Machine Learning

Lecture 04: Generative Models and Naive Bayes

Nevin L. Zhang
lzhang@cse.ust.hk

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

This set of notes is based on internet resources and
KP Murphy. Machine learning: a probabilistic perspective. MIT Press. (Chapters 3, 4, 8)
Andrew Ng. Lecture Notes on Machine Learning. Stanford.

Outline

Approaches to Classification

- Optimization approach:
  - Data $\rightarrow y = \text{sign}(z) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$.
  - Learn parameters by minimizing surrogate loss function:

$$\frac{1}{N}\sum_{i=1}^{N} L(y_i, z_i).$$

- Probabilistic approach:
  - Data $\rightarrow P(y|\mathbf{x}, \mathbf{w})$.
  - Learn parameters by minimizing (conditional) cross entropy:

$$-\frac{1}{N}\sum_{i=1}^{N} \log P(y_i|\mathbf{x}_i, w).$$

- Both models *directly* tell us what how to predict y from $\mathbf{x}$. Such models are called **discriminative models**.

Generative Approach

- Assume data generation process:

$$\begin{aligned} c &\sim P(y) \\ \mathbf{x} &\sim P(\mathbf{x}|y=c) \end{aligned}$$

- Learn parameters of the generation process from data:

$$\text{Data} \rightarrow p(y),\ p(\mathbf{x}|y)$$

by maximizing the loglikelihood,

$$\sum_{i=1}^{N} \log P(y_i, \mathbf{x}_i|\theta).$$

Or, equivalently, minimizing (joint) cross entropy:

$$-\frac{1}{N}\sum_{i=1}^{N} \log P(y_i, \mathbf{x}_i|\theta).$$

Question: What are the two distributions involved in the cross entropy?

Generative Learning

- To learn $p(y = c)$ ($c \in \{1, 2, \ldots, C\}$) is to determine the sizes of the classes. They are called **prior probabilities** of the classes.
- To learn $p(\mathbf{x}|y = c)$ ($c \in \{1, 2, \ldots, C\}$) is to determine the characteristics of the classes.
  - Here, we need to assume that features $\mathbf{x}$ follow certain distributions, e.g., Gaussian. In other words, we assume the data are **generated** from Gaussian distributions.
  - To learn $p(\mathbf{x}|y = c)$ is to determine the parameters of the **generative model**. A common way to do this is to use MLE.
  - $p(\mathbf{x}|y = c)$ ($c \in \{1, 2, \ldots, C\}$) are called **class conditional densities**.

Generative Learning

How to predict y from $\mathbf{x}$ using $p(y)$ and $p(\mathbf{x}|y)$?

- Use Bayes rule get the **posterior distribution**:

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})}$$

- Rule for classification:

$$\hat{y} = \arg\max_y p(y|\mathbf{x}) = \arg\max_y p(y)p(\mathbf{x}|y) = \arg\max_y [\log p(y) + \log p(\mathbf{x}|y)]$$

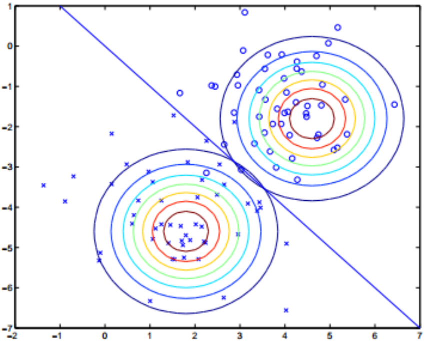
Outline

Gaussian Discriminant Analysis

- Gaussian Discriminant Analysis (GDA) is for applications with real-valued input features $\mathbf{x}$. It assumes that the class conditional densities are Gaussian:

$$p(\mathbf{x}|y = c, \theta) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c).$$

Let $\pi_c = P(y = c)$ and $\pi = (\pi_1, \ldots, \pi_C)^\top$. All parameters of GDA are: $\theta = \{\pi_c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c | c = 1, \ldots, C\}$



GDA: Parameter Estimation

- The log-likelihood function is as follows:

$$
\begin{aligned}
\log p(\mathcal{D}|\theta) &= \sum_{i=1}^{N} \log p(\mathbf{x}_i, y_i|\theta) \\
&= \sum_{i=1}^{N} [\log p(y_i|\theta) + \log p(\mathbf{x}_i|y_i, \theta)] \\
&= \sum_{i=1}^{N} \sum_{c=1}^{C} \mathbf{1}(y_i = c) [\log p(y = c|\theta) + \log p(\mathbf{x}_i|y = c, \theta)] \\
&= \sum_{c=1}^{C} \sum_{i:y_i=c} \log \pi_c + \sum_{c=1}^{C} \sum_{i:y_i=c} \log \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \\
&= \sum_{c=1}^{C} n_c \log \pi_c + \sum_{c=1}^{C} \sum_{i:y_i=c} \log \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)
\end{aligned}
$$

where $n_c = \sum_{i=1}^{N} \mathbf{1}(y_i = c)$ is the size of class c.

GDA: Parameter Estimation

- To maximize $\log p(\mathcal{D}|\theta)$, we can separately maximize
  - $\sum_{c=1}^{C} n_c \log \pi_c$, and
  - $\sum_{i:y_i=c} \log \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ for different c.
- By Gibbs' inequality,

$$\sum_{c=1}^{C} n_c \log \pi_c \leq \sum_{c=1}^{C} n_c \log \frac{n_c}{N}$$

where $N = \sum_c n_c$. Hence, the MLE of π_c is:

$$\hat{\pi}_c = \frac{n_c}{N}$$

GDA: Parameter Estimation

- The MLE $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ are as follows:

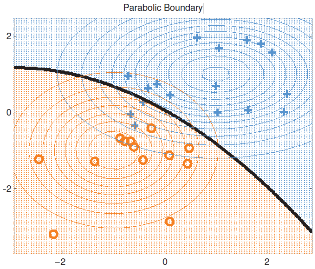
$$
\begin{aligned}
\hat{\boldsymbol{\mu}_c} &= \frac{1}{n_c} \sum_{i:y_i=c} \mathbf{x}_i \quad \text{(sample mean)} \\
\hat{\boldsymbol{\Sigma}}_c &= \frac{1}{n_c} \sum_{i:y_i=c} (\mathbf{x}_i - \hat{\boldsymbol{\mu}_c})(\mathbf{x}_i - \hat{\boldsymbol{\mu}_c})^\top \quad \text{(sample covariance matrix)}
\end{aligned}
$$

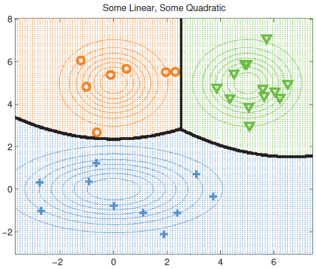
GDA: Classification Rule and Decision Boundary

- The classification rule of GDA is:

$$\begin{aligned}\hat{y} &= \arg\max_c [\log p(y=c) + \log p(\mathbf{x}|y=c)] \\ &= \arg\max_c \left[\log(\pi_c) - \frac{D}{2}\log(2\pi_c) - \frac{1}{2}\log(|\boldsymbol{\Sigma}_c|) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-1}(\mathbf{x}-\mu_c)\right]\end{aligned}$$

- This is a quadratic function. So, DGA in this case is also called **quadratic discriminant analysis**. The decision boundary is generally parabolic, but can also be linear.





Relationship with Softmax Regression

If $\Sigma_c = \Sigma$ for all c:

$$\begin{aligned}
p(y=c|\mathbf{x},\theta) &\propto p(y=c|\theta)p(\mathbf{x}|y=c,\theta) \qquad \text{(Bayes rule)} \\
&\propto \pi_c \exp\left[-\frac{(\mathbf{x}-\boldsymbol{\mu}_c)^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_c)}{2}\right] \\
&= \pi_c \exp\left[\boldsymbol{\mu}_c^\top \Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_c^\top \Sigma^{-1}\boldsymbol{\mu}_c\right] \exp\left[-\frac{1}{2}\mathbf{x}^\top \Sigma^{-1}\mathbf{x}\right] \\
&\propto \exp\left[\boldsymbol{\mu}_c^\top \Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_c^\top \Sigma^{-1}\boldsymbol{\mu}_c + \log \pi_c\right]
\end{aligned}$$

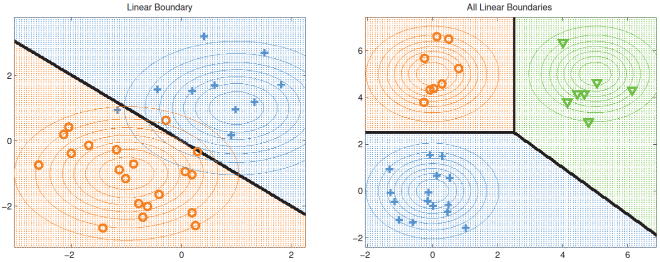
- Here, terms not depending on c are removed in steps 2 and 4.
- Let $b_c = -\frac{1}{2}\boldsymbol{\mu}_c^\top \Sigma^{-1}\boldsymbol{\mu}_c + \log \pi_c$, and $\mathbf{w}_c^\top = \boldsymbol{\mu}_c^\top \Sigma^{-1}$. Then

$$p(y=c|\mathbf{x},\theta) \quad \propto \quad \exp\left[\mathbf{w}_c^\top \mathbf{x} + b_c\right]$$

- This leads us back to the **softmax regression model**, and when $C=2$, the **logistic regress model**.

Relationship with Softmax Regression

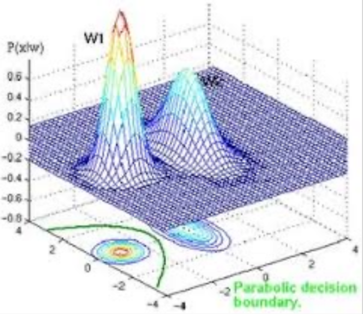
- The decision boundary is therefore linear:

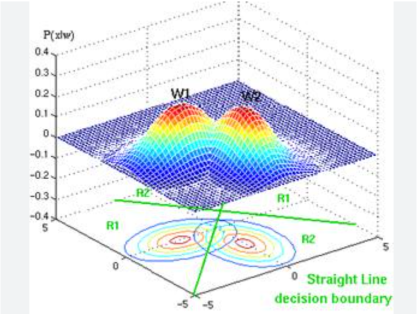


- So, in this case, Gaussian discriminant analysis becomes **linear discriminant analysis**.

Relationship with Softmax Regression

GDA is equivalent to Softmax when $\Sigma_c = \Sigma$ for all c:

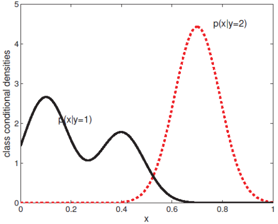


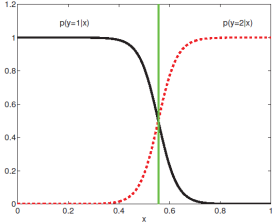


Outline

Gaussian Discriminant Analysis (GDA) vs Logistic Regression

- GDA makes stronger assumptions than logistic regression
  - $p(\mathbf{x}|y)$ is Gaussian (with shared Σ) implies that $p(y|\mathbf{x})$ is a logistic function $\sigma(\mathbf{w}^\top \mathbf{x})$.
  - The opposite is not true. Here is a counter-example.





- For parameter estimation, logistic regression maximizes the conditional log-likelihood $\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w})$, while GDA maximizes the joint log-likelihood $\sum_{i=1}^{N} \log p(y_i, \mathbf{x}_i|\theta)$.

GDA vs Logistic Regression

- Parameter estimation is easier in generative classifiers than in discriminative classifiers. For example, GDA has closed-form formulae for MLE, while logistic regression requires gradient descent to compute MLE.
- When the Gaussian assumptions made by GDA are correct, the GDA will need less training data than logistic regression to achieve a certain level of performance.
- In contrast, by making significantly weaker assumptions, logistic regression is more robust and less sensitive to incorrect modeling assumptions.

Generative vs Discriminative Classifiers

- It is easier to deal with missing data with generative classifiers: Use the EM algorithm during training and marginalization during testing. No principled way to handle missing data with discriminative classifiers.
- In generative classifiers, we can use unlabeled data to help with the training. There is a subfield called **semi-supervised learning** that does this. It is much harder to do in discriminative classifiers.
- In discriminative classifiers, we can do basis function expansion, replacing $\mathbf{x}$ with some $\phi(\mathbf{x})$. This is hard to do with generative models because the new feature are correlated in complex ways.

Outline

Problem Statement

- So far, we have been considering training set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, where $y_i \in \{1, 2, \ldots, C\}$ and **continuous-valued features** $\mathbf{x}_i \in \mathbb{R}^D$. We assume $p(\mathbf{x}|y = c, \theta) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$. The number of parameters for each class is $D + \frac{D(D+1)}{2}$.
- Next, we consider problems with **discrete-valued features** $\mathbf{x}_i \in \{1, 2, \ldots, K\}^D$
  - For each class, we have the joint distribution $p(\mathbf{x}|y = c, \theta) = p(x_1, \ldots, x_D|y = c, \theta)$.
  - When all features are binary, the number of parameters is $2^D - 1$. This is too many to handle. We need to reduce the number of parameter by making independence assumptions.

Naive Bayes Model

- In the **Naive Bayes model**, we assume that the features are **conditionally independent** of each other given the class label

$$p(\mathbf{x}|y=c,\theta) = \prod_{j=1}^{D} p(x_j|y=c,\theta_{jc})$$

- In the case of binary features ($K=2$), we can use the Bernoulli distribution for each feature:

$$p(\mathbf{x}|y=c,\theta) = \prod_{j=1}^{D} Ber(x_j|\mu_{jc})$$

  where μ_{jc} is the probability that feature j occurs in class c.

- In the general case ($K>2$), we can use the categorical distribution for each feature:

$$p(\mathbf{x}|y=c,\theta) = \prod_{j=1}^{D} Cat(x_j|\boldsymbol{\mu}_{jc})$$

  where $\boldsymbol{\mu}_{jc} = (\mu_{jc1},\ldots,\mu_{jcK})^\top$ where μ_{jck} is the probability that x_j takes value k in class c.

Naive Bayes: Parameter Estimation

- The log-likelihood function is as follows:

$$
\begin{aligned}
\log p(\mathcal{D}|\theta) &= \sum_{i=1}^{N} \log p(\mathbf{x}_i, y_i|\theta) \\
&= \sum_{i=1}^{N} [\log p(y_i|\theta) + \log p(\mathbf{x}_i|y_i, \theta)] \\
&= \sum_{i=1}^{N} \sum_{c=1}^{C} \mathbf{1}(y_i = c) [\log p(y = c|\theta) + \log p(\mathbf{x}_i|y = c, \theta)] \\
&= \sum_{c=1}^{C} \sum_{i:y_i=c} \log \pi_c + \sum_{c=1}^{C} \sum_{i:y_i=c} \log \prod_{j=1}^{D} Cat(x_{ij}|\boldsymbol{\mu}_{jc}) \\
&= \sum_{i=1}^{N} \sum_{c=1}^{C} \mathbf{1}(y_i = c) \log \pi_c + \sum_{c=1}^{C} \sum_{j=1}^{D} \sum_{i:y_i=c} \log p(x_{ij}|\boldsymbol{\mu}_{jc})
\end{aligned}
$$

Naive Bayes: Parameter Estimation

- To maximize $\log p(\mathcal{D}|\theta)$, we can separately maximize
  - $\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbf{1}(y_i=c)\log\pi_c$, and
  - $\sum_{j=1}^{D}\sum_{i:y_i=c}\log p(x_{ij}|\boldsymbol{\mu}_{jc}))$ for different c.
- As in the case of GDA,

$$\hat{\pi}_c = \frac{n_c}{N} \quad (n_c = \textstyle\sum_{i=1}^{N}\mathbf{1}(y_i=c) \text{ is the size of class } c)$$

- By Gibbs' inequality,

$$\sum_{j=1}^{D}\sum_{i:y_i=c}\log p(x_{ij}|\boldsymbol{\mu}_{jc})) = \sum_{j=1}^{D} n_{jck}\log p(x_{ij}|\boldsymbol{\mu}_{jc})) \leq \sum_{j=1}^{D} n_{jck}\log\frac{n_{jck}}{n_c}$$

Where n_{jck} is the number of examples in class c where $x_{ij}=k$. Hence,

$$\hat{\mu}_{jck} = \frac{n_{jck}}{n_c}$$

Laplace Smoothing

- In practice, we might encounter the case where $n_c = 0$. In such a case, $\hat{\mu}_{jck} = 0/0$.
- **Laplace smoothing** is used t avoid the division-by-zero problem:

$$\hat{\mu}_{jck} = \frac{n_{jck} + \alpha}{n_c + K\alpha}$$

  where $\alpha > 0$ is the **smoothing parameter**.
- The Naive Bayes algorithm is still considered very good, and very popular.

Independence Assumption for Continuous-Valued Data

- For continuous-value data, we can also make the conditional independence assumption

$$p(\mathbf{x}|y = c, \theta) = \prod_{j=1}^{D} p(x_j|y = c, \theta_{jc})$$

- If we further assume x_j follows a Gaussian distribution, we get

$$p(\mathbf{x}|y = c, \theta) = \prod_{j=1}^{D} \mathcal{N}(x_j|\mu_{jc}, \sigma_{jc}^2)$$

  This is equivalent to Gaussian discriminant analysis with diagonal covariance matrix.

- The independence assumption reduces the number of parameters for each class from $D + \frac{(D+1)D}{2}$ to $2D$.