

## Problem Set #2: Supervised Learning II

<b>Problem 1 Logistic Regression: Training stability</b>
--

- (a) The most notable difference in training the logistic regression model on datasets  $A$  and  $B$  is that the algorithm does not converge on dataset  $B$ .
- (b) To investigate why the training procedure behaves unexpectedly on dataset  $B$ , but not on  $A$ , we print the value of  $\theta$  after every 10000 iterations. We notice that for data set  $B$ , although the normalized  $\frac{\theta}{\|\theta\|}$  almost stop changing after several tens of thousands of iterations, each component of the unnormalized  $\theta$  keeps increasing. We also notice that dataset  $A$  is not linearly separable while dataset  $B$  is linearly separable.

From the code, we notice that the algorithm calculates the gradient of loss function as

$$\nabla_{\theta} J(\theta) = -\frac{1}{m} \sum_{i=1}^m \frac{y^{(i)} x^{(i)}}{1 + \exp(y^{(i)} \theta^T x^{(i)})}.$$

From this, we know that the algorithm uses gradient descent to minimize the loss function

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \log \frac{1}{1 + \exp(-y^{(i)} \theta^T x^{(i)})}.$$

Hence, for a dataset that is linearly separable, that is,  $y^{(i)} \theta^T x^{(i)} > 0$  for all  $i$ , a  $\theta$  with larger norm always leads to a smaller loss, preventing the algorithm from converging. However, on a dataset that is not linearly separable, there exists  $i$  such that  $y^{(i)} \theta^T x^{(i)} < 0$ . By plotting  $f(z) = \log(1 + e^{-z})$  in Figure 1, we notice that negative margin dominates when scaling  $\theta$  to a larger norm. Hence, we cannot always increase  $\theta$  to a larger norm while minimizing  $J(\theta)$ .

- (c) Consider the following modifications
- i. Using a different constant learning rate will not make the algorithm converge on dataset  $B$ , since scaling  $\theta$  to larger norm still always decreases the loss.
  - ii. Decreasing the learning rate over time will make the algorithm converge for dataset  $B$ , since in this way the change of  $\theta$  converge to 0.

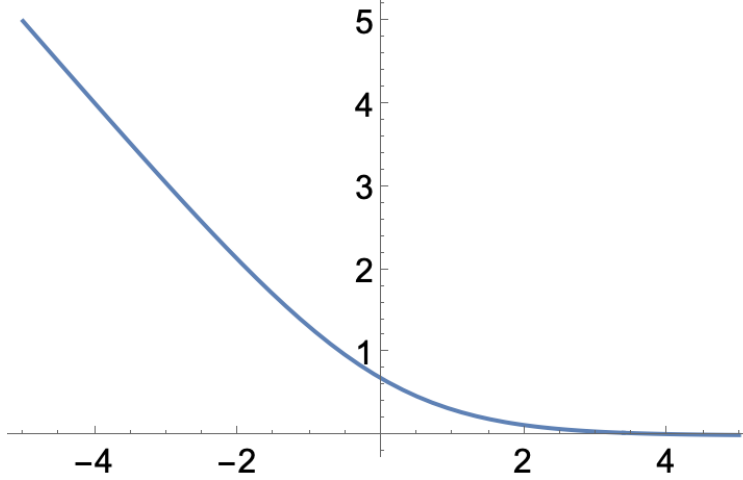


Figure 1: Plot of  $f(z) = \log(1 + e^{-z})$  for  $-5 \leq z \leq 5$ .

- iii. Linear scaling the input features does not help, since it does not change the dataset's linear separability.
  - iv. Adding a regularization term  $\|\theta\|_2^2$  helps, since now scaling  $\theta$  to larger norm penalize the algorithm.
  - v. Adding zero-mean Gaussian noise to the training data or labels helps as long as it makes the dataset not linearly separable.
- (d) Support vector machines, which uses hinge loss, are not vulnerable to datasets like  $B$ . In SVM, geometric margin is considered, instead of functional margin considered here. In other words,  $\theta$  is normalized, so for linearly separable datasets like  $B$ , the algorithm will still converge.

■

**Problem 2 Model Calibration**

Try to understand the output  $h_\theta(x)$  of the hypothesis function of a logistic regression model, in particular why we might treat the output as a probability.

When probabilities outputted by a model match empirical observation, the model is *well-calibrated*. For example, if a set of examples  $x^{(i)}$  for which  $h_\theta(x^{(i)}) \approx 0.7$ , around 70% of those examples should have positive labels. In a well-calibrated model, this property holds true at every probability value.

Suppose training set  $\{x^{(i)}, y^{(i)}\}_{i=1}^m$  with  $x^{(i)} \in \mathbb{R}^{n+1}$  and  $y^{(i)} \in \{0, 1\}$ . Assume we have an intercept term  $x^{(i)}_0 = 1$  for all  $i$ . Let  $\theta$  be the maximum likelihood parameters learned after training logistic regression model. In order for model to be well-calibrated, given any range of probabilities  $(a, b)$  such that  $0 \leq a < b \leq 1$ , and training examples  $x^{(i)}$  where the model outputs  $h_\theta(x^{(i)})$  fall in the range  $(a, b)$ , the fraction of positives in that set of examples should be equal to the average of the model outputs for those examples. That is,

$$\frac{\sum_{i \in I_{a,b}} P(y^{(i)} = 1 \mid x^{(i)}; \theta)}{|\{i \in I_{a,b}\}|} = \frac{\sum_{i \in I_{a,b}} \mathbb{I}\{y^{(i)} = 1\}}{|\{i \in I_{a,b}\}|},$$

where  $P(y^{(i)} = 1 \mid x; \theta) = h_\theta(x) = 1/(1 + \exp(-\theta^T x))$ ,  $I_{a,b} = \{i : h_\theta(x^{(i)}) \in (a, b)\}$ .

- (a) For the described logistic regression model over the range  $(a, b) = (0, 1)$ , we want to show the above equality holds. Recall the gradient of log-likelihood

$$\frac{\partial \ell}{\partial \theta_j} = \sum_{i=1}^m (y^{(i)} - h_\theta(x^{(i)})) x^{(i)}_j.$$

For a maximum likelihood estimation,  $\frac{\partial \ell}{\partial \theta} = 0$ . Hence  $\frac{\partial \ell}{\partial \theta_0} = 0$ . Since  $x^{(i)}_0 = 1$ , we have

$$\sum_{i=1}^m y^{(i)} - h_\theta(x^{(i)}) = 0.$$

The desired equality follows immediately since  $i \in I_{0,1}$  for all  $i$ .

- (b) A perfectly calibrated model — that is, the equality holds for any  $(a, b) \subset [0, 1]$  — does not imply that the model achieves perfect accuracy. Consider  $(a, b) = (\frac{1}{2}, 1)$ , the above equality implies

$$\frac{\sum_{i \in I_{a,b}} P(y^{(i)} = 1 \mid x^{(i)}; \theta)}{|\{i \in I_{a,b}\}|} = \frac{\sum_{i \in I_{a,b}} \mathbb{I}\{y^{(i)} = 1\}}{|\{i \in I_{a,b}\}|} < 1.$$

This shows that the model does not have perfect accuracy.

For the converse direction, a perfect accuracy does not imply perfectly calibrated. Consider again  $(a, b) = (\frac{1}{2}, 1)$ , then we have

$$\frac{\sum_{i \in I_{a,b}} \mathbb{I}\{y^{(i)} = 1\}}{|\{i \in I_{a,b}\}|} = 1 > \frac{\sum_{i \in I_{a,b}} P(y^{(i)} = 1 \mid x^{(i)}; \theta)}{|\{i \in I_{a,b}\}|}.$$

- (c) Discuss what effect of  $L_2$  regularization in the logistic regression objective has on model calibration. For  $L_2$  regularization in logistic regression, the gradient becomes

$$\frac{\partial \ell}{\partial \theta_j} = \sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x^{(i)}_j - 2C\theta_j = 0.$$

Hence, the equality does not hold unless  $\theta_0 = 0$ . ■

The interval  $(0, 1)$  is the only range for which logistic regression is guaranteed to be calibrated. When GLM assumptions hold, all ranges  $(a, b) \subset [0, 1]$  are well calibrated. In addition, when test set has same distribution and when model has not overfit or underfit, logistic regression are well-calibrated on test data as well. Thus logistic regression is popular when we are interested in level of uncertainty in the model output. △

### Problem 3 Bayesian Interpretation of Regularization

In Bayesian statistics, almost every quantity is a random variable. Joint distribution of all the random variables are called *model* (e.g.  $p(x, y, \theta)$ ). Every unknown quantity can be estimated by conditioning the model on all observed quantities. Such conditional distribution  $p(\theta | x, y)$  is called *posterior distribution*. A consequence of this approach is that we are required to endow a *prior distribution*  $p(\theta)$ .

In purest Bayesian interpretation, we are required to keep the entire posterior distribution over the parameters all the way until prediction to come up with the *posterior predictive distribution*, and the final prediction will be the EV of the posterior predictive distribution. However, this is computationally very expensive.

The compromise is to estimate a point value of the parameters instead of the full distribution, which is the mode of the posterior distribution. Estimating the mode of posterior distribution is also called *maximum a posteriori estimation* (MAP). That is,

$$\theta_{\text{MAP}} = \underset{\theta}{\operatorname{argmax}} p(\theta | x, y).$$

Compare this to the *maximum likelihood estimation* (MLE):

$$\theta_{\text{MLE}} = \underset{\theta}{\operatorname{argmax}} p(y | x, \theta).$$

In this problem, explore connections between MAP estimation and common regularization techniques that are applied with MLE estimation. In particular, we will show how choice of prior distribution over  $\theta$  is equivalent to different kinds of regularization.

- (a) Assume that  $p(\theta) = p(\theta | x)$ , we have

$$p(\theta | x, y) = \frac{p(y | \theta, x)p(\theta | x)}{p(y | x)} = \frac{p(y | \theta, x)p(\theta)}{p(y | x)}.$$

Since  $p(y \mid x)$  does not depend on  $\theta$ ,

$$\theta_{\text{MAP}} = \operatorname{argmax}_{\theta} p(y \mid \theta, x) p(\theta).$$

Note that the assumption  $p(\theta) = p(\theta \mid x)$  will be valid for models such as linear regression where the input  $x$  are not explicitly modeled by  $\theta$ . Note also that this means  $x$  and  $\theta$  are marginally independent, but not conditionally independent when  $y$  is given.

- (b) Now we show that MAP estimation with a zero-mean Gaussian priori over  $\theta$ , specifically  $\theta \sim \mathcal{N}(0, \eta^2 I)$ , is equivalent to applying  $L_2$  regularization with MLE estimation. Specifically, we need to show

$$\theta_{\text{MAP}} = \operatorname{argmin}_{\theta} -\log p(y \mid x, \theta) + \lambda \|\theta\|_2^2.$$

Recall the definition of multivariate normal, we have

$$\begin{aligned} \theta_{\text{MAP}} &= \operatorname{argmax}_{\theta} p(\theta \mid x, y) \\ &= \operatorname{argmax}_{\theta} p(y \mid \theta, x) p(\theta) \\ &= \operatorname{argmax}_{\theta} p(y \mid \theta, x) \exp\left(-\frac{1}{2\eta^2} \|\theta\|_2^2\right), \end{aligned}$$

where we have ignored some of the constants. Taking the negative log on both sides, it follows that

$$\theta_{\text{MAP}} = \operatorname{argmin}_{\theta} -\log p(y \mid x, \theta) + \frac{1}{2\eta^2} \|\theta\|_2^2,$$

as desired, where  $\lambda = \frac{1}{2\eta^2}$ .

- (c) Now consider a specific instance, a linear regression model given by  $y = \theta^T x + \varepsilon$ , where  $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ . Like before, assume a Gaussian prior on this model such that  $\theta \sim \mathcal{N}(0, \eta^2 I)$ . Let  $X$  be the design matrix of all training examples where each row is one example input, and  $y$  be the column vector of all the example outputs. We want to derive a closed form expression for  $\theta_{\text{MAP}}$ .

For this model, the likelihood of an example  $(x^{(i)}, y^{(i)})$  is

$$p(y^{(i)} \mid x^{(i)}, \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right).$$

Hence,

$$\log p(y^{(i)} \mid x^{(i)}, \theta) = -\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} + C,$$

where  $C$  is some constant, and  $\theta_{\text{MAP}}$  is given by

$$\begin{aligned}\theta_{\text{MAP}} &= \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^m \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} + \frac{1}{2\eta^2} \|\theta\|_2^2 \\ &= \underset{\theta}{\operatorname{argmin}} \frac{1}{2\sigma^2} (y - X\theta)^T (y - X\theta) + \frac{1}{2\eta^2} \theta^T \theta.\end{aligned}$$

Set the gradient of the function to 0, we have

$$0 = -\frac{1}{\sigma^2} X^T (y - X\theta_{\text{MAP}}) + \frac{1}{\eta^2} \theta_{\text{MAP}}$$

It follows that

$$\theta_{\text{MAP}} = \eta^2 (\eta^2 X^T X + \sigma^2 I)^{-1} X^T y.$$

(d) Now consider the Laplace distribution, whose density

$$f(z \mid \mu, b) = \frac{1}{2b} \exp\left(-\frac{|z - \mu|}{b}\right).$$

As before, consider a linear regression model given by  $y = \theta^T x + \varepsilon$  where  $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ . Assume a Laplace prior on this model where  $\theta \sim \mathcal{L}(0, bI)$ . We want to show that  $\theta_{\text{MAP}}$  in this case is equivalent to the solution of linear regression with  $L_1$  regularization, whose loss is specified as

$$J(\theta) = \|X\theta - y\|_2^2 + \gamma \|\theta\|_1.$$

Following the same approach,  $\theta_{\text{MAP}}$  for this model is given by

$$\theta_{\text{MAP}} = \underset{\theta}{\operatorname{argmin}} \frac{1}{2\sigma^2} \|y - X\theta\|_2^2 + \frac{1}{b} \|\theta\|_1.$$

Hence, this is equivalent to the solution of linear regression with  $L_1$  regularization, with  $\gamma = \frac{2\sigma^2}{b}$  in the above loss function  $J(\theta)$ .

**Note:** closed form solution for linear regression problem with  $L_1$  regularization does not exist. To optimize this, use gradient descent with a random initialization and solve it numerically. ■

- Linear regression with  $L_2$  regularization is also called *Ridge regression*, and  $L_1$  regularization is called *Lasso regression*. These regularizations can be applied to any generalized linear models just as above. Regularization techniques are also called *weight decay* and *shrinkage*. The Gaussian and Laplace priors encourages the parameter values to be closer to their mean (i.e. zero), which results in the shrinkage effect.

- Lasso regression is known to result in sparse parameters.

△

#### Problem 4 Constructing kernels

Choosing kernel  $K(x, z) = \phi(x)^T \phi(z)$ . Mercer's theorem tells us  $K$  is a Mercer kernel iff for any finite set  $\{x^{(i)}\}_{i=1}^m$ , the square matrix  $K \in \mathbb{R}^{m \times m}$  whose entries are  $K_{ij} = K(x^{(i)}, x^{(j)})$  is symmetric and positive semidefinite.

Let  $K_1, K_2$  be kernels over  $\mathbb{R}^n \times \mathbb{R}^n$ , let  $a \in \mathbb{R}^+$  be a positive real number, let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be a real-valued function, let  $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^d$  be a function mapping from  $\mathbb{R}^n$  to  $\mathbb{R}^d$ , let  $K_3$  be a kernel over  $\mathbb{R}^d \times \mathbb{R}^d$ , and let  $p$  a polynomial with *positive* coefficients.

For each  $K$  below, prove or give counter example to show whether it is necessarily a kernel.

- (a)  $K(x, z) = K_1(x, z) + K_2(x, z)$  is a kernel. For any finite set  $\{x^{(i)}\}_{i=1}^m$ , the matrix  $K_1$  and  $K_2$  are both symmetric and PSD. Hence, for  $K = K_1 + K_2$  is also symmetric and PSD.
- (b)  $K(x, z) = K_1(x, z) - K_2(x, z)$  is not necessarily a kernel. Consider matrix

$$K_1 = \text{diag}(1, 1, \dots, 1), \quad K_2 = \text{diag}(2, 2, \dots, 2).$$

Then

$$K = \text{diag}(-1, -1, \dots, -1)$$

is not PSD.

- (c)  $K(x, z) = aK_1(x, z)$  is a kernel.
- (d)  $K(x, z) = -aK_1(x, z)$  is not a kernel.
- (e)  $K(x, z) = K_1(x, z)K_2(x, z)$  is a kernel. For any finite set  $\{x^{(i)}\}_{i=1}^m$ , the matrix  $K$  will be an elementwise product of  $K_1$  and  $K_2$ . Hence  $K$  is symmetric. Now we prove  $K$  is PSD. We have

$$\begin{aligned} z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\ &= \sum_i \sum_j z_i K_{1ij} K_{2ij} z_j \\ &= \sum_i \sum_j z_i \sum_k \phi_k^1(x^{(i)}) \phi_k^1(x^{(j)}) \sum_l \phi_l^2(x^{(i)}) \phi_l^2(x^{(j)}) z_j \\ &= \sum_k \sum_l (z_i \phi_k^1(x^{(i)}) \phi_l^2(x^{(i)}))^2 \\ &\geq 0, \end{aligned}$$

as desired.

- (f)  $K(x, z) = f(x)f(z)$  is a kernel. For any finite set  $\{x^{(i)}\}_{i=1}^m$ , the matrix  $K$  is clearly symmetric. Now, we have

$$\begin{aligned} z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\ &= \sum_i \sum_j z_i f(x^{(i)}) f(x^{(j)}) z_j \\ &= \left( \sum_i z_i f(x^{(i)}) \right)^2 \\ &\geq 0. \end{aligned}$$

Hence,  $K$  is also PSD.

- (g)  $K(x, z) = K_3(\phi(x), \phi(z))$  is a kernel. Suppose the feature map corresponds to  $K_3$  is  $\psi$ , then

$$K(x, z) = (\psi \circ \phi(x))^T (\psi \circ \phi(z)).$$

Hence,  $K$  is a kernel corresponds to feature map  $\psi \circ \phi$ .

- (h)  $K(x, z) = p(K_1(x, z))$  is a kernel. Let  $p(t) = \sum_k a_k t^k$ . Then,  $K(x, z) = \sum_k a_k K_1(x, z)^k$ . By part (e),  $K_1(x, z)^k$  is a kernel for each  $k$ . Since  $a_k > 0$ ,  $K(x, z) = \sum_k a_k K_1(x, z)^k$  is a kernel. ■

### Problem 5 Kernelizing the Perceptron

Binary classification problem with  $y \in \{0, 1\}$ . The perceptron uses hypothesis of the form  $h_\theta(x) = g(\theta^T x)$ , where  $g(z) = \text{sgn } z = 1$  if  $z \geq 0$ , 0 otherwise. Consider stochastic gradient descent-like implementation of perceptron algorithm, where each update to  $\theta$  is made using one training example. However, unlike stochastic gradient descent, the perceptron algorithm will only make one pass through the entire training set. Update rule is

$$\theta^{(i+1)} := \theta^{(i)} + \alpha(y^{(i+1)} - h_{\theta^{(i)}}(x^{(i+1)}))x^{(i+1)},$$

where  $\theta^{(i)}$  is the value of parameters after the algorithm has seen the first  $i$  training examples. Prior to seeing any training examples,  $\theta(0)$  is initialized to 0.

- (a) Let  $K$  be a Mercer kernel corresponding to some very high-dimensional feature mapping  $\phi$ . Suppose  $\phi$  is so high-dimensional that it is infeasible to ever represent  $\phi(x)$  explicitly. Describe how to apply "kernel trick" to the perceptron to make it work in the high-dimensional feature space  $\phi$ , but without ever explicitly computing  $\phi(x)$ .

No need to worry about the intercept term. If you like, think of  $\phi$  as having the property that  $\phi_0(x) = 1$  so that this is taken care of.



- i. To represent the high-dimensional parameter vector  $\theta$  implicitly, we will represent it as a linear combination of  $\phi(x^{(i)})$ , where  $x^{(i)}$  is the training examples. That is,

$$\theta^{(i)} = \sum_k \beta_k^{(i)} \phi(x^{(k)}).$$

In particular, the initial value  $\theta^{(0)}$  is represented by setting  $\beta_k = 0$  for all  $i$ .

- ii. To efficiently make a prediction on a new input  $x^{(i+1)}$ , we compute

$$\begin{aligned} h_{\theta^{(i)}}(x^{(i+1)}) &= g((\theta^{(i)})^T \phi(x^{(i+1)})) \\ &= g\left(\left(\sum_k \beta_k^{(i)} \phi(x^{(k)})\right)^T \phi(x^{(i+1)})\right) \\ &= g\left(\sum_k \beta_k^{(i)} K(x^{(k)}, x^{(i+1)})\right). \end{aligned}$$

- iii. For a new training example  $(x^{(i+1)}, y^{(i+1)})$ , consider the original update rule

$$\theta^{(i+1)} := \theta^{(i)} + \alpha(y^{(i+1)} - h_{\theta^{(i)}}(x^{(i+1)}))x^{(i+1)}.$$

For our representation, we have

$$\sum_k \beta_k^{(i+1)} \phi(x^{(k)}) := \sum_k \beta_k^{(i)} \phi(x^{(k)}) + \alpha(y^{(i+1)} - h_{\theta^{(i)}}(x^{(i+1)}))\phi(x^{(i+1)}).$$

Hence, the modified update rule is

$$\beta_{i+1}^{(i+1)} := \beta_{i+1}^{(i)} + \alpha\left(y^{(i+1)} - g\left(\sum_k \beta_k^{(i)} K(x^{(k)}, x^{(i+1)})\right)\right)$$

for  $\beta_{i+1}$ , and

$$\beta_k^{(i+1)} := \beta_k^{(i)}$$

for  $\beta_k$  with  $k \neq i + 1$ .

(b) **Coding problem.**

- (c) The dot kernel performs extremely poorly, since there is no feature mapping and it is equivalent to logistic regression, while the dataset is not linearly separable. ■

**Problem 6 Spam classification**

Use Bayes algorithm and an SVM to build a spam classifier to detect SMS spam messages.

- (a) **Coding problem.** Implement code for processing the spam messages into numpy arrays.

- (b) **Coding problem.** Implement a naive Bayes classifier for spam classification with multinomial event model and Laplace smoothing.

For multinomial event model, the model assume the probability that word  $k$  appears in some location is  $\phi_{k|y=1}$  given  $y = 1$  and  $\phi_{k|y=0}$  given  $y = 0$ . That is

$$\phi_{k|y=1} = p(x_j = k \mid y = 1) \quad \phi_{k|y=0} = p(x_j = k \mid y = 0),$$

for  $k$  that index into the dictionary size  $|V|$  and  $j$  that index into the lenght of the message. Also, the model assume the probability of a spam email  $\phi_y = p(y)$ . Hence, the log-likelihood for Naive Bayes algorithm

$$\begin{aligned} \ell(\phi_y, \phi_{k|y=1}, \phi_{k|y=0}) &= \log \prod_{i=1}^m \left( \prod_{j=1}^{d_i} p(x_j^{(i)} \mid y) \right) p(y^{(i)}) \\ &= \sum_{i=1}^m \sum_{j=1}^{d_i} \left( \log p(x_j^{(i)} \mid y) \right) + \log p(y^{(i)}). \end{aligned}$$

and the maximum likelihood estimation with Laplace smoothing

$$\begin{aligned} \phi_y &= \frac{\sum_{i=1}^m \mathbb{I}\{y^{(i)} = 1\}}{m}, \\ \phi_{k|y=0} &= \frac{\sum_{i=1}^m \sum_{j=1}^{d_i} \mathbb{I}\{x_j^{(i)} = k \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^m \mathbb{I}\{y^{(i)} = 1\} d_i + |V|}, \\ \phi_{k|y=0} &= \frac{\sum_{i=1}^m \sum_{j=1}^{d_i} \mathbb{I}\{x_j^{(i)} = k \wedge y^{(i)} = 0\} + 1}{\sum_{i=1}^m \mathbb{I}\{y^{(i)} = 0\} d_i + |V|}. \end{aligned}$$

Notice that  $\sum_{j=1}^{d_i} \mathbb{I}\{x_j^{(i)} = k \wedge y^{(i)} = 1\}$  is just the count of word  $k$  in dictionary for message  $x^{(i)}$ , so we only need to count the number of each words in each message.

To predict from the Naive Bayes model, we need to compare

$$p(y = 1 \mid x) = \frac{p(x \mid y = 1)p(y = 1)}{p(x)}, \quad p(y = 0 \mid x) = \frac{p(x \mid y = 0)p(y = 0)}{p(x)}.$$

When calculating  $p(x \mid y) = \prod_j p(x_j \mid y)$ , underflow may happen since every term is smaller than one. We resolve this issue by using logarithm

$$\begin{aligned} \log p(x \mid y)p(y) &= \log \prod_j p(x_j \mid y)p(y) \\ &= \sum_j \log p(x_j \mid y) + \log p(y). \end{aligned}$$

and compare

$$\begin{aligned} \log p(y = 1 \mid x) &= \sum_j \log p(x_j \mid y = 1) + \log p(y = 1) - \log p(x), \\ \log p(y = 0 \mid x) &= \sum_j \log p(x_j \mid y = 0) + \log p(y = 0) - \log p(x). \end{aligned}$$

by taking the difference.

- (c) **Coding problem.** Some tokens may be particularly indicative of an SMS being in a particular class. Try to get an informal sense of how indicative token  $k$  is for the SPAM class by looking at

$$\log \frac{p(x_j = k \mid y = 1)}{p(x_j = k \mid y = 0)} = \log \frac{P(\text{token } k \mid \text{email is SPAM})}{P(\text{token } k \mid \text{email is NOTSPAM})}.$$

- (d) **Coding problem.** Select the appropriate kernel radius for SVM parameterized by an RBF kernel. That is,

$$K(x, z) = \exp(-\gamma \|x - z\|^2).$$

■