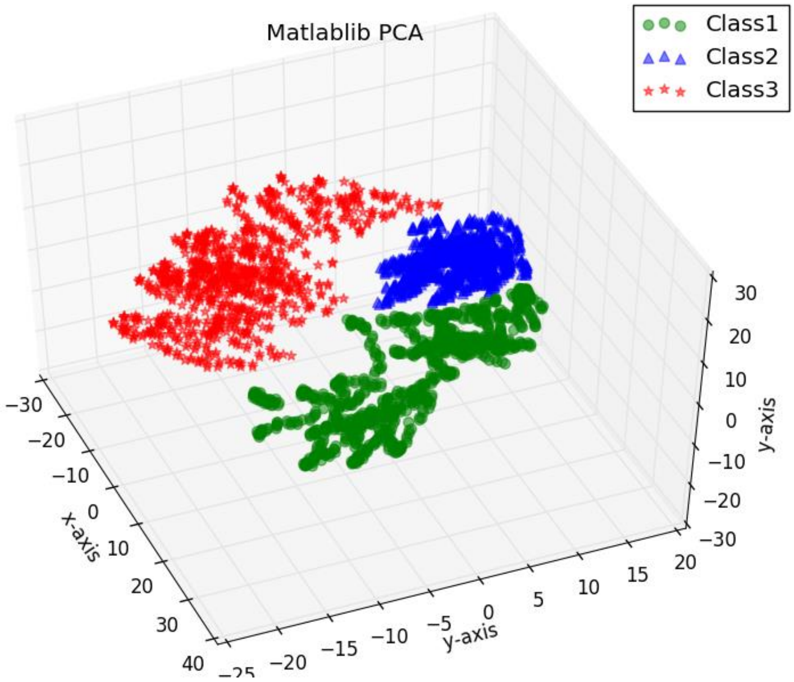
Report

Homework 2

0550193 陳昱瑋

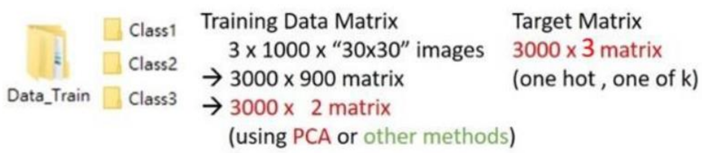
2017/4/11



X_train is a 3000x900 matrix (for 3 class, 1000 for each class). Every row contains the gray-level picture we want to classified.

T_train is a 3000x3 matrix. 3000 rows of data and 3 classes of objects.

Phi_train is a 3000x2 matrix according to the spec



What you can notice that the class 1 is sparser than class 2 or 3. We can let class 1 be the bias dataset to have better solution? I will verify the idea in bias-training part

Probabilistic Generative Model

To get generative mode, we need prior and conditional probability to get the posterior distribution.

For multi-class problems,

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \end{aligned}$$

softmax function

where $a_k = \ln p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$

Supposed conditional density is Gaussian, and the COV-matrix is identical across classes, maximum likelihood of $p(\mathbf{x}|C_k)$ can be written as

$$\mathrm{p}(\mathrm{x}|\mathrm{C_k}) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma|^{1/2}}\exp\{-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\}$$

. we can simplify the equation with formula in class note p.20

$$\mathrm{p}(C_k|\mathrm{x}) = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_k)p(C_k)} = \frac{\exp(a_k)}{\sum_j \exp(a_k)}$$, where

For two classes case

$$a_k = \ln p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$$

For multi-class case

$$a_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$$

$$\begin{aligned} \mathbf{w}_k &= \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \\ w_{k0} &= -\frac{1}{2}\boldsymbol{\mu}_k^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln p(\mathcal{C}_k) \end{aligned}$$

- Result

N: 3000(# of datasets)

M: 900(# of feature of bmp graph; picture vector)

Training dataset number: 2400

Testing dataset number: 600

If I use all the features (900) I have to plot out the decision region, the plot will be like the first page has shown. As for the accuracy calculation, instead of doing error calculate, I chose to cross validation my model by taking out 1/5 of data and do it for five iterations to get average performance.

- For 900 features:

Cross	0	1	2	3	4	Average
Class1	100.0	100.0	100.0	100.0	100.0	100.0
Class2	100.0	100.0	100.0	100.0	100.0	100.0
Class3	100.0	100.0	100.0	100.0	100.0	100.0

- For 5 features: (PCA)

Class	0	1	2	3	4	Average
Class1	87.5	84.0	92.0	90.5	83.5	87.5
Class2	10.0	5.0	4.0	3.5	8.5	6.2
Class3	72.5	79.5	74.5	82.5	82.5	78.3

- For 2 features: (PCA)

Class	0	1	2	3	4	Average
Class1	0.0	0.0	0.0	0.0	0.0	0
Class2	19.5	15.5	21.5	21.5	22.5	20.1
Class3	97.5	98.0	98.0	98.5	98.5	98.1

- Discussion

The feature of 2 will make the result a little bit strange for class 1. I think this is because class 1 has sparser distribution according to the graph on the first page. The spacer distribution might cause serious mis-classified for generative model but not a serious issue in discriminative model. That's the thing I found this time.

Probabilistic Discriminative Model

Supposed posterior is logistic sigmoid function or soft-max function; therefore, we can get:

$$p(C_k|\Phi) = \ y_k(\Phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \ ,where\ \ a_j = \mathbf{w}_k^T \Phi$$

$$\frac{\partial y_k}{\partial a_k} = y_k(I_{kj} - y_j)$$

For two classes case

For multi-class case

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} \left\{1 - y_n\right\}^{1-t_n} \rightarrow p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(\mathcal{C}_k|\phi_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

The error function for two / multi- class is

$$E(\mathbf{w}) = -\ln p(\mathbf{t}\,|\,\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} \rightarrow E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

cross-entropy error function *for the multiclass classification problem*

Next, for discriminative case, we are suggested to use Newton-Raphson to get proper w (basically is to iterate until the difference of $w^{(new)}$ and $w^{(old)}$ converged to the expected value.)

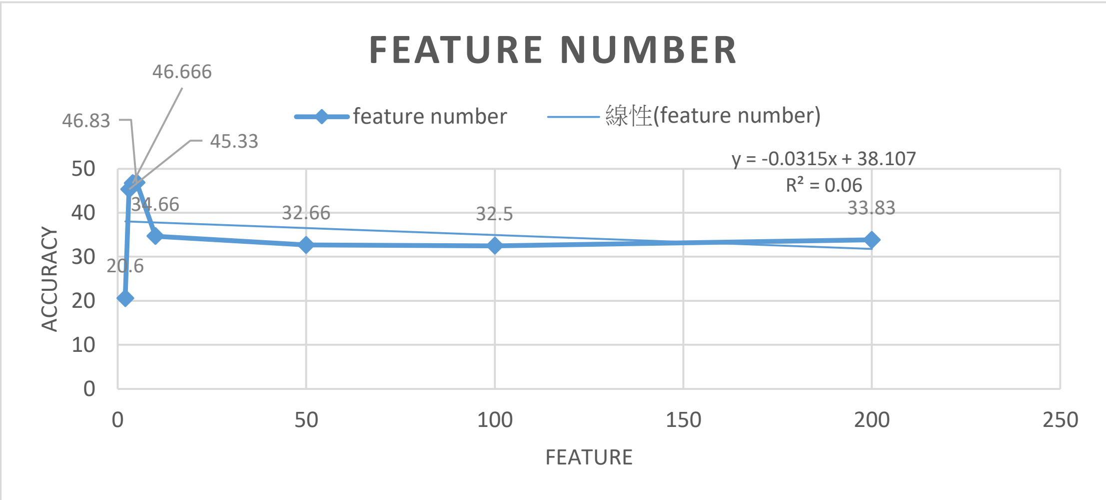
$$w^{(new)} = w^{(old)} - H^{-1}\nabla E(w)$$

N: 3000(# of datasets)

M: 900(# of feature of bmp graph; picture vector)

- Result

Accuracy(Phi)	2	3	4	5	10	50	100	200	500	Average
w_0	20.6	45.33	46.666	46.83	34.66	32.66	32.5	33.83	32.0	36.12%
No w_0	41.5	42.33	40.5	41.16	32	34	35	38	34	37.61%



- Discussion

```
> Training start
> Reading files..
> Divided the data into 5 chunks for the 0th cross validation set.
> Getting W Matrix..
In the iteration 100
In the iteration 0
In the iteration 0
In the iteration 0
In the iteration 0
In the iteration 0
```

I can't get a converged solution at first for several potential reason

1. R_matrix is too sparse

```
print R_matrix
```

```
[[ 0.22222222  0.          0.          ...,  0.          0.          0.        ]
 [ 0.          0.22222222  0.          ...,  0.          0.          0.        ]
 [ 0.          0.          0.22222222 ...,  0.          0.          0.        ]
 ...,
 [ 0.          0.          0.          ...,  0.          0.          0.        ]
 [ 0.          0.          0.          ...,  0.          0.          0.        ]
 [ 0.          0.          0.          ...,  0.          0.          0.        ]]
```

2. For p matrix which store possibility of each class is too small, which might lead to slow convergence.

```
print p
```

```
[[ 0.33333333]
 [ 0.33333333]
 [ 0.33333333]
 ...,
 [ 1.        ]
 [ 1.        ]
 [ 1.        ]]
```

After change my approach to R_matrix without divide the number of training set, I

```
> Training start
> Reading files..
> Divided the data into 5 chunks for the 0th cross validation set.
> Getting W Matrix..
In the iteration 100
In the iteration 0
In the iteration 0
In the iteration 0
In the iteration 0
In the iteration 0
In the iteration 0
In the iteration 0
In the iteration 0
In the iteration 0
In the iteration 0
In the iteration 0
training finished!
```

have larger R_matrix which will make converge success in this case
Successfully trained the model:

The discriminative model is open form solution which depends heavily on iteration times and computing ability, I didn't mention that in my case I just iterate for 1000 times and forced to break the loop of computing here. Compare to generative close form, it is suitable for the self-gen data and more applicable for the less-data reality problem.

- ## Unbalanced Data training

As I've mention before, the class 1 has sparser dataset than class 2 and class 3. I will

```
dir_address = "Data_Train"
cross_number = 1
training(dir_address,cross_number)
```

```
> Training start
> Reading files..
> Divided the data into 5 chunks for the 1th cross validation set.
> Getting W Matrix..
In the iteration 100.000000
training finished!
> Evaluating..
Discriminative model cross validation get 33.1666666667%
```

shrink the dataset of class 1 and see what would happen.

generative	Data	Generative(900 features)	Discriminative(3 features)
Class 1	500	50	
Class 2	1000	50	
Class 3	1000	100	
Total	2500	66.6	42.3333333333

Therefore, as we can see here for discriminative model improve from 36.12% to 42.33%, and as for generative model it is decreasing for less dataset we trained.