Machine Learning 2017

HW2: Linear Models for Classification

A052153 洪章瑋

Probabilistic Generative Model

Inference and decision problem can be splitted into two stage - inference, decision. In Probabilistic Generative Model method, we want to find posterior class probability via the following rule.

$$p(C_k|\phi) = \frac{p(\phi|C_k)p(C_k)}{\sum p(\phi|C_j)p(C_j)}$$

$p(\phi|C_k)$ is class-conditional probability. In this method we need to evaluate both class prior and class-conditional probability which is time-consuming and unnecessary.

Probabilistic Discriminative Model

Instead of finding class-conditional probabilities and priors, discriminative model directly solve posterior class probability with minimizing cross entropy loss. When optimization, I apply **Iterative Reweighted Least Squares (IRLS)** and **Newton-Raphson** update technique. (will be mentioned later)

Sigmoid and Softmax

We can rewrite $p(C_k|\phi) = \frac{p(\phi|C_k)p(C_k)}{\sum p(\phi|C_j)p(C_j)}$ to $\frac{1}{1+\exp(-a)} = \sigma(a)$ in binary classification case and $a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$. $a_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$ Assume $p(\phi|C_k)$ is Gaussian distribution, we can rewrite to the following formula:

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$

$$\begin{aligned} \mathbf{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}. \end{aligned}$$

When multi-class classification case, sigmoid can be generalized to softmax as the following formula.

$$p(\mathcal{C}_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

In Probabilistic Generative Model, we solve w via solving μ, $\mathbf{\Sigma}$. In Probabilistic Discriminative Model, we solve w via minimize cross-entropy loss.

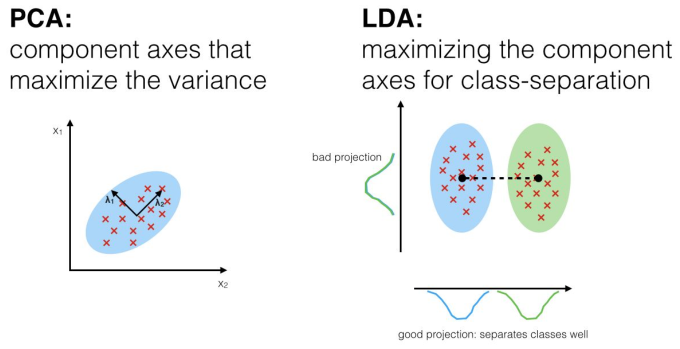
Principle Component Analysis

In order to overcome curse-of-dimensionality when handling high-dimensional data, dimension reduction is required. Principle Component Analysis (PCA) is a kind of dimension reduction technique. In this project, we do PCA by the following algorithm.

1. Calculate **Covariance matrix**
2. Given **Covariance matrix**, calculate **Eigen Vector**, **Eigen Value**.
3. Sort (**Eigen Value, Eigen Vector**) pairs by **|Eigen Value|**
4. Choose top-k Eigen Vector
5. Given input data X, k eigen vector, calculate multiplication of X and eigen vector to get $\mathbf{\Phi}$

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is another dimension reduction technique. The goal is to project a dataset onto a lower-dimensional space with good class-separability in order to avoid overfitting. The general LDA approach is very similar to a Principal Component Analysis, but in addition to finding the component axes that maximize the variance of our data (PCA), we are additionally interested in the axes that maximize the separation between multiple classes (LDA).


PCA:
component axes that maximize the variance
x_1
x_2
λ_1
λ_2
LDA:
maximizing the component axes for class-separation
bad projection
good projection: separates classes well

(PCA vs LDA, image from *sebastianraschka.com*)

Iterative Reweighted Least Squares - Newton-Raphson

There is no necessary for optimization in Probabilistic Generative Model. However, due to nonlinearity in logistic regression, there is no closed form solution in this case. Therefore, we use Newton-Raphson method to optimize weight in Probabilistic Discriminative Model.

Combine Newton-Raphson update method with sequential learning, we have a sequential weight update formula.

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

$$\mathbf{H} = \nabla\nabla E(\mathbf{w})$$

H is Hessian matrix. In multi-class classification case, this matrix can be calculated by the following formulas. **H** is (KxM)x(KxM) matrix which comprises block of size MxM in which block j, k. (K is number of class, M is number of components in feature vector)

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\, \boldsymbol{\phi}_n$$

$$\nabla_{\mathbf{w}_k}\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\sum_{n=1}^{N} y_{nk}(I_{kj} - y_{nj})\boldsymbol{\phi}_n\boldsymbol{\phi}_n^{\mathrm{T}}.$$

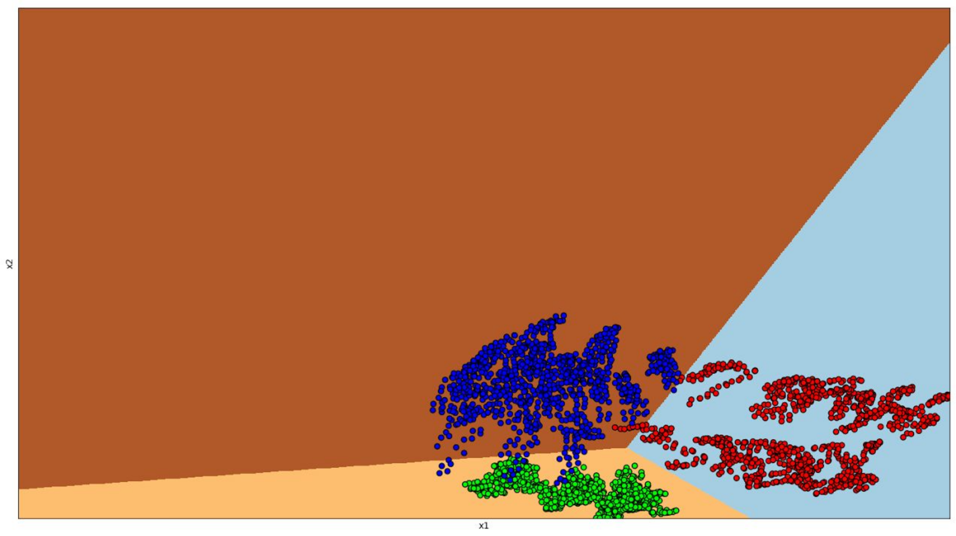
Experiment

Performance

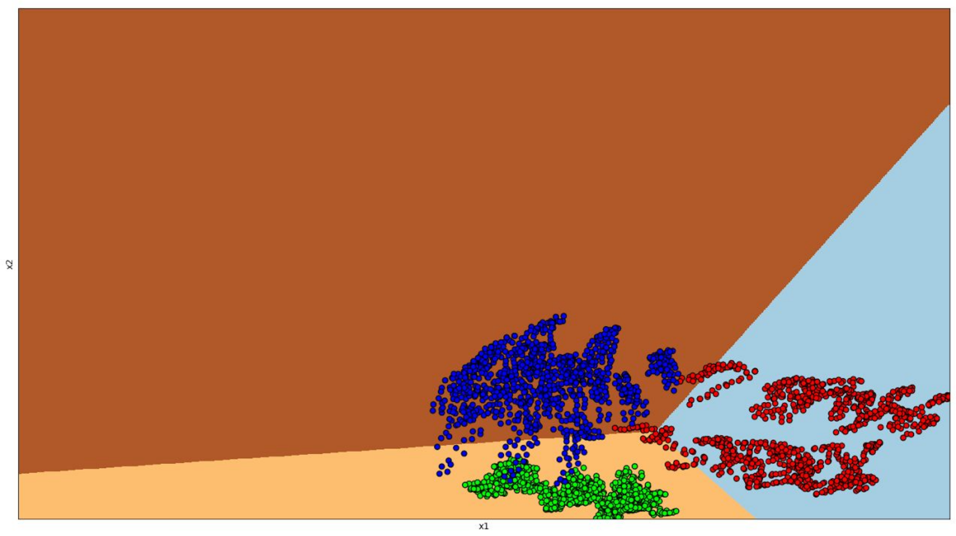
Error rate	
Probabilistic Generative Model	Probabilistic Discriminative Model
0.025	**0.013750**

The models of result above are trained with 800 data for each class, tested with 200 data for each class. Dimension reduced with PCA method.

Visualization



(Decision boundary of Probabilistic Discriminative Model, x1, x2 are components of ϕ(x))

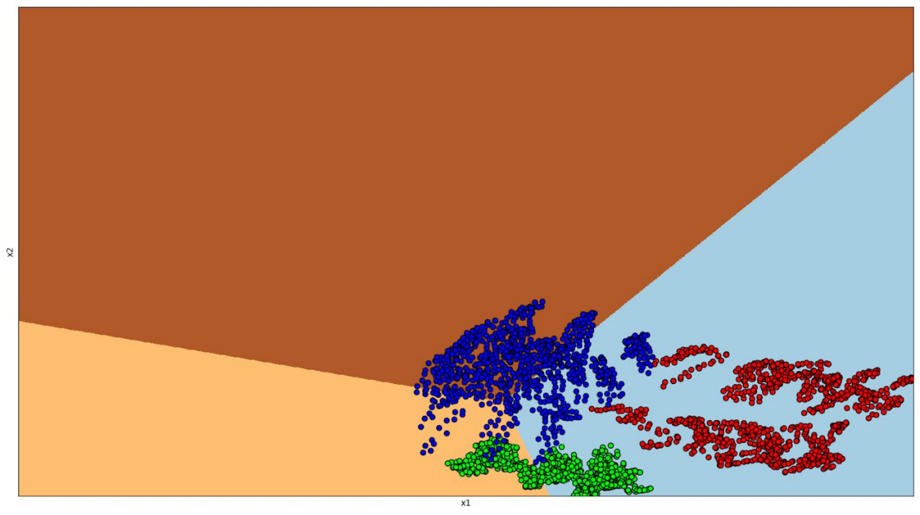


(Decision boundary of Probabilistic Generative Model, x1, x2 are components of ϕ(x))

It is obvious that the decision boundaries of both methods are similar in best case (i.e. enough and balanced training data)

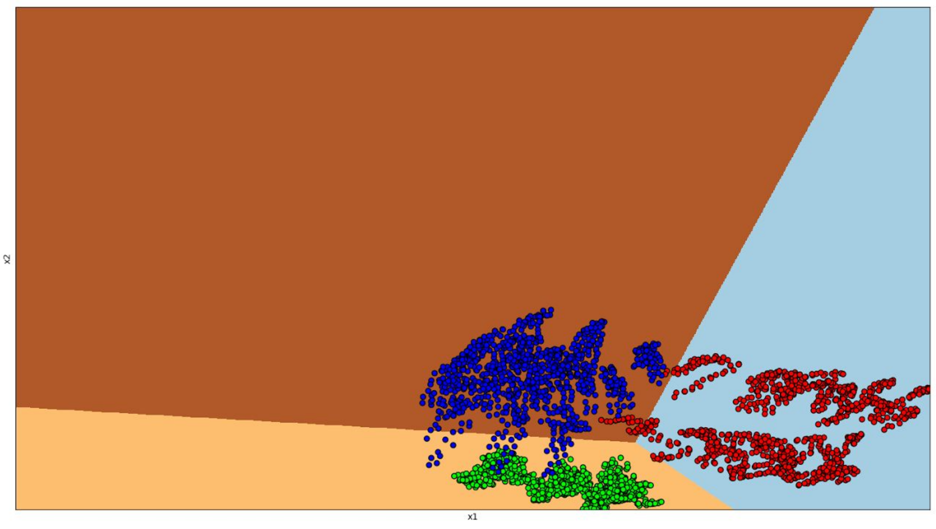
Balance vs Unbalance

In this section, I would show that the different influence of balance of training data to these two methods. We take Probabilistic Generative Model for example. Here we ignore the same experiment result for Probabilistic Discriminative Model since there is the same result as Probabilistic Discriminative Model.



(Decision boundary of Probabilistic Discriminative Model trained with **unbalanced** training data)

The result above is trained with 990 for class 1 (Red), 10 for class 2 (Green) and 10 for class 3 (Blue). It is not surprising that Probabilistic Generative Model fails when some datas are rare in training datas. The result accuracy is only **0.556** However, if we train model with the same amount of training datas for each class ?



(Decision boundary of Probabilistic Discriminative Model trained with **few but balance** training data)

The picture above is trained with 10 for class 1 (Red), 10 for class 2 (Green) and 10 for class 3 (Blue). To our surprise, we have only **0.97** accuracy in result. In conclusion, we learned that training with fewer datas cannot harm the accuracy of model, but **unbalanced amount of training datas for each class can**.

Amount of training data

In this section, I will discuss the influence of amount of training data. When there is only small amount of training data, which model performs better ?

Error rate (with 1 for each class)	
Probabilistic Generative Model	Probabilistic Discriminative Model
0.67	**0.08**

Error rate (with 10 for each class)	
Probabilistic Generative Model	Probabilistic Discriminative Model
0.05233	**0.05033**

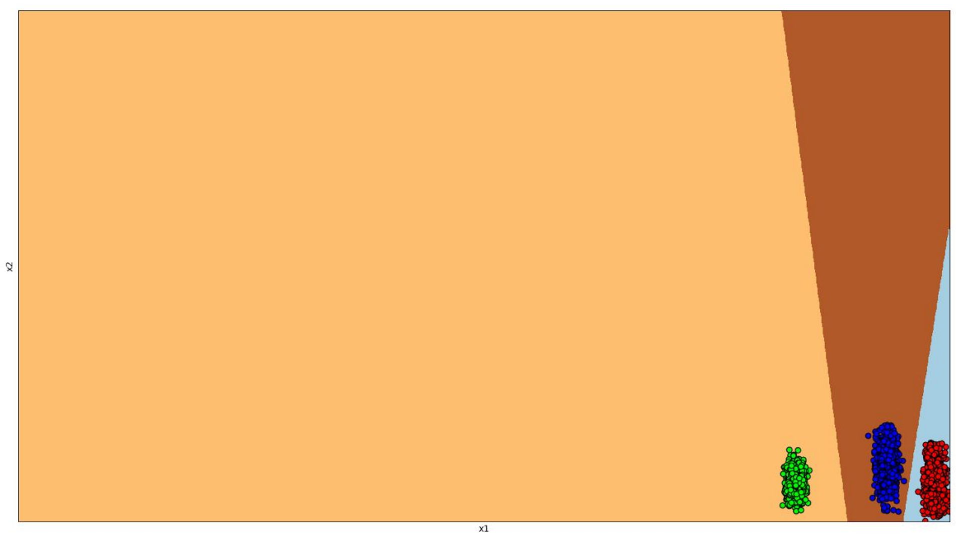
To conclude, **Probabilistic Discriminative Model performs better than Probabilistic Generative Model when there is small amount of training datas**.

Other dimension reduction - Linear Discriminant Analysis

In addition to PCA methods, I applied **Linear Discriminant Analysis (LDA)** which is mentioned in the previous sections to reduce dimension.

Error rate	
Probabilistic Generative Model	Probabilistic Discriminative Model
0.842	**0**

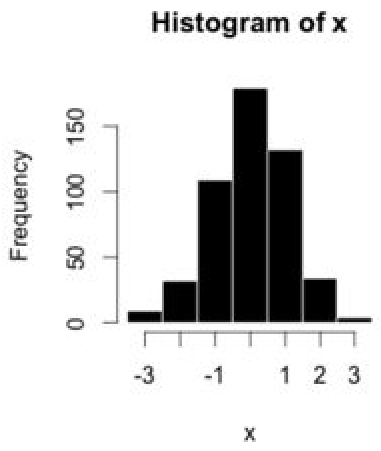
The final testing error rate is zero in Probabilistic Discriminative Model, however Probabilistic Generative Model get a error rate higher than PCA version. The following picture is the decision boundary of Probabilistic Discriminative Model.



(Decision boundary of Probabilistic Discriminative Model trained with LDA method)

Other dimension reduction - Image histogram

I convert the images into histogram (which dimension is 256). Each component in histogram means number of element in the image equals to specific level of intensity.



In the experiment, I trained both models with same settings as above (sees Section. Performance).

Error rate	
Probabilistic Generative Model	Probabilistic Discriminative Model
0.712	**0.0**

To our surprise, there is no error in Probabilistic Discriminative Model but Probabilistic Generative Model fails to fit histogram datas. Since this method map image into space with 256 dimension, I cannot plot its decision boundary in 2 dimensional space.