Pattern Informatics Report

Le Trung Kien
03-120291, 3rd Year
Department of Mechano-Informatics
The University of Tokyo

February 17, 2013

Note

- All programs are tested on Ubuntu 64 bit with eigen installed. They should work fine on ubuntu 32 bit as well.
- Source codes can be accessed via github:

  `https://github.com/letrungkien211/Pattern.git`
- Compile by using Makefile.

Problem 1+2:Widrow-Hoff Algorithm & Pseudo-Inverse Matrix

Widrow-Hoff Algorithm based on gradient descent method to update model's weight as following:

$$W_{new} = W_{old} - \alpha W_{old}(XW_{old} - T)$$

In which, α is learning rate, W is weight, X is training data, T is training label. The training will stop when error reaches a threshold or number of iterations exeeds limit. In my implementation, learning rate is 0.1, threshold is 0.001 and max number of iterations is 100.
On the other hand, optimal value can be directly calculated by using pseudo-inverse matrix:

$$W = (X^T X)^{-1} X^T T$$

Results after using the two above algorithms are shown in Figure 1.

According to Figure 1, the two weights trained by the two algorithms are almost the same, which is not obvious fact. In this case, the two classes are linearly separable, so the two algorithms should lead to very simal final weight. Still, they are slightly different because the weight calculated by pseudo-inverse matrix is deterministic while the one trained by Willow-Hoff depends on numerous factors such as intial values, error's threshold and number of iterations. Moreover, Willow-Hoff algorithm is always a valid method, but when pseudo-inverse matrix of X does not exist, we cannot use it to find the optimal weight for classification.

An example of running the program (Willow-Hoff and Pseudo-Inverse in order):

```
@:~/Pattern$ ./2 1
0.642855
0.223755
-0.412262
Precision's rate: 0.875
@:~/Pattern$ ./2 2
0.644933
0.223556
-0.412747
Precision's rate: 0.875
```

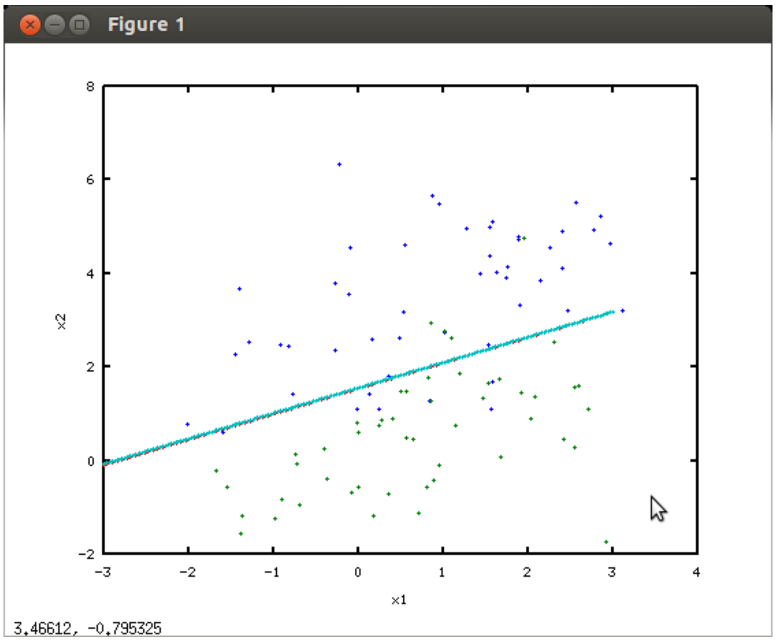

Figure 1: Classification by Widrow-Hoff algorithm and pseudo-inverse matrix.

Problem 3: Leave-one-out Method

In order to choose the best hyper-parameter k for k-nereast neighbourhood, the result after using leave-one-out method is shown in Figure 2. The optimal k is 3, and precision rate for the test set is 85%.

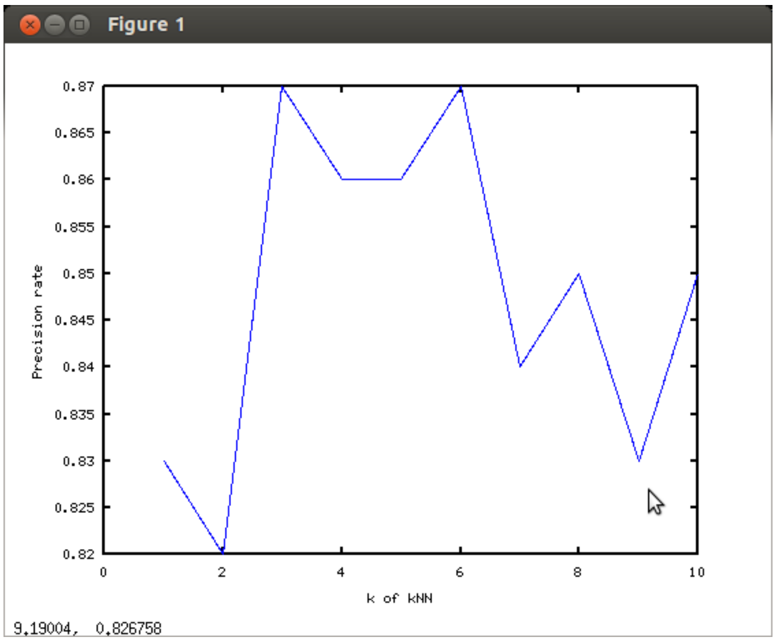

Figure 2: Precision rate and k (of kNN) relationship.

Problem 4: Mahalanobis Distance

Mahalanobis distance is calculated as following:

$$d(x, \mu, \Sigma) = \sqrt{(x - \mu)\Sigma^{-1}(x - \mu)}$$

In which, x is a point in space, μ, Σ is center and covariance matrix of a cluster of points, respectively. Moreover, μ, Σ are estimated by:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\Sigma = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T$$

Running the program for 4th problem, all distances from test points to cluster of points are shown in Table 1. When $P(\omega_1) = P(\omega_2) = P(\omega_3) = \frac{1}{3}$, classified data are shown in Table 1 due to closest distance

Test	ω_1	ω_2	ω_3	Classify
1	1.01497	0.858051	2.67476	ω_2
2	1.55714	1.75568	0.647009	ω_3
3	0.489962	0.268432	2.2415	ω_2
4	0.487237	0.451834	1.46234	ω_2

Table 1: Mahalanobis Distance.

from a point to a cluster. However, when $P(\omega_1) = 0.8, P(\omega_2) = P(\omega_3) = 0.1$, we have to use the following formula:

$$P(\omega \mid x) = P(x \mid \omega)P(\omega)$$

Results from normalizing vector of inversed distances from a point x to each clusters of points $\omega_1, \omega_2, \omega_3$. As we can see, classifying results are completely changed when prior probabilities are different.

Test	$P(\omega_1 \mid x)$	$P(\omega_2 \mid x)$	$P(\omega_3 \mid x)$	Classify
1	0.312214	0.046164	0.0148092	ω_1
2	0.186324	0.0206567	0.0560527	ω_1
3	0.262825	0.0599657	0.00718123	ω_1
4	0.331738	0.0447163	0.0138165	ω_1

Table 2: Classifying test data.