Spam Dataset Classifers

Xin Zhengfang A0206597U

Beta-binomial Naive Bayes

Intro

The class labels:

1. The class labels' λ is estimated using ML(maxmum likelihodd).
2. λ^{ML} is used as the plug-in estimator for testing

The features distribution:

1. A $\text{Beta}(\alpha, \alpha)$ prior is assumed on the features distribution.
2. The error rate is evaluated with $\alpha = \{0, 0.5, 1, 1.5, 2, \cdots, 100\}$ on the test data.
3. The Bayesian(i.e., posterior predictive) is used on training and testing.

Posterior Predictive Distribution with $\text{Beta}(a, b)$ prior:

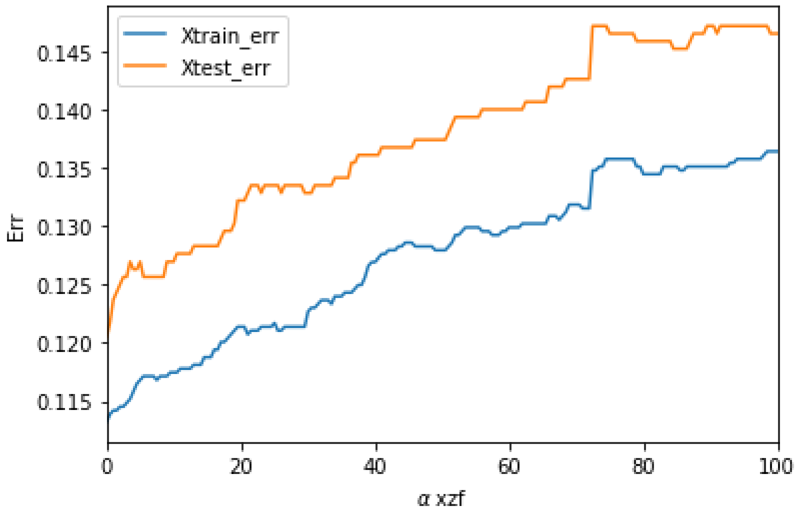
$$
\begin{aligned}
p(\tilde{x} = 1|D) &= \int_0^1 p(\tilde{x} = 1, \theta|D)d\theta = \int_0^1 p(\tilde{x} = 1|\theta, D)p(\theta|D)d\theta \\
&= \int_0^1 p(\tilde{x} = 1|\theta)p(\theta|D)d\theta = \int_0^1 \theta p(\theta|D)d\theta \\
&= E(\theta|D) = \frac{N_1 + a}{N + a + b}
\end{aligned}
$$

Maxmum likelihood for the class labels with binomial:

$$\hat{\theta}_{ML} = \frac{N_1}{N} \text{ by setting a} = \text{b} = 1 \text{ (uniform prior)}$$

Result

Plots of training and test error rates versus α



What do you observe about the training and test errors as α change?

As α increases, the training and test error_rate are both tend to increase.

Training and testing error rates for α = 1, 10 and 100.

```
Training error rates:
     α=1   0.11419249592169656
     α=10  0.1174551386623165
     α=100 0.13637846655791186
Testing error rates:
     α=1   0.12369791666666663
     α=10  0.126953125
     α=100 0.146484375
```

Gaussian Naive Bayes

Intro

The class label:

1. Because dataset has a lot of spam and non-spam emails, we don't need do some prior assumption.The maxmum likelihood λ^{ML} can be used as the plug-in estimator for testing.

The features distribution:

1. To simplify the question, Maxmum likehood is used with univariate gaussian prior.

ML estimation of μ ,σ giving training data $D = \{x_1, \ldots, x_N\}$ $D = \{x_1, \ldots, x_N\}$:

$$\frac{\partial L}{\partial \mu} = \frac{\partial}{\partial \mu}\left(\sum_{n=1}^{N} -\frac{(x_n-\mu)^2}{2\sigma^2}\right) = \sum_{n=1}^{N}\frac{(x_n-\mu)}{\sigma^2} = 0$$
$$\implies \hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

$$\frac{\partial L}{\partial \sigma} = \frac{\partial}{\partial \sigma}\left(\sum_{n=1}^{N} -\frac{(x_n-\mu)^2}{2\sigma^2} - N\log\sigma\right) = \sum_{n}\frac{(x_n-\mu)^2}{\sigma^3} - \frac{N}{\sigma} = 0$$
$$\implies \hat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu)^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n-\hat{\mu})^2$$

Result

Training and testing error rates for the log-transformed data.

```
Training error rates:  0.10995106035889068
Testing error rates:  0.109375
```

The preprocessing does truly have impact on the error rates. The `log(data+1e-10)` was used into preprocessing in this result. You can change the preprocessing in my code to check the standard answer.

Logistic Regression

Intro

The class label:

1. Because dataset has a lot of spam and non-spam emails, we don't need do some prior assumption.The maxmum likelihood λ^{ML} can be used as the plug-in estimator for testing.

The features distribution:

1. We use logistic regression model to fit the spamdata distribution. In logistic regression, we use parameters w and sigmiod function to simulate the spamdata distribution.

$$\text{Binary case: } p(y|x,w) = \text{Ber}(y|\mu(x,w)) = \text{Ber}\big(y|\,\text{sigm}\big(w^T x\big)\big)$$

2. In the training, we adjust w to get best erro rate.

Numerical Optimization

1. The loss is negative log likelihood to estimate the performance of fitting.

$$\log p\left(y_i = 1|x_i, w\right) = \log\frac{1}{1+\exp(-w^T x_i)} = \log \mu_i$$

$$\log p\left(y_i = 0|x_i, w\right) = \log(1 - p\left(y_i = 1|x_i, w\right)) = \log(1-\mu_i)$$

$$NLL(w) = -\sum_{i=1}^{N}\log p\left(y_i|x_i, w\right) = -\sum_{i=1}^{N}\left[y_i\log\mu_i + (1-y_i)\log(1-\mu_i)\right]$$

where y_i is ith label, x_i is ith sample's feature vector. $w^T x_i$ should be a scalar.

2. The loss with Regularization

$$NLL_{reg}(\mathbf{w}) = NLL(\mathbf{w}) + \frac{1}{2}\lambda w^T w$$

```
PS:don't place penalize on the bias.
```

2. Using Newton's method to find better w. Taylor expentation:

$$f(\theta_k + d_k) \approx f_{quad} = f(\theta_k) + d_k^T \nabla f + \frac{1}{2} d_k^T H d_k$$

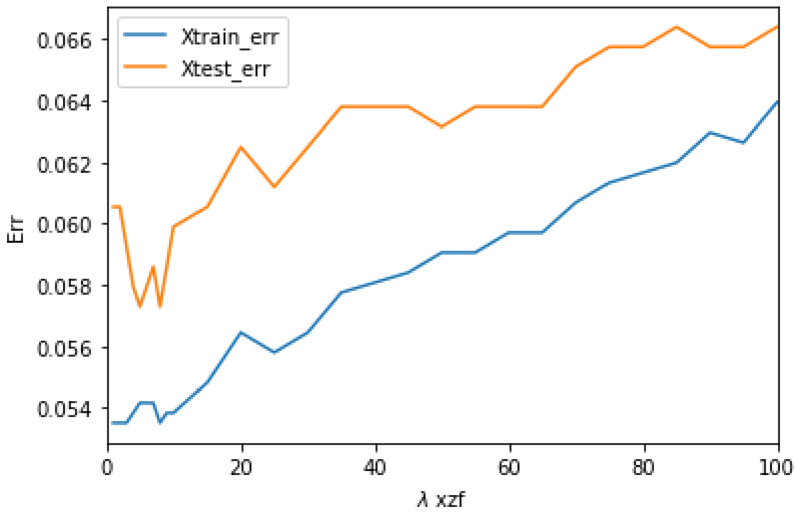
Differentiate f_{quad} equal to zero:

$$\nabla f + H d_k = 0 \Longrightarrow d_k = -H^{-1} \nabla f$$

3. Stop optimizing when the loss converge

Result

Plots of training and test error rates versus λ



There are a lot of uncertainity in training and predicting, such as the learning rate and tolerances. What's more, it is really slow to run and debug.

What do you observe about the training and test errors as λchange?

Generally, as λ increases, the training and test error are both tend to increase. Because constraint is so strong to fit the data.

There is a overfitting phenomenon from λ =1 to about 7 or 8. λ from 1 to 7, the overfitting become weaken. Therefore, test error decreases and train error incrase.

Training and testing error rates for λ = 1, 10 and 100.

```
Training error rates:
     λ=1    0.05350734094616638
     λ=10   0.0538336052202284
     λ=100  0.06394779771615011
Testing error rates:
     λ=1    0.060546875
     λ=10   0.05989583333333337
     λ=100  0.06640625
```

K Nearest neighbor classifier

Intro

1. Define a kind of distance.
2. Measure the distance between the candidate sample with all other training samples
3. Choose k nearest traning samples as voters
4. Vote for the candidate's label.

Above is just my simple peronal understanding.

See the detailed context in Pattern_XINClassification_by_Richard_O._Dud__CHAPTER 4.4

We use L2 distance here:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

where p q are feature vectors.

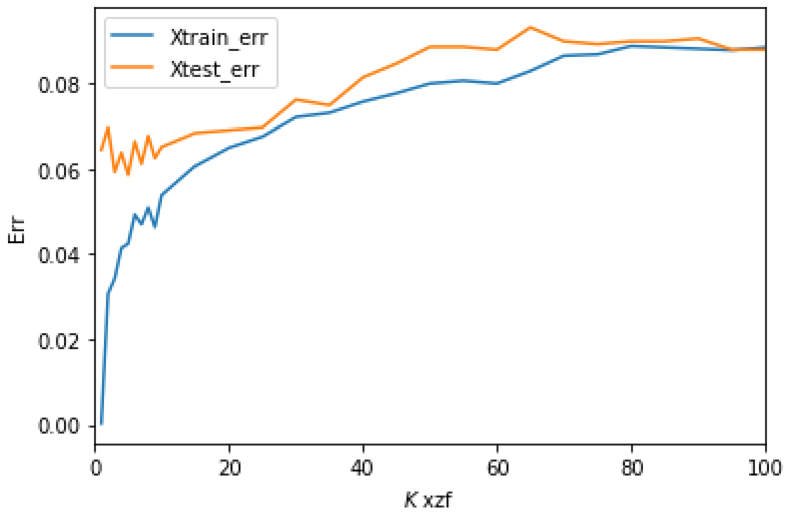
Use martix operation to accelate the calculation.

$$(X_1 - X_2)^2 = X_1^2 + X_2^2 - 2X_1^T X_2$$

where X_1 is (N_1, D), and X_2 is (N_2, D)

Result

Plots of training and test error rates versus K



What do you observe about the training and test errors as K change?

As K increases, the training and test error are both tend to increase. There is a weak overfitting phenomenon from k=1 to k=4. k from 1 to 4, the overfitting become weaken. Therefore, test error decreases and train error inrease from k=1

Training and testing error rates for K = 1, 10 and 100.

```
Training error rates:
     K=1   0.00032626427406201586
     K=10  0.0538336052202284
     K=100 0.0884176182707993
Testing error rates:
     K=1   0.064453125
     K=10  0.06510416666666663
     K=100 0.087890625
```

Survey

12 hours are for Beta-binomial Naive Bayes, where 8 hours are for debug and 4 hours are for writing framework.

10 hours are for Gaussian Naive Bayes, where 8 hours are for debug and 2 hours are for writing framework.

20 hours are for Logistic Regression, where 14 hours are for debug and 6 hours are for writing framework. I rewrite it for twice. At first time , I have so many functions, which makes me really hard to figure out what's wrong in my code. It is also diffcult to debug in jupyter, better to use `pycharm` or `VS code`.

6 hours are for KNN, where 4 hours are for debug and 2 hours for writing framework.