Probabilistic Classification Algorithms

Tseng-Ching James Shen

Probabilistic Classifiers

- Bayes Classifier
- Logistic Regression

Bayes' Theorem

Likelihood

Prior Probability

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})}.$$

Posterior Probability

Bayes Classifier

- Given the data point $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_d)$, the classifier calculates the probability that it belongs to class C_i (I = 1, 2, … M)

$$p(\boldsymbol{t} = C_i|\boldsymbol{x})$$

- According to the Bayes' theorem, the classifier calculates the conditional probability

$$p(\boldsymbol{t} = C_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{t}=C_i)\, p(\boldsymbol{t}=C_i)}{p(\boldsymbol{x})}$$

$$p(\boldsymbol{x}) = \Sigma_i\, p(\boldsymbol{x}|\boldsymbol{t} = C_i)\, p(\boldsymbol{t} = C_i)$$

Bayes Classifier (Cont.)

- Use the training set $\boldsymbol{X}_{tr}$, $\boldsymbol{t}_{tr}$ to estimate the prior $p(\boldsymbol{t} = C_i)$ and likelihood probability $p(\boldsymbol{x}|\boldsymbol{t} = C_i)$
- Two ways to estimate the prior probability
  - Uniform prior

$$p(\boldsymbol{t} = C_i) = \frac{1}{N}$$

  - Class size prior

$$p(\boldsymbol{t} = C_i) = \frac{N_i}{N}$$

Estimation of Likelihood Probability

- Use the following Gaussian class-conditional distribution to derive the likelihood probability for C_i from the training dataset

$$\mathcal{N}(\mu_i, \Sigma_i)$$

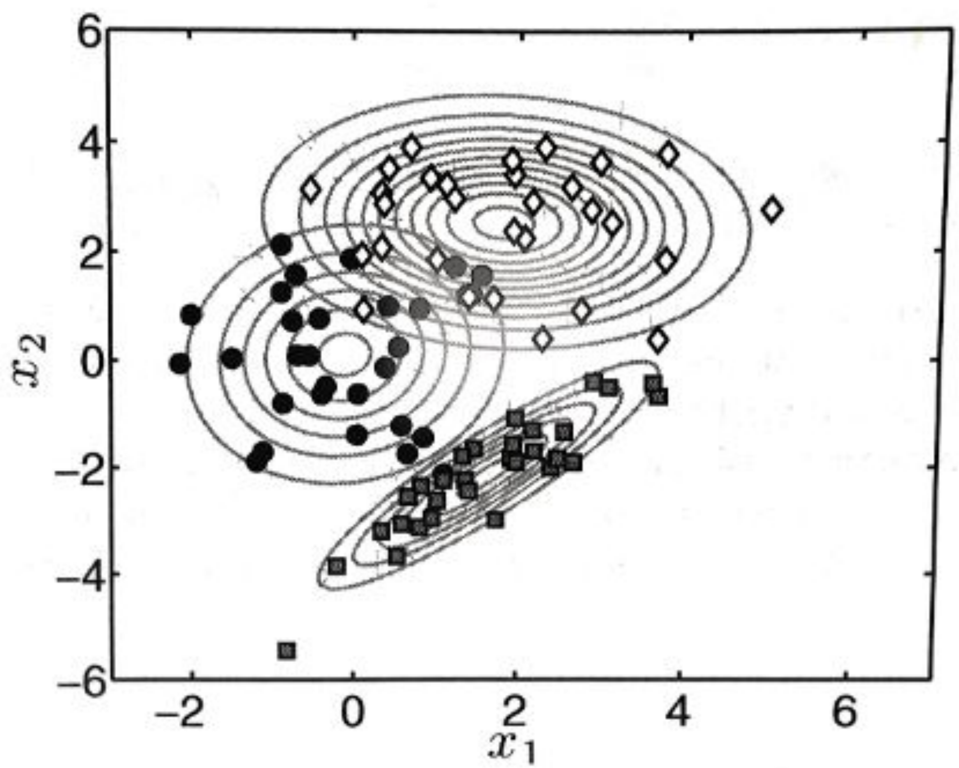
where

$$\mu_i = \frac{1}{N_i} \sum_n x_{\mathrm{tr},\, n}$$

$$\Sigma_i = \frac{1}{N_i} \sum_n (x_{tr,n} - u_i)\left(x_{tr,n} - u_i\right)^{\mathrm{T}}$$

Bayes Classifier Example

Three classes generated with gaussian distribution



Bayes Classifier Example (cont.)

Make prediction

TABLE 5.1 Likelihood and priors for $\mathbf{x}_{\text{new}} = [2, 0]^{\mathsf{T}}$ for the Gaussian class-conditional Bayesian classification example.

c	$p(\mathbf{x}_{\text{new}} \mid T_{\text{new}} = c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$	$P(T_{\text{new}} = c \mid \mathbf{X}, \mathbf{t})$	$p(\mathbf{x}_{\text{new}} \mid T_{\text{new}} = c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) P(T_{\text{new}} = c \mid \mathbf{X}, \mathbf{t})$
1	0.0138	$\frac{1}{3}$	0.0046
2	0.0061	$\frac{1}{3}$	0.0020
3	0.0002	$\frac{1}{3}$	0.0001

Naïve-Bayes Classifier

- Assumes that all features of the dataset points are random variables $\boldsymbol{x}_i$ which are independent
- The likelihood probability becomes the product form:

$$p(\boldsymbol{x}|\boldsymbol{t} = C_i) = \prod_{j=1}^{d} p(\boldsymbol{x}_j|\boldsymbol{t} = C_i)$$

- We just need to estimate $p(\boldsymbol{x}_j|\boldsymbol{t} = C_i)$ using the training set for j = 1, 2, …, d

An example

- Compute all probabilities required for classification

A	B	C
m	b	t
m	s	t
g	q	t
h	s	t
g	q	t
g	q	f
g	s	f
h	b	f
h	q	f
m	b	f

$\Pr(C = t) = 1/2,$ $\Pr(C = f) = 1/2$

$\Pr(A=m \mid C=t) = 2/5$ $\Pr(A=g \mid C=t) = 2/5$ $\Pr(A=h \mid C=t) = 1/5$

$\Pr(A=m \mid C=f) = 1/5$ $\Pr(A=g \mid C=f) = 2/5$ $\Pr(A=h \mid C=n) = 2/5$

$\Pr(B=b \mid C=t) = 1/5$ $\Pr(B=s \mid C=t) = 2/5$ $\Pr(B=q \mid C=t) = 2/5$

$\Pr(B=b \mid C=f) = 2/5$ $\Pr(B=s \mid C=f) = 1/5$ $\Pr(B=q \mid C=f) = 2/5$

Now we have a test example:

$A = m \quad B = q \quad C = ?$

An Example (cont …)

- For C = t, we have

$$\Pr(C=t)\prod_{j=1}^{2}\Pr(A_j=a_j \mid C=t)=\frac{1}{2}\times\frac{2}{5}\times\frac{2}{5}=\frac{2}{25}$$

- For class C = f, we have

$$\Pr(C=f)\prod_{j=1}^{2}\Pr(A_j=a_j \mid C=f)=\frac{1}{2}\times\frac{1}{5}\times\frac{2}{5}=\frac{1}{25}$$

- C = t is more probable. t is the final class.

Logistic Regression

- Binary classification i.e. only two classes
- Use the following function to model the posterior distribution

$$p(y = 1|\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^T\boldsymbol{x})}$$

- We need to estimate the parameters ***w*** using the training set ***X***, ***y***
- Parameters ***w*** can be estimated as a solution of an optimization problem

Optimization Problem

- Use *Kullback-Leibler (KL) divergence* to measure the distance between two density functions, which is also called Training Loss

$$\frac{1}{N}\sum_{i=1}^{N} \ln \frac{p(y_i=0|\mathbf{x}_i)}{p(y_i=0 \mid \mathbf{x}_i, \mathbf{w})} + \frac{1}{N}\sum_{i=1}^{N} \ln \frac{p(y_i=1|\mathbf{x}_i)}{p(y_i=1 \mid \mathbf{x}_i, \mathbf{w})}$$

- The goal is to find the optimal **w** which will minimize the training loss
- Minimizing the training loss is equivalent to *maximizing the likelihood* $p(y_1=k_1, y_2=k_2, \ldots, y_N=k_N \mid \mathbf{x}_i, \mathbf{w})$ which can be expressed as

Log-likelihood Objective Function

- Likelihood function

$$L(\boldsymbol{w}, \boldsymbol{X}, \boldsymbol{y}) = \prod_{i=1}^{N} \left(\frac{1}{1+\exp(-\boldsymbol{w}^T\boldsymbol{x}_i)}\right)^{yi} \left(1- \frac{1}{1+\exp(-\boldsymbol{w}^T\boldsymbol{x}_i)}\right)^{(1-yi)}$$

- Take the natural log on both side to get the log-likelihood function

$$\log L(\boldsymbol{w}) = \sum_{i=1}^{n}\left[y^{(i)}\log\left(\phi\left(z^{(i)}\right)\right)+\left(1-y^{(i)}\right)\log\left(1-\phi\left(z^{(i)}\right)\right)\right]$$

- We will find ***w*** which minimizes the function

$$J(\boldsymbol{w}) = \sum_{i=1}^{n}\left[-y^{(i)}\log\left(\phi\left(z^{(i)}\right)\right)-\left(1-y^{(i)}\right)\log\left(1-\phi\left(z^{(i)}\right)\right)\right]$$

Single-instance Example

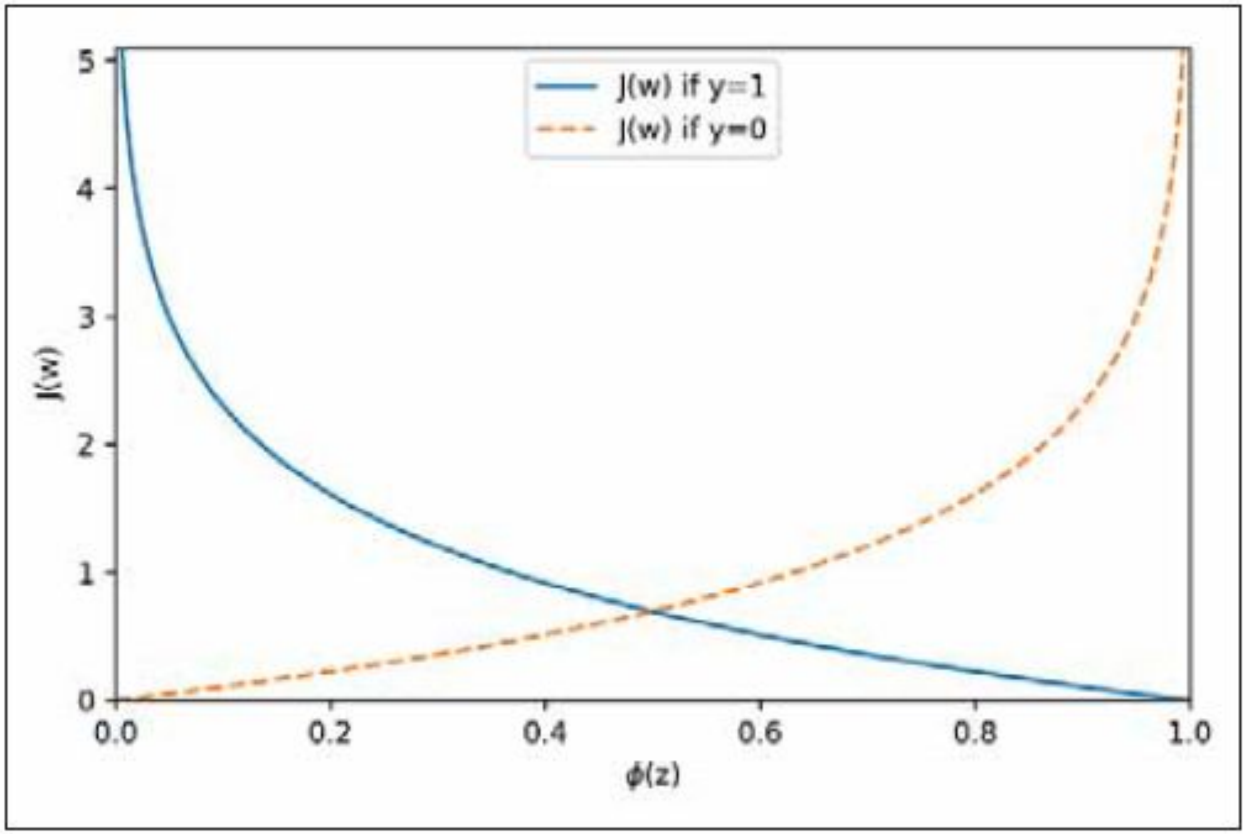
- The objective function becomes

$$J(\phi(z), y; \boldsymbol{w}) = -y\log(\phi(z)) - (1-y)\log(1-\phi(z))$$

$$J(\phi(z), y; \boldsymbol{w}) = \begin{cases} -\log(\phi(z)) & \text{if } y=1 \\ -\log(1-\phi(z)) & \text{if } y=0 \end{cases}$$

- Log-likelihood function is a convex function of ***w***
  - Lesson5_ConvexAnalysis.pdf (ens.fr)
  - machine learning - Logistic regression - Prove That the Cost Function Is Convex - Mathematics Stack Exchange

Single-instance Example (cont.)



Gradient Descent Solution

- Log-likelihood function is a convex function
- Gradient descent approach can be used to find the optimal solution
- Derivation of gradient

$$w_j := w_j + \eta \sum_{i=1}^{n} \left(y^{(i)} - \phi\left(z^{(i)} \right) \right) x_j^{(i)}$$

- Gradient descent algorithm iteratively update ***w*** using the above equation

Multi-class Logistics Regression

- Assume there are *K* classes (class 1, 2, ..., K)
- The posterior probability is modeled as a Softmax function

$$p(y = k|\boldsymbol{x}) = \frac{\exp(-\boldsymbol{w}_k{}^T\boldsymbol{x})}{\sum_{j=1}^{K}\exp(-\boldsymbol{w}_j{}^T\boldsymbol{x})}$$

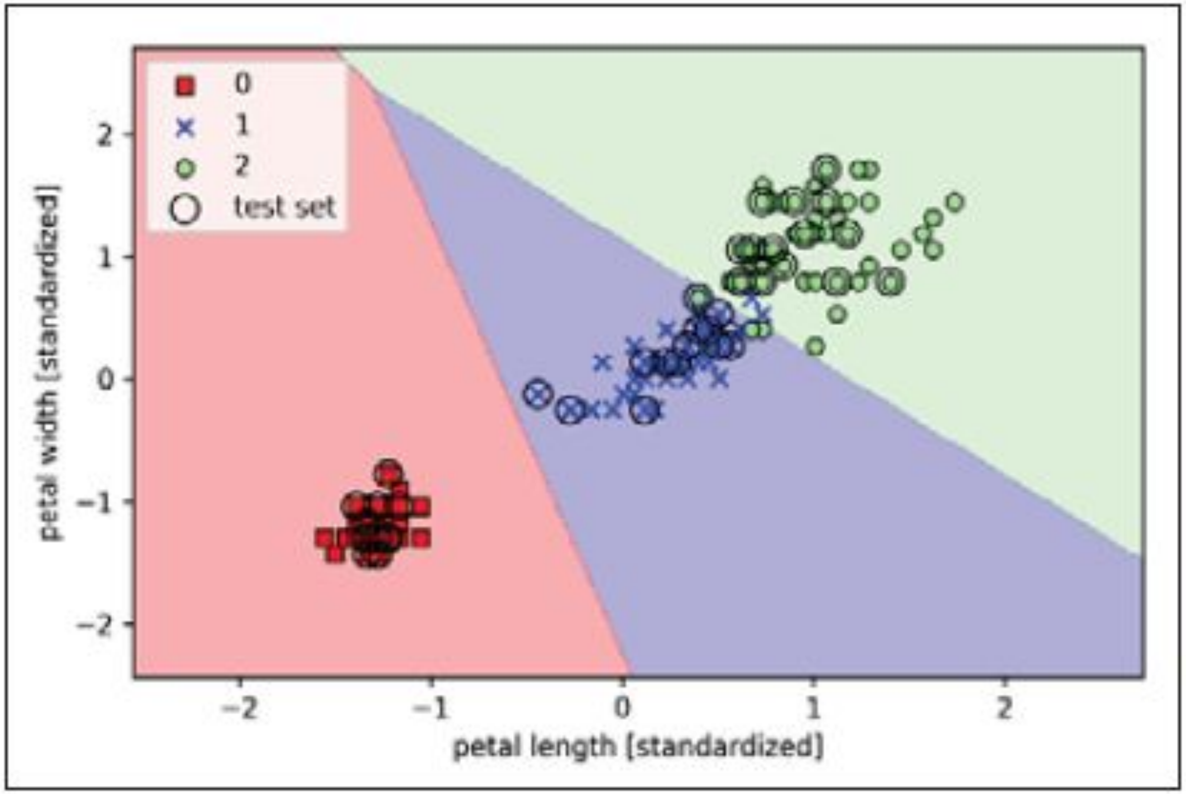
  $\boldsymbol{w}_j$ is the parameters for class j
- Follow the similar steps for binary class case, we can define the log-likelihood (*cross entropy*) function, and derive the gradient (but a little more complicated)
  - ➢ [MultiClass Logistic Regression]()

Scikit Learn Example

- Apply Logistic Regression on Irish dataset (three classes)

```
>>> from sklearn.linear_model import LogisticRegression
>>> lr = LogisticRegression(C=100.0, random_state=1)
>>> lr.fit(X_train_std, y_train)
>>> plot_decision_regions(X_combined_std,
...                       y_combined,
...                       classifier=lr,
...                       test_idx=range(105, 150))
>>> plt.xlabel('petal length [standardized]')
>>> plt.ylabel('petal width [standardized]')
>>> plt.legend(loc='upper left')
>>> plt.show()
```

Scikit Learn Example (cont.)



Scikit Learn Example (cont.)

```
>>> lr.predict_proba(X_test_std[:3, :])
```

This code snippet returns the following array:

```
array([[  3.20136878e-08,   1.46953648e-01,   8.53046320e-01],
       [  8.34428069e-01,   1.65571931e-01,   4.57896429e-12],
       [  8.49182775e-01,   1.50817225e-01,   4.65678779e-13]])
```

```
>>> lr.predict(X_test_std[:3, :])
array([2, 0, 0])
```