Problem Set 2, CS229(Machine Learning)

Ma Yubo

July, 4th, 2021

1 Logistic Regression: Training Stability

1.1

When training on dataset A, the logistic regression converges rapidly.
When training on dataset B, the algorithm fails to converge.

1.2

1.2.1 Linear Separability or not

We claim that **dataset B is linear seperable while dataset A is not**.

You can have an intuitive impression on it by the scatter plot of these two dataset.

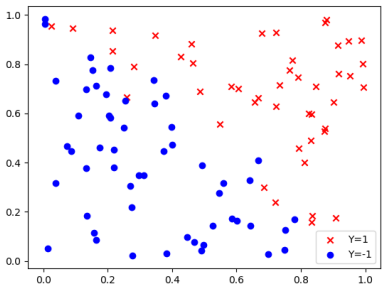

Figure 1.1: Scatter plot of dataset A

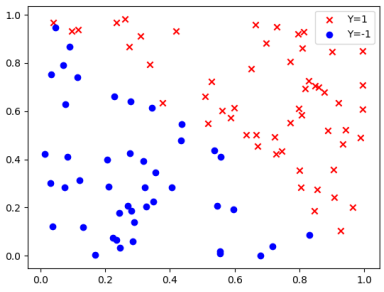

Figure 1.2: Scatter plot of dataset B

To get a more solid evidence about their separability, we train a logistic regression model on these two datsets, respectively. Then we get their decision boundaries and see whether the predicted categories of $Y's$ fit their true values.

Clearly, samples in B are classified perfectly by decision boundary while samples in A are not.

Table 1.1: Accuracy of fitting result (according to learned decision boundary)

Dataset	Acc
A	0.92
B	1.00

1.2.2 why linear separable results in divergence

It is anti-intuitive to some extent that a (generalize) linear model trained with a linear-seperable dataset should lead to divergence! We'll explore the mechanism and explain why it will occur.

Denote the sample in dataset as $\{(x_i, y_i)\}_{i=1}^{N}$, where x_i a vector and $y_i \in \{-1, 1\}$. The loss function of this model is:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} log(1 + exp(-y_i \theta^T x_i)) \tag{1.1}$$

[1] If the samples in dataset are linear seperable, then there exists θ^* satisfying $y_i \theta^{*T} x_i > 0, \forall i \in \{1, 2, ..., N\}$.
If so, $\forall \theta^*$ and $\forall K > 1$, we have:

$$J(K\theta^*) < J(\theta^*) \tag{1.2}$$

Therefore, when $y_i \theta^{*T} x_i \to \infty$ by increasing θ iteratively, the algorithm will not end by just constraining the $||\theta_{t-1} - \theta_t||$.

[2] If the samples in dataset are not linear seperable, however, there doesn't exist an "optimal" θ as the last situation. So $y_i \theta^T x_i$ will not go infinity for every i and the norm of learned θ will not increase without any constraint and the algorithm will converge at some time.

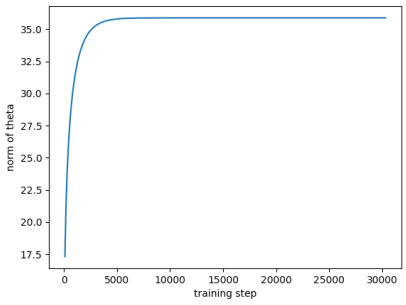

Figure 1.3: norm of learned parameter in dataset A

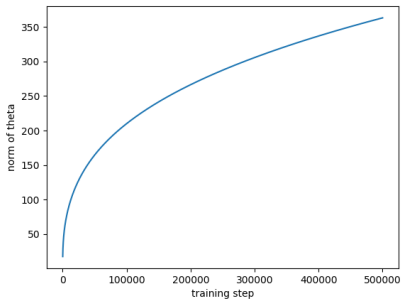

Figure 1.4: norm of learned parameter in dataset B

1.3

1.3.1 distant constant leanring rate

Practically Not. Too small learning rate will cause θ and $J(\theta)$ hard to change and the algorithm fails to converge within acceptable time range.

1.3.2 learning rate decay

Yes. See these two figures below. Note a longer training time is needed for convergence if the learning rate decays.

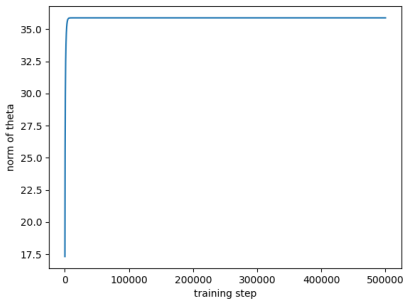

Figure 1.5: norm of learned parameter in dataset A

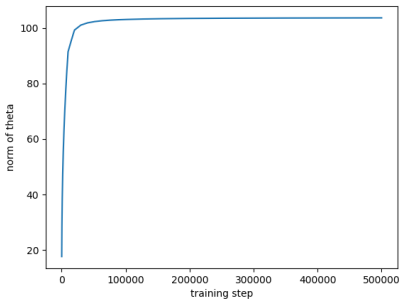

Figure 1.6: norm of learned parameter in dataset B

1.3.3 input feature scaling

No. The reason is similar to smaller constant learning rate.

1.3.4 parameter regularization

Yes. Preventing θ from being too large is valid.

1.3.5 noise

Maybe. If the co-variance matrix of this Gaussian distribution is appropriate, a linear separable dataset may convert to a linear unseparable dataset.

1.4

Support vector machine with hinge loss is not vulnerable for linear separable dataset.

- From intuitive perspective, the SVM algorithm optimize the *geometric margin* rather than the *functional margin* between samples of two categories.

- From mathematical perspective, SVM with hinge loss is equivalent to SVM with soft margin. Its unconstrained objective function is:

$$min_{\omega,b}\left\{\frac{1}{2}\omega^T\omega + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\lambda_i[1-\xi_i - y_i(\omega^T x_i + b)] - \sum_{i=1}^{N} r_i\xi_i\right\} \tag{1.3}$$

  Unlike logistic regression (without extra regularization), the scale of parameter in SVM is implicitly constrained in the format of $min\frac{1}{2}\omega^T\omega$.

In summary, SVM with hinge loss is optimized with regularization automatically so it will not collapse when encountering linear separable dataset.

2 Model Calibration

2.1

The formula we need to prove is:

$$\sum_{i\in S} h_\theta(x^{(i)}) = \sum_{i\in S} I(y^{(i)} = 1) \tag{2.1}$$

Write down the log-likelihood function about parameters θ and take derivation on it to get optimal parameters: θ^*_{MLE}

$$L(\theta) = -\frac{1}{N}\sum_i [y^{(i)} log h_\theta(x^{(i)}) + (1-y^{(i)}) log(1-h_\theta(x^{(i)}))] \tag{2.2}$$

$$\frac{\partial L}{\partial \theta} = -\frac{1}{N}\sum_i [(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}] = 0 \tag{2.3}$$

Write the equation above as the format of matrix and we get:

$$X^T(Y - H) = 0 \tag{2.4}$$

where

$$X = \begin{bmatrix} \text{———} & x^{(1)} & \text{———} \\ & \vdots & \\ \text{———} & x^{(N)} & \text{———} \end{bmatrix} \tag{2.5}$$

$$Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix} \tag{2.6}$$

$$H = \begin{bmatrix} h_\theta(x^{(1)}) \\ \vdots \\ h_\theta(x^{(N)}) \end{bmatrix} \tag{2.7}$$

Consider the first row of X^T (the first column of X), which is composed of the bias term 1 of each sample. Use the first row of X^T to left multiply the $(Y - H)$, we get:

$$\begin{aligned} &\sum_{i \in S} (h_\theta(x^{(i)}) - y^{(i)}) \\ &= \sum_{i \in S} (h_\theta(x^{(i)}) - I(y^{(i)} = 1)) = 0 \end{aligned} \tag{2.8}$$

2.2

- No. Consider a toy example with a 2-sample dataset: $\left\{(x^{(i)}), y^{(i)})\right\}_{i=1}^{2}$. It consists a positive example and a negative example: $y^{(1)} = 1$ and $y^{(2)} = 0$. If our model gives a totally converse prediction: $h_\theta(x^{(1)}) = 0$ and $h_\theta(x^{(2)}) = 1$, the equation above still holds true but the prediction accuracy of this model is **0** !
- Necessarily true. Otherwise, the equation $X^T(Y - H) = 0$ doesn't hold true and θ at this point will not be the optimal MLE estimation θ^*_{MLE}.

2.3

Similarly, the objective function:

$$L(\theta) = -\frac{1}{N} \sum_i [y^{(i)} log h_\theta(x^{(i)}) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)}))] + \lambda \theta^T \theta \tag{2.9}$$

$$\frac{\partial L}{\partial \theta} = -\frac{1}{N} \sum_i [(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}] + 2\lambda\theta = 0 \tag{2.10}$$

Write the equation above as the format of matrix and we get:

$$X^T(Y - H) = C\theta \tag{2.11}$$

where C is a non-zero constant.

In summary, the property of well-calibrated will not hold perfectly true since we estimate the parameters of model from both data and parameter prior.

3 Bayesian Interpretation of Regularization

3.1

By Bayesian formula, we have:

$$\begin{aligned} P(\theta|x, y) &= \frac{P(y|x, \theta)P(\theta|x)}{P(y|x)} \\ &= \frac{P(y|x, \theta)P(\theta)}{P(y|x)} \qquad if P(\theta) = P(\theta|x) \end{aligned} \tag{3.1}$$

Therefore, $\theta_{MAP} = argmax_\theta P(y|x, \theta)P(\theta)$

3.2

We know $\theta \sim N(0, \eta^2 I)$. Therefore, $P(\theta) = \frac{1}{(2\pi)^{d/2}\eta^d} exp(-\frac{1}{2\eta^2}\theta^T\theta)$

$$\begin{aligned} \theta_{MAP} &= argmax_\theta P(y|x,\theta)P(\theta) \\ &= argmax_\theta(logP(y|x,\theta) + logP(\theta)) \\ &= argmin_\theta(-logP(y|x,\theta) + \frac{1}{2\eta^2}||\theta||^2) \end{aligned} \tag{3.2}$$

Thus $\lambda = \frac{1}{2\eta^2}$

3.3

By the property of multivariate gaussian distribution,

$$y|x,\theta \sim N(X\theta, \sigma^2 I) \tag{3.3}$$

Therefore,

$$\begin{aligned} &\theta_{MAP} \\ &= argmin_\theta(-logP(y|x,\theta) + \frac{1}{2\eta^2}||\theta||^2) \\ &= argmin_\theta[\frac{1}{2\sigma^2}(y - X\theta)^T(y - X\theta) + \frac{1}{2\eta^2}\theta^T\theta] \end{aligned} \tag{3.4}$$

Take derivation on θ,

$$\begin{aligned} &\nabla_\theta[\frac{1}{2\sigma^2}(y - X\theta)^T(y - X\theta) + \frac{1}{2\eta^2}\theta^T\theta] \\ &= \frac{1}{\sigma^2}(X^TX\theta - X^Ty) + \frac{1}{\eta^2}\theta \\ &= 0 \end{aligned} \tag{3.5}$$

Finally we get the analytical form of θ_{MAP} under this specific situation.

$$\theta_{MAP} = (X^TX + \frac{\sigma^2 I}{\eta^2})^{-1}X^Ty \tag{3.6}$$

3.4

Similarly,

$$\begin{aligned} &\theta_{MAP} \\ &= argmin_\theta(-logP(y|x,\theta) - logP(\theta)) \\ &= argmin_\theta \quad \frac{1}{2\sigma^2}(y - X\theta)^T(y - X\theta) + \frac{|\theta|}{b} \\ &= argmin_\theta \quad ||y - X\theta||_2^2 + \frac{2\sigma^2}{b}||\theta||_1 \end{aligned} \tag{3.7}$$

Thus $\gamma = \frac{2\sigma^2}{b}$

4 Constructing Kernels

According to Mercer Theorem, $K(x_i, x_j)$ is a valid kernel **if and only** its corresponding kernel matrix $K_{ij} = K(x_i, x_j)$ is symmetric and positive semi-definite. Now we know both K_1 and K_2 is valid kernel. Therefore, $\forall z \in R^m$, $z^T K_1 z \geq 0$, $z^T K_2 z \geq 0$.

4.1 $K(x_i, x_j) = K_1(x_i, x_j) + K_2(x_i, x_j)$

Yes. Given z is a non-zero vector in R^m

$$\begin{aligned} & z^T K z \\ & = z^T K_1 z + z^T K_2 z \geq 0 \end{aligned} \tag{4.1}$$

4.2 $K(x_i, x_j) = K_1(x_i, x_j) - K_2(x_i, x_j)$

May not. There may exist $z \in R^m$

$$\begin{aligned} & z^T K z \\ & = z^T K_1 z - z^T K_2 z \leq 0 \end{aligned} \tag{4.2}$$

4.3 $K(x_i, x_j) = aK_1(x_i, x_j)$

Yes. Given z is a non-zero vector in R^m

$$\begin{aligned} & z^T K z \\ & = a z^T K_1 z \geq 0 \end{aligned} \tag{4.3}$$

4.4 $K(x_i, x_j) = -aK_1(x_i, x_j)$

No. Given z is a non-zero vector in R^m

$$\begin{aligned} & z^T K z \\ & = -a z^T K_1 z \leq 0 \end{aligned} \tag{4.4}$$

4.5 $K(x_i, x_j) = K_1(x_i, x_j) K_2(x_i, x_j)$

Yes.
By definition, Kernel matrix $K_{ij} = K(x_i, x_j) = K_1(x_i, x_j) K_2(x_i, x_j)$. So $K = K_1 * K_2$, where $*$ means element-wise multiplication.

Given z is a non-zero vector in R^m

$$\begin{aligned} & z^T K z \\ & = z^T (K_1 * K_2) z \\ & = \sum_i \sum_j z_i K_1(x_i, x_j) K_2(x_i, x_j) z_j \\ & = \sum_i \sum_j z_i \phi_1(x_i)^T \phi_1(x_i) \phi_2(x_i)^T \phi_2(x_i) z_j \\ & = (\sum_i z_i \phi_1(x_i) \phi_2(x_i))^T (\sum_j z_j \phi_1(x_j) \phi_2(x_j)) \\ & = ||\mathbf{z} * \phi_1(\mathbf{x}) * \phi_2(\mathbf{x})||^2 \geq 0 \end{aligned} \tag{4.5}$$

4.6 $K(x_i, x_j) = f(x_i) f(x_j)$

Yes. Given z is a non-zero vector in R^m

$$\begin{aligned} & z^T K z \\ & = \sum_i \sum_j z_i f(x_i) f(x_j) z_j \\ & = ||\mathbf{z} * f(\mathbf{x})||^2 \geq 0 \end{aligned} \tag{4.6}$$

4.7 $K(x_i, x_j) = K_3(\phi(x_i), \phi(x_j))$

Yes. Given z is a non-zero vector in R^m

$$\begin{aligned} & z^T K z \\ & = \sum_i \sum_j z_i K_3(\phi(x_i), \phi(x_j)) z_j \\ & = \sum_i \sum_j z_i \psi(\phi(x_i))^T \psi(\phi(x_j)) z_j \\ & = ||\mathbf{z} * \psi(\phi(\mathbf{x}))||^2 \geq 0 \end{aligned} \tag{4.7}$$

4.8 $K(x_i, x_j) = \boldsymbol{p}(K_1(x_i, x_j))$

Yes.
Denote $p(x) = \sum_k c_k x^k$. From mathematical induction, we know that if $K_1(x_i, x_j)$ is positive semi-definite matrix, then $K_1(x_i, x_j)^k$ is positive semi-definite matrix as well, $\forall k \in N^+$.
Therefore, $p(K_1(x_i, x_j)) = \sum_k c_k K_1(x_i, x_j)^k$ is positive matrix as well.

5 Kernelizing the Perceptron

5.1

5.1.1

We will proof that $\theta^{(i)}$ can be represented as $\theta^{(i)} = \sum_{j=1}^{i} \beta_j \phi(x^{(j)})$ by mathematical induction.

- Initialize $\theta^{(0)}$ as 0. When $i = 1$, $\theta^{(1)} = \alpha(y^{(1)} - 1)\phi(x^{(1)})$.
- Suppose the assumption is right when $i \leq j$.
  When $i = j + 1$,

$$\begin{aligned}
&\theta^{(i+1)} = \theta^{(i)} + \alpha(y^{(i+1)} - h_{\theta^{(i)}}(\phi(x^{(i+1)})))\phi(x^{(i+1)}) \\
&= \sum_{j=1}^{i} \beta_j \phi(x^{(j)}) + \alpha(y^{(i+1)} - h_{\theta^{(i)}}(\phi(x^{(i+1)})))\phi(x^{(i+1)}) \\
&= \sum_{j=1}^{i+1} \beta_j \phi(x^{(j)})
\end{aligned} \tag{5.1}$$

  where $\beta^{(i+1)} = \alpha(y^{(i+1)} - 1)$ or $\alpha y^{(i+1)}$ determined by the value of sign function.

Q.E.D.

5.1.2

$$\begin{aligned}
&h_{\theta^{(i)}}(\phi(x^{(i+1)})) \\
&= g({\theta^{(i)}}^T \phi(x^{(i+1)})) \\
&= sign(\sum_{j=1}^{i} \beta_j \phi(x^{(j)})^T \phi(x^{(i+1)})) \\
&= sign(\sum_{j=1}^{i} \beta_j K(x^{(j)}, x^{(i+1)}))
\end{aligned} \tag{5.2}$$

5.1.3

The updating rule on θ is:

$$\begin{aligned}
&\theta^{(i+1)} = \theta^{(i)} + \alpha(y^{(i+1)} - h_{\theta^{(i)}}(\phi(x^{(i+1)})))\phi(x^{(i+1)}) \\
&= \sum_{j=1}^{i} \beta_j \phi(x^{(j)}) + \alpha(y^{(i+1)} - sign(\sum_{j=1}^{i} \beta_j K(x^{(j)}, x^{(i+1)}))\phi(x^{(i+1)})
\end{aligned} \tag{5.3}$$

Thus we have the recursive formula about β:

- $\beta_0 = 0$
- $\beta_i = \alpha(y^{(i)} - sign(\sum_{j=1}^{i-1} \beta_j K(x^{(j)}, x^{(i)})))$ $\quad \forall i = 1, \ldots$

5.2

Coding problem. Implemented in **p05_percept.py**.

5.3

The test result is shown as below. Two background colors represent predict results by Perceptron while two kinds of data points represent their ground truth labels.

From the plots we find that **dot kernel** performs badly. That's because a dot

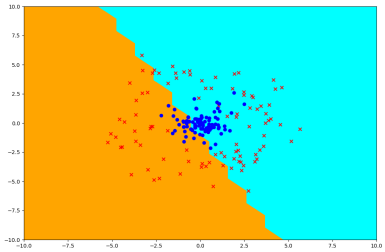

Figure 5.1: Prediction result (dot kernel)

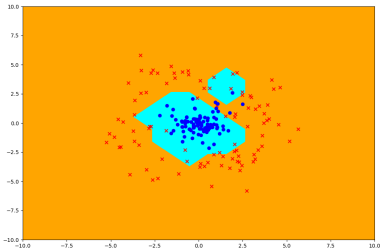

Figure 5.2: Prediction result (rbf kernel)

kernel function is equivalent to a **identical mapping**:

$$K(x, z) = x^T z \Longleftrightarrow \phi(x) = x \tag{5.4}$$

Thus the kernel method with a dot kernel doesn't actually do nothing on features of sample (x_i) and they are still linear-inseparable after feature mapping.

6 Spam Classification

6.1

Coding problem. Implemented in **p06_spam.py**.

6.2

Solve the spam classification problem with **Multinomial Event Model** and **Laplace Smoothing**.

Given a processed dataset $\left\{(x^{(i)}, y^{(i)})\right\}_{(i=1)}^{N}$, where $x^{(i)} \in R^{|V|}$, $|V|$ is the size of vocabulary and j-th component in $x^{(i)}$ (denote as $x_j^{(i)}$) represents the occurrence times of j-th word in vocabulary in i-th raw text. Also, $y^{(i)}$ is the binary category

of the text (spam email or not).
In multinomial event model, we have the prior of text category:

$$y \sim Binomial(\mu) \tag{6.1}$$

and the posterior probabilities:

$$P(x_j^{(i)} = k|y = 1) = \phi_{k|y=1} \tag{6.2}$$

$$P(x_j^{(i)} = k|y = 0) = \phi_{k|y=0} \tag{6.3}$$

Then the log-likelihood of data is given by:

$$\begin{aligned} L(\phi, \mu) &= \prod_{n=1}^{N} P(x^{(i)}, y^{(i)}; \phi, \mu) \\ &= \prod_{n=1}^{N} \prod_{j=1}^{d_i} P(x_j^{(i)}|y^{(i)}; \phi) P(y^{(i)}; \mu) \end{aligned} \tag{6.4}$$

where d_i is the length of i-th rat text sample.

$$\begin{aligned} &log \quad P(x_j^{(i)}|y^{(i)}; \phi) \\ &= log \quad \prod_{k=1}^{|V|} (\phi_{k|y=1})^{I(y^{(i)}=1 \wedge x_j^{(i)}=k)} (\phi_{k|y=0})^{I(y^{(i)}=0 \wedge x_j^{(i)}=k)} \\ &= \sum_{k=1}^{|V|} I(y^{(i)} = 1 \wedge x_j^{(i)} = k) log(\phi_{k|y=1}) + I(y = 0 \wedge x_j^{(i)} = k) log(\phi_{k|y=0}) \end{aligned} \tag{6.5}$$

$$\begin{aligned} &log \quad L(\phi, \mu) \\ &= \sum_{n=1}^{N} [\sum_{j=1}^{d_i} \sum_{k=1}^{|V|} I(y^{(i)} = 1 \wedge x_j^{(i)} = k) log(\phi_{k|y=1}) + I(y^{(i)} = 0 \wedge x_j^{(i)} = k) log(\phi_{k|y=0}) \\ &+ I(y^{(i)} = 0) log(1 - \mu) + I(y^{(i)} = 1) log(\mu)] \end{aligned} \tag{6.6}$$

Maximizing this yields the MLE estimation of these parameters:

$$\mu = \frac{\sum_{i=1}^{N} I(y^{(i)} = 1)}{N} \tag{6.7}$$

$$\phi_{k|y=1} = \frac{\sum_{n=1}^{N} \sum_{j=1}^{d_i} I(y^{(i)} = 1 \wedge x_j^{(i)} = k)}{\sum_{n=1}^{N} I(y^{(i)} = 1) d_i} \tag{6.8}$$

$$\phi_{k|y=0} = \frac{\sum_{n=1}^{N} \sum_{j=1}^{d_i} I(y^{(i)} = 0 \wedge x_j^{(i)} = k)}{\sum_{n=1}^{N} I(y^{(i)} = 0) d_i} \tag{6.9}$$

The final estimation with **Laplace Smoothing Calibration** is:

$$\mu = \frac{1 + \sum_{i=1}^{N} I(y^{(i)} = 1)}{N + 2} \tag{6.10}$$

$$\phi_{k|y=1} = \frac{1 + \sum_{n=1}^{N} \sum_{j=1}^{d_i} I(y^{(i)} = 1 \wedge x_j^{(i)} = k) + 1}{|V| + \sum_{n=1}^{N} I(y^{(i)} = 1) d_i} \tag{6.11}$$

$$\phi_{k|y=0} = \frac{1 + \sum_{n=1}^{N} \sum_{j=1}^{d_i} I(y^{(i)} = 0 \wedge x_j^{(i)} = k)}{|V| + \sum_{n=1}^{N} I(y^{(i)} = 0) d_i} \tag{6.12}$$

Given a new sample (x, y), we will compare the joint probabilities $P(x, y = 1)$ and $P(x, y = 0)$ and choose the larger one as our prediction result.

$$P(x, y = 1) = \mu \prod_{j=1}^{d} \phi_{k|y=1}(k = x_j) \tag{6.13}$$

$$P(x, y = 0) = (1 - \mu) \prod_{j=1}^{d} \phi_{k|y=0}(k = x_j) \tag{6.14}$$

We implement Naive Bayes algorithm on dataset provided and obtain an accuracy of **%97.85** on the testing set.

6.3

The most 5 indicative tokens are: **claim, won, prize, tone, urgent!**
And their log-ratios are:

$$logratio(w) = log \frac{\phi_{k|y=1}(k = idx(w))}{\phi_{k|y=0}(k = idx(w))} \tag{6.15}$$

Table 6.1: most 5 indicative tokens and their log-ratios

token	logratio
claim	5.71
won	5.10
prize	5.08
tone	4.80
urgent!	4.77

6.4

We use SVM with radius kernel on the same dataset. With different radius, we obtain different validation accuracy.

Thus the optimal radius under this setting is **0.1**. We use model trained with radius 0.1 to predict and obtain an accuracy of **%96.95** on the testing set.

Table 6.2: validation accuracy with different radius

radius	validation accuracy
0.01	0.917
0.1	0.946
1	0.930
10	0.876