$p(\mathbf{x}_n, \mathcal{C}_k) = P[\mathcal{C}_k]p(\mathbf{x}_n/\mathcal{C}_k)$$

ML approach to Generative Models

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ are input variables, $t_n \in \{0, 1\}$ are targets
  - Let N_1 points are from $\mathcal{C}_1$ and N_2 points are from $\mathcal{C}_2$: $N = N_1 + N_2$
- Generative models require estimation of joint density $p(\mathbf{x}_n, \mathcal{C}_k)$

$$p(\mathbf{x}_n, \mathcal{C}_k) = P[\mathcal{C}_k]p(\mathbf{x}_n/\mathcal{C}_k)$$

- Let the priors for $\mathcal{C}_1$ and $\mathcal{C}_2$ be π and $1 - \pi$, respectively

ML approach to Generative Models

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ are input variables, $t_n \in \{0, 1\}$ are targets
  - Let N_1 points are from $\mathcal{C}_1$ and N_2 points are from $\mathcal{C}_2$: $N = N_1 + N_2$
- Generative models require estimation of joint density $p(\mathbf{x}_n, \mathcal{C}_k)$

$$p(\mathbf{x}_n, \mathcal{C}_k) = P[\mathcal{C}_k] p(\mathbf{x}_n / \mathcal{C}_k)$$

- Let the priors for $\mathcal{C}_1$ and $\mathcal{C}_2$ be π and $1 - \pi$, respectively
- Let class conditional densities $p(\mathbf{x}_n / \mathcal{C}_k)$ follow Gaussian distributions

$$p(\mathbf{x}_n / \mathcal{C}_k) \sim \mathcal{N}(\mathbf{x}_n / \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

ML approach to Generative Models

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ are input variables, $t_n \in \{0, 1\}$ are targets
  - Let N_1 points are from $\mathcal{C}_1$ and N_2 points are from $\mathcal{C}_2$: $N = N_1 + N_2$
- Generative models require estimation of joint density $p(\mathbf{x}_n, \mathcal{C}_k)$

$$p(\mathbf{x}_n, \mathcal{C}_k) = P[\mathcal{C}_k] p(\mathbf{x}_n/\mathcal{C}_k)$$

- Let the priors for $\mathcal{C}_1$ and $\mathcal{C}_2$ be π and $1 - \pi$, respectively
- Let class conditional densities $p(\mathbf{x}_n/\mathcal{C}_k)$ follow Gaussian distributions

$$p(\mathbf{x}_n/\mathcal{C}_k) \sim \mathcal{N}(\mathbf{x}_n/\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- The joint densities are given by

$$\begin{array}{lll} \text{For } \mathbf{x}_n \in \mathcal{C}_1 & p(\mathbf{x}_n/\mathcal{C}_1) & = \pi \mathcal{N}(\mathbf{x}_n/\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \\ \text{For } \mathbf{x}_n \in \mathcal{C}_2 & p(\mathbf{x}_n/\mathcal{C}_1) & = (1 - \pi) \mathcal{N}(\mathbf{x}_n/\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \end{array}$$

ML Solution

ML Solution

- Estimate $\theta = (\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ to maximize the likelihood function

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

ML Solution

- Estimate $\theta = (\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ to maximize the likelihood function

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

- It is convenient to maximize log-likelihood $\log p(\mathbf{t}/\theta)$

ML Solution

- Estimate $\theta = (\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ to maximize the likelihood function

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

- It is convenient to maximize log-likelihood $\log p(\mathbf{t}/\theta)$
- The terms involving π in log-likelihood $\mathcal{L}(\theta)$ are

$$\sum_{n=1}^{N} t_n \log \pi + (1-t_n)\log(1-\pi)$$

ML Solution

- Estimate $\theta = (\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ to maximize the likelihood function

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

- It is convenient to maximize log-likelihood $\log p(\mathbf{t}/\theta)$
- The terms involving π in log-likelihood $\mathcal{L}(\theta)$ are

$$\sum_{n=1}^{N} t_n \log \pi + (1 - t_n) \log(1 - \pi)$$

  - Setting the derivative of $\mathcal{L}(\theta)$ w.r.t to π to zero,

$$\pi = \frac{N_1}{N} \implies p[\mathcal{C}_1] = \frac{N_1}{N} \text{ and } p[\mathcal{C}_2] = \frac{N_2}{N}$$

ML Solution

- Estimate $\theta = (\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ to maximize the likelihood function

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

- It is convenient to maximize log-likelihood $\log p(\mathbf{t}/\theta)$
- The terms involving π in log-likelihood $\mathcal{L}(\theta)$ are

$$\sum_{n=1}^{N} t_n \log \pi + (1-t_n)\log(1-\pi)$$

  - Setting the derivative of $\mathcal{L}(\theta)$ w.r.t to π to zero,

$$\pi = \frac{N_1}{N} \implies p[\mathcal{C}_1] = \frac{N_1}{N} \text{ and } p[\mathcal{C}_2] = \frac{N_2}{N}$$

ML Solution

ML Solution

- $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ can be evaluated in a similar manner

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n \qquad \boldsymbol{\Sigma}_1 = \tfrac{1}{N_1} \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathsf{T}} (\mathbf{x}_n - \boldsymbol{\mu}_1)$$

$$\boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n \qquad \boldsymbol{\Sigma}_2 = \tfrac{1}{N_2} \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)^{\mathsf{T}} (\mathbf{x}_n - \boldsymbol{\mu}_2)$$

ML Solution

- $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ can be evaluated in a similar manner

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

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

- Equivalent to estimating individual CCDs

ML Solution

- $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ can be evaluated in a similar manner

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n \qquad \boldsymbol{\Sigma}_1 = \tfrac{1}{N_1} \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathsf{T}} (\mathbf{x}_n - \boldsymbol{\mu}_1)$$

$$\boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n \qquad \boldsymbol{\Sigma}_2 = \tfrac{1}{N_2} \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)^{\mathsf{T}} (\mathbf{x}_n - \boldsymbol{\mu}_2)$$

- Equivalent to estimating individual CCDs
- While estimating CCD of one class, data from other class is not considered

ML Solution

- $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ can be evaluated in a similar manner

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

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

- Equivalent to estimating individual CCDs
- While estimating CCD of one class, data from other class is not considered
- Offers a descriptive model, but need not be discriminative

Home Work - Shared Covariance Matrix

Home Work - Shared Covariance Matrix

- Let both the classes share the same covaraince matrix

$$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$$

Home Work - Shared Covariance Matrix

- Let both the classes share the same covaraince matrix

$$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$$

- The shape of the distributions is same for both classes

Home Work - Shared Covariance Matrix

- Let both the classes share the same covaraince matrix

$$\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \mathbf{\Sigma}$$

- The shape of the distributions is same for both classes
- The ML solution to the shared covariance matrix is given by

$$\mathbf{\Sigma} = \frac{N_1}{N}\mathbf{\Sigma}_1 + \frac{N_2}{N}\mathbf{\Sigma}_2$$

Home Work - Shared Covariance Matrix

- Let both the classes share the same covaraince matrix

$$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$$

- The shape of the distributions is same for both classes
- The ML solution to the shared covariance matrix is given by

$$\boldsymbol{\Sigma} = \frac{N_1}{N}\boldsymbol{\Sigma}_1 + \frac{N_2}{N}\boldsymbol{\Sigma}_2$$

- Shared covariance is given by a weighted combination of class-specific covariances.

Decision Boundary(Shared Covariance)

Decision Boundary(Shared Covariance)

- The decision boundary is locus of points satisfying $P[\mathcal{C}_1/\mathbf{x}] = P[\mathcal{C}_2/\mathbf{x}]$

$$P[\mathcal{C}_1]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = p[\mathcal{C}_2]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

Decision Boundary(Shared Covariance)

- The decision boundary is locus of points satisfying $P[\mathcal{C}_1/\mathbf{x}] = P[\mathcal{C}_2/\mathbf{x}]$

$$P[\mathcal{C}_1]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = p[\mathcal{C}_2]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

- The quadratic terms cancel because of shared covariance

Decision Boundary(Shared Covariance)

- The decision boundary is locus of points satisfying $P[\mathcal{C}_1/\mathbf{x}] = P[\mathcal{C}_2/\mathbf{x}]$

$$P[\mathcal{C}_1]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = p[\mathcal{C}_2]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

- The quadratic terms cancel because of shared covariance
- The decision boundary is a linear in x: $\mathbf{w}^\mathsf{T}\mathbf{x} + w_0$

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

Decision Boundary(Shared Covariance)

- The decision boundary is locus of points satisfying $P[\mathcal{C}_1/\mathbf{x}] = P[\mathcal{C}_2/\mathbf{x}]$

$$P[\mathcal{C}_1]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = p[\mathcal{C}_2]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

- The quadratic terms cancel because of shared covariance
- The decision boundary is a linear in x: $\mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0$

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \quad w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \log\frac{P[\mathcal{C}_1]}{P[\mathcal{C}_2]}$$

Decision Boundary(Shared Covariance)

- The decision boundary is locus of points satisfying $P[\mathcal{C}_1/\mathbf{x}] = P[\mathcal{C}_2/\mathbf{x}]$

$$P[\mathcal{C}_1]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = p[\mathcal{C}_2]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

- The quadratic terms cancel because of shared covariance
- The decision boundary is a linear in x: $\mathbf{w}^\mathsf{T}\mathbf{x} + w_0$

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \quad w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \log\frac{P[\mathcal{C}_1]}{P[\mathcal{C}_2]}$$

- Priors affect only the bias parameters, not the orientation

Decision Boundary(Shared Covariance)

- The decision boundary is locus of points satisfying $P[\mathcal{C}_1/\mathbf{x}] = P[\mathcal{C}_2/\mathbf{x}]$

$$P[\mathcal{C}_1]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = p[\mathcal{C}_2]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

- The quadratic terms cancel because of shared covariance
- The decision boundary is a linear in x: $\mathbf{w}^\mathsf{T}\mathbf{x} + w_0$

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \quad w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \log\frac{P[\mathcal{C}_1]}{P[\mathcal{C}_2]}$$

- Priors affect only the bias parameters, not the orientation
- The posterior density of $\mathcal{C}_1$ is given by $P[\mathcal{C}_1/\mathbf{x}] = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x} + w_0)$

Decision Boundary(Shared Covariance)

- The decision boundary is locus of points satisfying $P[\mathcal{C}_1/\mathbf{x}] = P[\mathcal{C}_2/\mathbf{x}]$

$$P[\mathcal{C}_1]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = p[\mathcal{C}_2]\ \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

- The quadratic terms cancel because of shared covariance
- The decision boundary is a linear in x: $\mathbf{w}^\mathsf{T}\mathbf{x} + w_0$

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \quad w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \log\frac{P[\mathcal{C}_1]}{P[\mathcal{C}_2]}$$

- Priors affect only the bias parameters, not the orientation
- The posterior density of $\mathcal{C}_1$ is given by $P[\mathcal{C}_1/\mathbf{x}] = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x} + w_0)$
- The decision boundary would be quadratic, if covariance is not shared

Illustration of Decision Boundaries

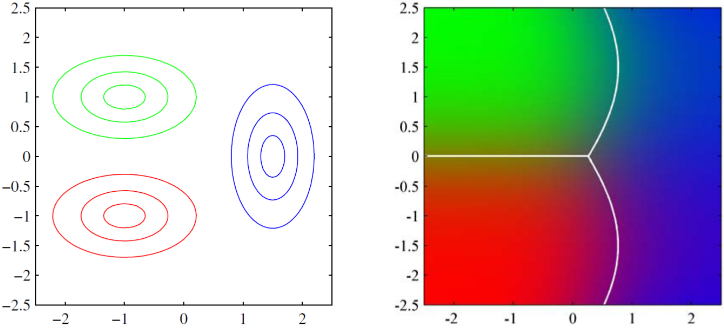
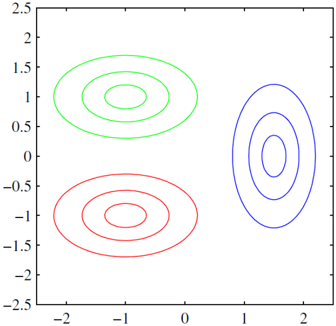
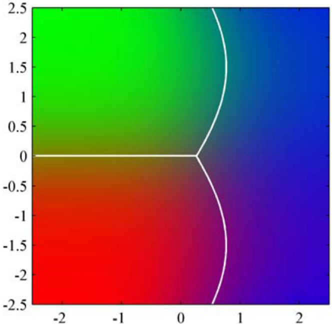


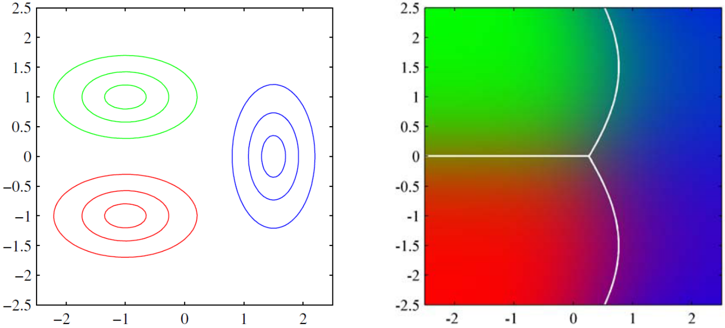
Illustration of Decision Boundaries





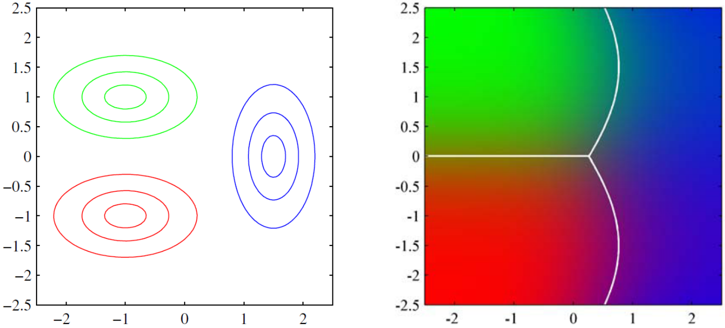
- Red and Green classes share the same covariance matrix

Illustration of Decision Boundaries



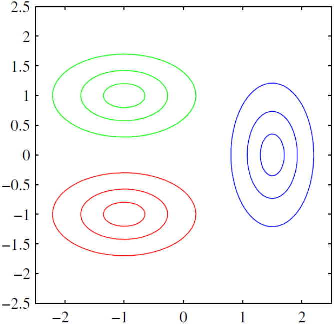
- Red and Green classes share the same covariance matrix
    - Decision boundary is linear

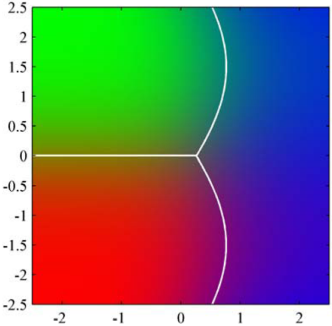
Illustration of Decision Boundaries



- Red and Green classes share the same covariance matrix
  - Decision boundary is linear
- Blue has a different covariance - decision boundaries are quadratic

Illustration of Decision Boundaries





- Red and Green classes share the same covariance matrix
  - Decision boundary is linear
- Blue has a different covariance - decision boundaries are quadratic
- Nonlinear decision boundaries can be modeled with pdfs having higher order moments!

Probabilistic Discriminative Models

Probabilistic Discriminative Models

- Discriminative models impose a parametric function on posterior

$$P[\mathcal{C}_k/\mathbf{x}] = y(\mathbf{x}, \mathbf{W})$$

Probabilistic Discriminative Models

- Discriminative models impose a parametric function on posterior

$$P[\mathcal{C}_k/\mathbf{x}] = y(\mathbf{x}, \mathbf{W})$$

- The function $y(.)$ should satisfy the axioms of probability

Probabilistic Discriminative Models

- Discriminative models impose a parametric function on posterior

$$P[\mathcal{C}_k/\mathbf{x}] = y(\mathbf{x}, \mathbf{W})$$

- The function $y(.)$ should satisfy the axioms of probability
- The posterior probability of the k^{th} class is given by (Bayes)

$$P[\mathcal{C}_k/\mathbf{x}] = \frac{P[\mathcal{C}_k]p(\mathbf{x}/\mathcal{C}_k)}{\sum_{j=1}^{K} P[\mathcal{C}_j]p(\mathbf{x}/\mathcal{C}_j)}$$

Probabilistic Discriminative Models

- Discriminative models impose a parametric function on posterior

$$P[\mathcal{C}_k/\mathbf{x}] = y(\mathbf{x}, \mathbf{W})$$

- The function $y(.)$ should satisfy the axioms of probability
- The posterior probability of the k^{th} class is given by (Bayes)

$$P[\mathcal{C}_k/\mathbf{x}] = \frac{P[\mathcal{C}_k]p(\mathbf{x}/\mathcal{C}_k)}{\sum\limits_{j=1}^{K} P[\mathcal{C}_j]p(\mathbf{x}/\mathcal{C}_j)}$$

- Let $a_k = \log P[\mathcal{C}_k]p(\mathbf{x}/\mathcal{C}_k)$ be parameterized as $a_k = \mathbf{w}_k^{\mathsf{T}}\mathbf{x}$

Probabilistic Discriminative Models

- Discriminative models impose a parametric function on posterior

$$P[\mathcal{C}_k/\mathbf{x}] = y(\mathbf{x}, \mathbf{W})$$

- The function $y(.)$ should satisfy the axioms of probability
- The posterior probability of the k^{th} class is given by (Bayes)

$$P[\mathcal{C}_k/\mathbf{x}] = \frac{P[\mathcal{C}_k]p(\mathbf{x}/\mathcal{C}_k)}{\sum_{j=1}^{K} P[\mathcal{C}_j]p(\mathbf{x}/\mathcal{C}_j)}$$

- Let $a_k = \log P[\mathcal{C}_k]p(\mathbf{x}/\mathcal{C}_k)$ be parameterized as $a_k = \mathbf{w}_k^T\mathbf{x}$
- Posterior probability can be expressed as a softmax over activations a_k

$$P[\mathcal{C}_k/\mathbf{x}] = \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)} \qquad \text{Softmax Fn.}$$

Binary Classifier

Binary Classifier

- For binary classifier, we need not evaluate two weight vectors.

Binary Classifier

- For binary classifier, we need not evaluate two weight vectors.
- The posterior probability can be expressed as

$$p[\mathcal{C}_1/\mathbf{x}] = \frac{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1)}{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1) + P[\mathcal{C}_2]p(\mathbf{x}/\mathcal{C}_2)}$$

Binary Classifier

- For binary classifier, we need not evaluate two weight vectors.
- The posterior probability can be expressed as

$$p[\mathcal{C}_1/\mathbf{x}] = \frac{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1)}{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1) + P[\mathcal{C}_2]p(\mathbf{x}/\mathcal{C}_2)}$$

- Let $a = -\log \frac{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1)}{P[\mathcal{C}_2]p(\mathbf{x}/\mathcal{C}_2)}$ be parameterized as $a = \mathbf{w}^\mathsf{T}\mathbf{x}$

Binary Classifier

- For binary classifier, we need not evaluate two weight vectors.
- The posterior probability can be expressed as

$$p[\mathcal{C}_1/\mathbf{x}] = \frac{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1)}{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1) + P[\mathcal{C}_2]p(\mathbf{x}/\mathcal{C}_2)}$$

- Let $a = -\log \frac{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1)}{P[\mathcal{C}_2]p(\mathbf{x}/\mathcal{C}_2)}$ be parameterized as $a = \mathbf{w}^\mathsf{T}\mathbf{x}$
- Posterior probability of $\mathcal{C}_1$ can be expressed as Sigmoid over a

$$P[\mathcal{C}_1/\mathbf{x}] = \frac{1}{1 + \exp(-a)} = \sigma(a)$$

Binary Classifier

- For binary classifier, we need not evaluate two weight vectors.
- The posterior probability can be expressed as

$$p[\mathcal{C}_1/\mathbf{x}] = \frac{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1)}{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1) + P[\mathcal{C}_2]p(\mathbf{x}/\mathcal{C}_2)}$$

- Let $a = -\log \frac{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1)}{P[\mathcal{C}_2]p(\mathbf{x}/\mathcal{C}_2)}$ be parameterized as $a = \mathbf{w}^{\mathsf{T}}\mathbf{x}$
- Posterior probability of $\mathcal{C}_1$ can be expressed as Sigmoid over a

$$P[\mathcal{C}_1/\mathbf{x}] = \frac{1}{1 + \exp(-a)} = \sigma(a)$$

- Posterior probability of $\mathcal{C}_2$ is given by $P[\mathcal{C}_2/\mathbf{x}] = 1 - \sigma(a)$

Binary Classifier

- For binary classifier, we need not evaluate two weight vectors.
- The posterior probability can be expressed as

$$p[\mathcal{C}_1/\mathbf{x}] = \frac{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1)}{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1) + P[\mathcal{C}_2]p(\mathbf{x}/\mathcal{C}_2)}$$

- Let $a = -\log \frac{P[\mathcal{C}_1]p(\mathbf{x}/\mathcal{C}_1)}{P[\mathcal{C}_2]p(\mathbf{x}/\mathcal{C}_2)}$ be parameterized as $a = \mathbf{w}^{\mathsf{T}}\mathbf{x}$
- Posterior probability of $\mathcal{C}_1$ can be expressed as Sigmoid over a

$$P[\mathcal{C}_1/\mathbf{x}] = \frac{1}{1 + \exp(-a)} = \sigma(a)$$

- Posterior probability of $\mathcal{C}_2$ is given by $P[\mathcal{C}_2/\mathbf{x}] = 1 - \sigma(a)$
- Derivative of sigmoid function:

$$\frac{d\sigma}{da} = \sigma(a)(1 - \sigma(a))$$

Logistic Regression

Logistic Regression

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$

Logistic Regression

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ are input variables, $t_n \in \{0, 1\}$ are targets: $P[\mathcal{C}_1/\mathbf{x}_n]$

Logistic Regression

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ are input variables, $t_n \in \{0, 1\}$ are targets: $P[\mathcal{C}_1/\mathbf{x}_n]$
  - Let the target for $\mathcal{C}_1$ be 1 and $\mathcal{C}_2$ be 0

Logistic Regression

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ are input variables, $t_n \in \{0, 1\}$ are targets: $P[\mathcal{C}_1/\mathbf{x}_n]$
  - Let the target for $\mathcal{C}_1$ be 1 and $\mathcal{C}_2$ be 0
- Let the posterior probability be estimated as

$$\hat{P}[\mathcal{C}_1/\mathbf{x}_n] = y(\mathbf{x}_n, \mathbf{w}) = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}_n) \qquad \hat{P}[\mathcal{C}_2/\mathbf{x}_n] = 1 - \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}_n)$$

Logistic Regression

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ are input variables, $t_n \in \{0, 1\}$ are targets: $P[\mathcal{C}_1/\mathbf{x}_n]$
  - Let the target for $\mathcal{C}_1$ be 1 and $\mathcal{C}_2$ be 0
- Let the posterior probability be estimated as

$$\hat{P}[\mathcal{C}_1/\mathbf{x}_n] = y(\mathbf{x}_n, \mathbf{w}) = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}_n) \qquad \hat{P}[\mathcal{C}_2/\mathbf{x}_n] = 1 - \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}_n)$$

- Assuming that data points are i.i.d., the likelihood of data is given by

$$P(\mathbf{t}/\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}$$

Logistic Regression

- Training Data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_n, t_n), \cdots, (\mathbf{x}_N, t_N)\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ are input variables, $t_n \in \{0, 1\}$ are targets: $P[\mathcal{C}_1/\mathbf{x}_n]$
  - Let the target for $\mathcal{C}_1$ be 1 and $\mathcal{C}_2$ be 0
- Let the posterior probability be estimated as

$$\hat{P}[\mathcal{C}_1/\mathbf{x}_n] = y(\mathbf{x}_n, \mathbf{w}) = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}_n) \qquad \hat{P}[\mathcal{C}_2/\mathbf{x}_n] = 1 - \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}_n)$$

- Assuming that data points are i.i.d., the likelihood of data is given by

$$P(\mathbf{t}/\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}$$

- $\mathbf{w}$ can be estimated by minimizing negative log of the likelihood, also referred to as *cross-entropy* loss

$$J(\mathbf{w}) = -\log P(\mathbf{t}/\mathbf{X}, \mathbf{w}) = -\sum_{n=1}^{N}\{t_n \log y_n + (1 - t_n)\log(1 - y_n)\}$$

Parameter Estimation

Parameter Estimation

- Model parameters $\mathbf{w}$ can be updated using gradient descent

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J(\mathbf{w})$$

Parameter Estimation

- Model parameters **w** can be updated using gradient descent

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J(\mathbf{w})$$

- The 1st and 2nd order derivatives of the loss function are given by

Parameter Estimation

- Model parameters **w** can be updated using gradient descent

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J(\mathbf{w})$$

- The 1st and 2nd order derivatives of the loss function are given by

$$\nabla J(\mathbf{w}) = \sum_{n=1}^{N} \mathbf{x}_n (y_n - t_n) = \mathbf{X}^{\mathsf{T}} (\mathbf{y} - \mathbf{t})$$

Parameter Estimation

- Model parameters $\mathbf{w}$ can be updated using gradient descent

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J(\mathbf{w})$$

- The 1st and 2nd order derivatives of the loss function are given by

$$\nabla J(\mathbf{w}) = \sum_{n=1}^{N} \mathbf{x}_n (y_n - t_n) = \mathbf{X}^{\mathsf{T}}(\mathbf{y} - \mathbf{t})$$

$$\mathbf{H} = \nabla^2 J(\mathbf{w}) = \sum_{n=1}^{N} \mathbf{x}_n y_n (1 - y_n) \mathbf{x}_n^{\mathsf{T}} = \mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}$$

Parameter Estimation

- Model parameters **w** can be updated using gradient descent

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J(\mathbf{w})$$

- The 1st and 2nd order derivatives of the loss function are given by

$$\nabla J(\mathbf{w}) = \sum_{n=1}^{N} \mathbf{x}_n (y_n - t_n) = \mathbf{X}^\mathsf{T}(\mathbf{y} - \mathbf{t})$$

$$\mathbf{H} = \nabla^2 J(\mathbf{w}) = \sum_{n=1}^{N} \mathbf{x}_n y_n (1 - y_n) \mathbf{x}_n^\mathsf{T} = \mathbf{X}^\mathsf{T}\mathbf{R}\mathbf{X}$$

where **R** is a diagonal matrix with entries $R_{nn} = y_n(1 - y_n)$

Parameter Estimation

- Model parameters $\mathbf{w}$ can be updated using gradient descent

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J(\mathbf{w})$$

- The 1st and 2nd order derivatives of the loss function are given by

$$\nabla J(\mathbf{w}) = \sum_{n=1}^{N} \mathbf{x}_n (y_n - t_n) = \mathbf{X}^\mathsf{T} (\mathbf{y} - \mathbf{t})$$

$$\mathbf{H} = \nabla^2 J(\mathbf{w}) = \sum_{n=1}^{N} \mathbf{x}_n y_n (1 - y_n) \mathbf{x}_n^\mathsf{T} = \mathbf{X}^\mathsf{T} \mathbf{R} \mathbf{X}$$

  where $\mathbf{R}$ is a diagonal matrix with entries $R_{nn} = y_n(1 - y_n)$

- Hessian matrix varies depending on parameters $\mathbf{w}$ through $\mathbf{R}$

Parameter Estimation

- Model parameters $\mathbf{w}$ can be updated using gradient descent

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J(\mathbf{w})$$

- The 1st and 2nd order derivatives of the loss function are given by

$$\nabla J(\mathbf{w}) = \sum_{n=1}^{N} \mathbf{x}_n (y_n - t_n) = \mathbf{X}^{\mathsf{T}}(\mathbf{y} - \mathbf{t})$$

$$\mathbf{H} = \nabla^2 J(\mathbf{w}) = \sum_{n=1}^{N} \mathbf{x}_n y_n (1 - y_n) \mathbf{x}_n^{\mathsf{T}} = \mathbf{X}^{\mathsf{T}} \mathbf{R} \mathbf{X}$$

  where $\mathbf{R}$ is a diagonal matrix with entries $R_{nn} = y_n(1 - y_n)$

- Hessian matrix varies depending on parameters $\mathbf{w}$ through $\mathbf{R}$
- Since $0 < y_n < 1$, Hessian matrix $\mathbf{H}$ is positive definite: $\mathbf{u}^{\mathsf{T}} \mathbf{H} \mathbf{u} > 0$

Parameter Estimation

- Model parameters $\mathbf{w}$ can be updated using gradient descent

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \nabla J(\mathbf{w})$$

- The 1st and 2nd order derivatives of the loss function are given by

$$\nabla J(\mathbf{w}) = \sum_{n=1}^{N} \mathbf{x}_n (y_n - t_n) = \mathbf{X}^\mathsf{T} (\mathbf{y} - \mathbf{t})$$

$$\mathbf{H} = \nabla^2 J(\mathbf{w}) = \sum_{n=1}^{N} \mathbf{x}_n y_n (1 - y_n) \mathbf{x}_n^\mathsf{T} = \mathbf{X}^\mathsf{T} \mathbf{R} \mathbf{X}$$

  where $\mathbf{R}$ is a diagonal matrix with entries $R_{nn} = y_n(1 - y_n)$
- Hessian matrix varies depending on parameters $\mathbf{w}$ through $\mathbf{R}$
- Since $0 < y_n < 1$, Hessian matrix $\mathbf{H}$ is positive definite: $\mathbf{u}^\mathsf{T} \mathbf{H} \mathbf{u} > 0$
- Error function $J(\mathbf{w})$ is convex in $\mathbf{w}$ $\implies$ admits unique minimum

Iterative Reweighted Least Squares

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}(\mathbf{y} - \mathbf{t})$$

Iterative Reweighted Least Squares

$$
\begin{aligned}
\mathbf{w}^{new} &= \mathbf{w}^{old} - \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t}) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\mathbf{w}^{old} - \mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t})\right)
\end{aligned}
$$

Iterative Reweighted Least Squares

$$
\begin{aligned}
\mathbf{w}^{new} &= \mathbf{w}^{old} - \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t}) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\mathbf{w}^{old} - \mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t})\right) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{R}\left(\mathbf{X}\mathbf{w}^{old} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})\right)
\end{aligned}
$$

Iterative Reweighted Least Squares

$$
\begin{aligned}
\mathbf{w}^{new} &= \mathbf{w}^{old} - \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t}) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\mathbf{w}^{old} - \mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t})\right) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{R}\left(\mathbf{X}\mathbf{w}^{old} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})\right) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{z}
\end{aligned}
$$

Iterative Reweighted Least Squares

$$
\begin{aligned}
\mathbf{w}^{new} &= \mathbf{w}^{old} - \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t}) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\mathbf{w}^{old} - \mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t})\right) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{R}\left(\mathbf{X}\mathbf{w}^{old} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})\right) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{z}
\end{aligned}
$$

where $\mathbf{z} = \left(\mathbf{X}\mathbf{w}^{old} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})\right)$.

Iterative Reweighted Least Squares

$$
\begin{aligned}
\mathbf{w}^{new} &= \mathbf{w}^{old} - \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t}) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\mathbf{w}^{old} - \mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t})\right) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{R}\left(\mathbf{X}\mathbf{w}^{old} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})\right) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{z}
\end{aligned}
$$

where $\mathbf{z} = \left(\mathbf{X}\mathbf{w}^{old} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})\right)$.

- The solution takes the form of normal equations of weighted LS.

Iterative Reweighted Least Squares

$$
\begin{aligned}
\mathbf{w}^{new} &= \mathbf{w}^{old} - \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t}) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\mathbf{w}^{old} - \mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t})\right) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{R}\left(\mathbf{X}\mathbf{w}^{old} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})\right) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{z}
\end{aligned}
$$

where $\mathbf{z} = \left(\mathbf{X}\mathbf{w}^{old} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})\right)$.

- The solution takes the form of normal equations of weighted LS.
- However, the weighing matrix $\mathbf{R}$ is not constant, but depends on $\mathbf{w}$

Iterative Reweighted Least Squares

$$
\begin{aligned}
\mathbf{w}^{new} &= \mathbf{w}^{old} - \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t}) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\mathbf{w}^{old} - \mathbf{X}^{\mathsf{T}}(\mathbf{y}-\mathbf{t})\right) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{R}\left(\mathbf{X}\mathbf{w}^{old} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})\right) \\
&= \left(\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{R}\mathbf{z}
\end{aligned}
$$

where $\mathbf{z} = \left(\mathbf{X}\mathbf{w}^{old} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})\right)$.

- The solution takes the form of normal equations of weighted LS.
- However, the weighing matrix **R** is not constant, but depends on **w**
- Hence, the normal equations need to be applied iteratively.

Multiclass Logistic Regression (Homework)

Multiclass Logistic Regression (Homework)

- Consider multiclass data examples denoted by $X = \{(\mathbf{x}_{1:N}, \mathbf{t}_{1:N})\}$

Multiclass Logistic Regression (Homework)

- Consider multiclass data examples denoted by $X = \{(\mathbf{x}_{1:N}, \mathbf{t}_{1:N})\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ represents input observations

Multiclass Logistic Regression (Homework)

- Consider multiclass data examples denoted by $X = \{(\mathbf{x}_{1:N}, \mathbf{t}_{1:N})\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ represents input observations
  - $\mathbf{t}_n$ is a K-dim one-hot vector denoting the class posteriors

Multiclass Logistic Regression (Homework)

- Consider multiclass data examples denoted by $X = \{(\mathbf{x}_{1:N}, \mathbf{t}_{1:N})\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ represents input observations
  - $\mathbf{t}_n$ is a K-dim one-hot vector denoting the class posteriors
- For this case, the posterior probabilities can be estimated as

$$y(\mathbf{x}_n, \mathbf{w}_k) = \hat{P}[\mathcal{C}_k/\mathbf{x}_n] = \frac{\exp(a_{nk})}{\sum_{j=1}^{K} \exp(a_{nj})} \qquad a_{nk} = \mathbf{w}_k^{\mathsf{T}}\mathbf{x}_n$$

Multiclass Logistic Regression (Homework)

- Consider multiclass data examples denoted by $X = \{(\mathbf{x}_{1:N}, \mathbf{t}_{1:N})\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ represents input observations
  - $\mathbf{t}_n$ is a K-dim one-hot vector denoting the class posteriors
- For this case, the posterior probabilities can be estimated as

$$y(\mathbf{x}_n, \mathbf{w}_k) = \hat{P}[\mathcal{C}_k/\mathbf{x}_n] = \frac{\exp(a_{nk})}{\sum_{j=1}^{K} \exp(a_{nj})} \qquad a_{nk} = \mathbf{w}_k^{\mathsf{T}}\mathbf{x}_n$$

- The likelihood function can be written as

$$P[\mathbf{T}/\mathbf{X}, \mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K] = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

Multiclass Logistic Regression (Homework)

- Consider multiclass data examples denoted by $X = \{(\mathbf{x}_{1:N}, \mathbf{t}_{1:N})\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ represents input observations
  - $\mathbf{t}_n$ is a K-dim one-hot vector denoting the class posteriors
- For this case, the posterior probabilities can be estimated as

$$y(\mathbf{x}_n, \mathbf{w}_k) = \hat{P}[\mathcal{C}_k/\mathbf{x}_n] = \frac{\exp(a_{nk})}{\sum_{j=1}^{K} \exp(a_{nj})} \qquad a_{nk} = \mathbf{w}_k^{\mathsf{T}}\mathbf{x}_n$$

- The likelihood function can be written as

$$P[\mathbf{T}/\mathbf{X}, \mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K] = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

$$J(\mathbf{W}) = -P[\mathbf{T}/\mathbf{X}, \mathbf{W}] = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\log y_{nk} \qquad \text{(CE)}$$

Multiclass Logistic Regression (Homework)

- Consider multiclass data examples denoted by $X = \{(\mathbf{x}_{1:N}, \mathbf{t}_{1:N})\}$
  - $\mathbf{x}_n \in \mathbb{R}^D$ represents input observations
  - $\mathbf{t}_n$ is a K-dim one-hot vector denoting the class posteriors
- For this case, the posterior probabilities can be estimated as

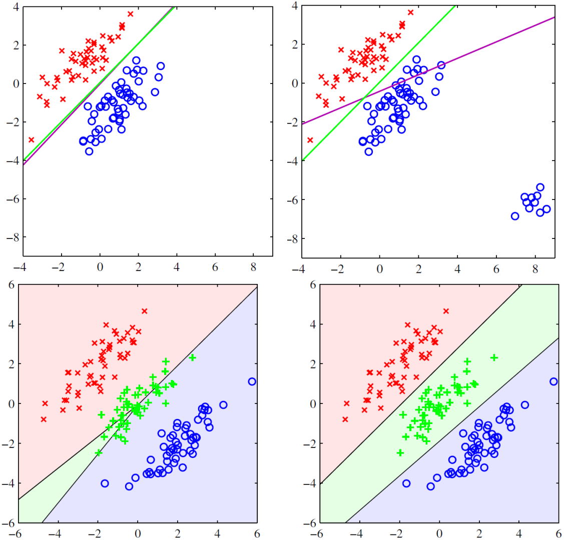
$$y(\mathbf{x}_n, \mathbf{w}_k) = \hat{P}[\mathcal{C}_k/\mathbf{x}_n] = \frac{\exp(a_{nk})}{\sum_{j=1}^{K} \exp(a_{nj})} \qquad a_{nk} = \mathbf{w}_k^{\mathsf{T}}\mathbf{x}_n$$

- The likelihood function can be written as

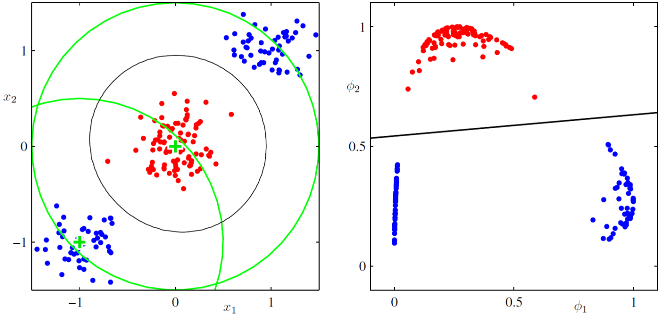
$$P[\mathbf{T}/\mathbf{X}, \mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K] = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

$$J(\mathbf{W}) = -P[\mathbf{T}/\mathbf{X}, \mathbf{W}] = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \log y_{nk} \qquad \text{(CE)}$$

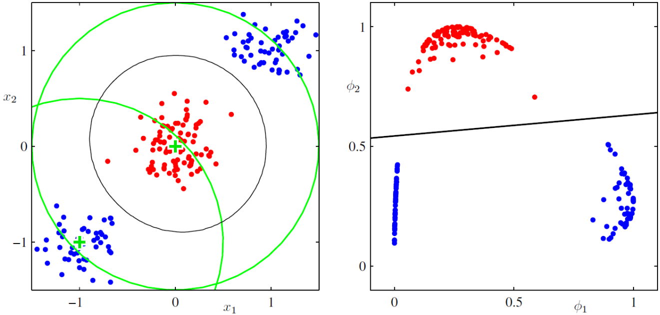
Illustration of Logistic Regression



Nonlinear Decision Boundary (Transformed Space $\phi(.)$)

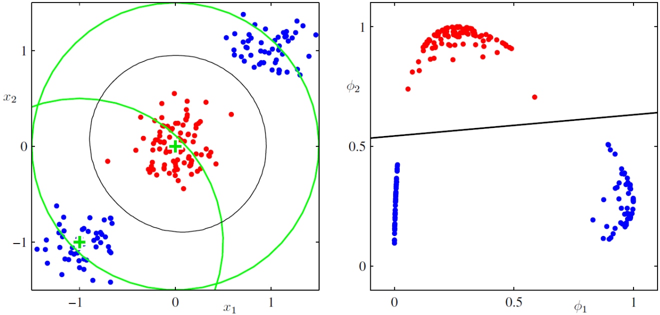


Nonlinear Decision Boundary (Transformed Space $\phi(.)$)



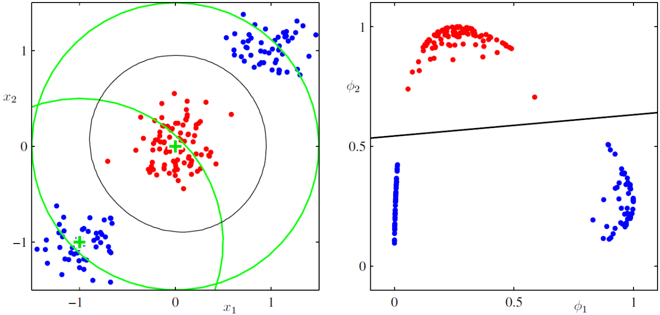
- Two Gaussian kernels are used to transform the data

Nonlinear Decision Boundary (Transformed Space $\phi(.)$)



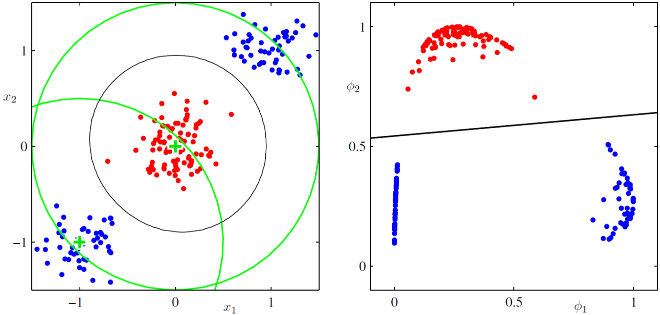
- Two Gaussian kernels are used to transform the data
- In general, we need to design the kernel ϕ from the data.

Nonlinear Decision Boundary (Transformed Space $\phi(.)$)



- Two Gaussian kernels are used to transform the data
- In general, we need to design the kernel ϕ from the data.
- DNN can be used to learn the optimal transformation from the data

Nonlinear Decision Boundary (Transformed Space $\phi(.)$)



- Two Gaussian kernels are used to transform the data
- In general, we need to design the kernel ϕ from the data.
- DNN can be used to learn the optimal transformation from the data
- Last layer of a DNN classifier performs logistic regression

Summary of Linear Classifiers

- Assumption: Classes are separable by linear hyperplanes

Summary of Linear Classifiers

- Assumption: Classes are separable by linear hyperplanes
- Linear discriminant functions model the separating hyperplanes
  - Least squares, Fisher discriminant, Perceptron, SVM (later)
  - May not work even if classes are separable because of outliers

Summary of Linear Classifiers

- Assumption: Classes are separable by linear hyperplanes
- Linear discriminant functions model the separating hyperplanes
  - Least squares, Fisher discriminant, Perceptron, SVM (later)
  - May not work even if classes are separable because of outliers
- Generative models estimate posteriors from priors and CCDs
  - ML or MAP estimators are employed to model CCD
  - Don't consider other class examples while estimating CCD

Summary of Linear Classifiers

- Assumption: Classes are separable by linear hyperplanes
- Linear discriminant functions model the separating hyperplanes
  - Least squares, Fisher discriminant, Perceptron, SVM (later)
  - May not work even if classes are separable because of outliers
- Generative models estimate posteriors from priors and CCDs
  - ML or MAP estimators are employed to model CCD
  - Don't consider other class examples while estimating CCD
- Discriminative models directly estimate posterior probabilities
  - Binary classes - logistic activation - binary cross entropy
  - Multiple classes - softmax activation - cross entropy
  - Rely on discriminative features - vulnerable to adversarial examples

Summary of Linear Classifiers

- Assumption: Classes are separable by linear hyperplanes
- Linear discriminant functions model the separating hyperplanes
  - Least squares, Fisher discriminant, Perceptron, SVM (later)
  - May not work even if classes are separable because of outliers
- Generative models estimate posteriors from priors and CCDs
  - ML or MAP estimators are employed to model CCD
  - Don't consider other class examples while estimating CCD
- Discriminative models directly estimate posterior probabilities
  - Binary classes - logistic activation - binary cross entropy
  - Multiple classes - softmax activation - cross entropy
  - Rely on discriminative features - vulnerable to adversarial examples
- If decision boundary is not linear, apply these techniques on $\phi(\mathbf{x})$
  - Neural networks offer *a way* of learning $\phi(\mathbf{x})$ from data

Thank You!