EECS 545: Machine Learning
Lecture 6. Classification 3

Honglak Lee
1/29/2025

Outline
(grey: already covered)

- Probabilistic discriminative models
  - ✓ Logistic Regression
  - ✓ Softmax Regression
- Probabilistic generative models
  - ✓ Gaussian discriminant analysis
  - ✓ Naive Bayes
- Discriminant functions (non-probabilistic)
  - Fisher's linear discriminant
  - Perceptron learning algorithm

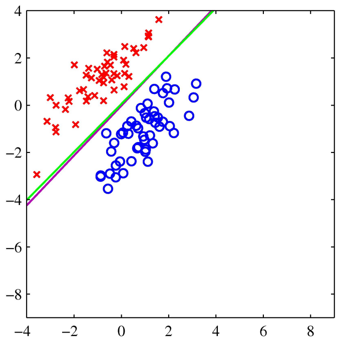
Discriminant Functions

Linear Discriminant functions: Discriminating two classes

- Specify a weight vector $\mathbf{w}$ and a bias w_0

$$h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$

- Assign $\mathbf{x}$ to C_1 if h(x) ≥ 0 and to C_0 otherwise.
- Q: How to pick $\mathbf{w}$?



Linear Discriminant functions: Discriminating K>2 classes

- Instead each class C_k gets its own function

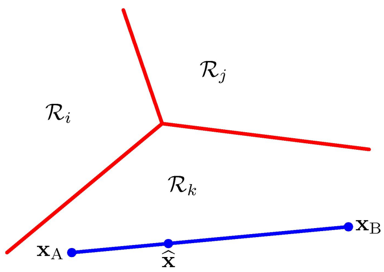
$$h_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k,0}$$

  – Assign $\mathbf{x}$ to C_k if

$$h_k(\mathbf{x}) > h_j(\mathbf{x}) \text{ for all } j \neq k$$

- The decision regions are convex polyhedra.

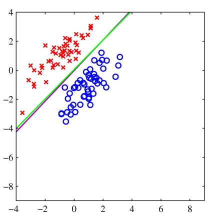
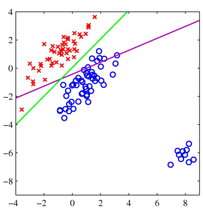
Decision Regions



- Decision regions are convex, with piecewise linear boundaries.

How do we set the weights w?

- How about $\mathbf{w}$ that minimizes squared error?
  – Label $\mathbf{y}$ versus linear prediction $h(\mathbf{w})$.
  – Least squares is too sensitive to outliers. (why?)

Read Bishop book

Learning Linear Discriminant Functions

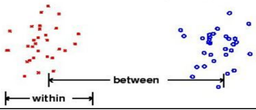
- Fisher's linear discriminant
- Perceptron learning algorithm

Fisher's Linear Discriminant

- Let's consider binary classification case.
- Use $\mathbf{w}$ to project $\mathbf{x}$ to one dimension.

$$\text{if} \quad \mathbf{w}^\top \mathbf{x} \geq -w_0 \quad \text{then} \quad C_1 \quad \text{else} \quad C_0$$

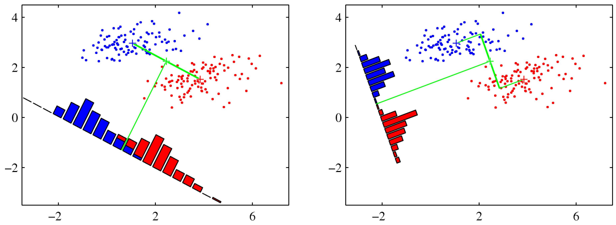
- Select w that best <u>separates</u> the classes.
- By "separating", the algorithm simultaneously
  - maximizes between-class (inter-class) variances
  - minimizes within-class (intra-class) variances



Read Bishop book

Fisher's Linear Discriminant

- Maximizing separation alone is not enough.
  - Minimizing class variance is a big help.



Read Bishop book

Objective function

- We want to maximize the "distance between classes"

$$\underbrace{m_2}_{\text{Projected mean}} - m_1 \equiv \mathbf{w}^\top(\underbrace{\mathbf{m}_2}_{\text{Mean}} - \mathbf{m}_1) \qquad \text{where } \mathbf{m}_k = \frac{1}{N_k}\sum_{n \in C_k} \mathbf{x}_n$$

- While minimizing the "distance within each class"

$$s_1^2 + s_2^2 \equiv \sum_{n \in C_1}(\mathbf{w}^\top \mathbf{x}_n - m_1)^2 + \sum_{n \in C_2}(\mathbf{w}^\top \mathbf{x}_n - m_2)^2$$

- Objective function: $J(\mathbf{w}) = \dfrac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$

Read Bishop book

Derivation of objective

- Numerator: $m_2 - m_1 \equiv \mathbf{w}^\top(\mathbf{m}_2 - \mathbf{m}_1)$

$$\|m_2 - m_1\|^2 = \mathbf{w}^\top \underbrace{(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\top}_{= S_B} \mathbf{w}$$

- Denominator:
  - $s_k^2 = \sum_{n \in C_k}(\mathbf{w}^\top \mathbf{x}_n - m_k)^2$
    $= \sum_{n \in C_k} \mathbf{w}^\top (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^\top \mathbf{w}$
  - $s_1^2 + s_2^2 = \mathbf{w}^\top \left[\underbrace{\sum_{k=1,2}\sum_{n \in C_k}(\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^\top}_{= S_W}\right] \mathbf{w}$
- After definition of terms, we get

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top S_B \mathbf{w}}{\mathbf{w}^\top S_W \mathbf{w}}$$

  - Solution: $\mathbf{w} \propto S_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$

Read Bishop book

Fisher's Linear Discriminant Analysis: Pros and Cons

Pros:

- Simple and effective approach for classification.
- Can effectively handle correlations between features
- Minimal assumptions about the underlying data distribution.
- Easy to interpret and explain

Cons:

- Only suitable for two-class classification problems
- Can be sensitive to outliers and may produce suboptimal results when the data has noisy features/labels

The Perceptron

- A "generalized linear function"

$$h(\mathbf{x}) = f(\mathbf{w}^\top \phi(\mathbf{x}))$$

where

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

- Uses target code: *y*=+1 for C_1, *y*=-1 for C_2.
- This means that we always want:

$$\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) y^{(n)} > 0$$

The Perceptron Criterion

- Only count errors from misclassified points:

$$E_P(\mathbf{w}) = -\sum_{\mathbf{x}^{(n)} \in \mathcal{M}} \mathbf{w}^\top \phi(\mathbf{x}^{(n)}) y^{(n)}$$

  - where $\mathcal{M}$ is the set of **misclassified** points.
- Stochastic gradient descent:
  - Update the weight vector according to the each misclassified sample (i.e., take gradient per sample):

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

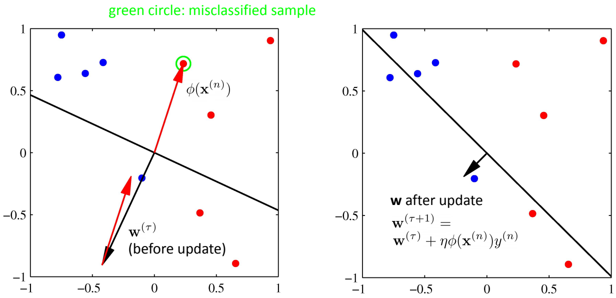
Note: update only for misclassified examples

Perceptron Learning (1)

red: positve (y= +1)
blue: negative (y= -1)

- If $\mathbf{x}^{(n)}$ is misclassified, add $\phi(\mathbf{x}^{(n)})$ into $\mathbf{w}$.

green circle: misclassified sample

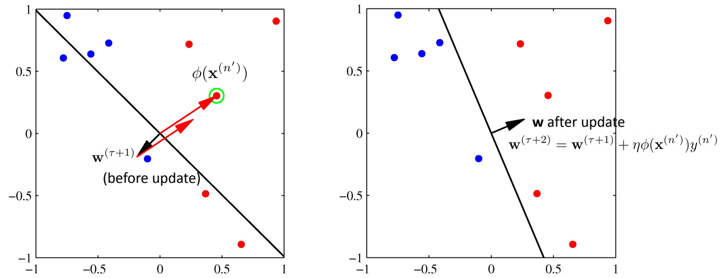


Perceptron Learning (2)

red: positve (y= +1)
blue: negative (y= -1)

- If $\mathbf{x}^{(n)}$ is misclassified, add $\phi(\mathbf{x}^{(n)})$ into $\mathbf{w}$.

green circle: misclassified sample



Perceptron Learning

- Perceptron Convergence Theorem (Block, 1962, and Novikoff, 1962):
  - If there exists an exact solution (i.e., if the training data is linearly separable)
  - then the learning algorithm will find it in a finite number of steps.
- Limitations of perceptron learning:
  - The convergence can be very slow.
  - If dataset is not linearly separable, it won’t converge.
  - Does not generalize well to K>2 classes.