1 [23 points] Logistic regression

Consider the log-likelihood function for logistic regression:

$$\ell(\mathbf{w}) = \sum_{i=1}^{N} y^{(i)} \log h(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log(1 - h(\mathbf{x}^{(i)})),$$

where $h(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x}) = \dfrac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$.

(a) Find the Hessian of $\ell(w)$

We proved in class that

$$\nabla l(w) = \sum_{i=1}^{N} (y^{(i)} - h^{(i)}) x^{(i)}$$

where $h^{(i)} = \sigma(w^T x^{(i)})$

and $\nabla_w (h^{(i)}) = h^{(i)}(1 - h^{(i)}) x^{(i)}$

Now $H_l = \frac{\partial}{\partial w}(\nabla l)(w) = -\sum_{i=1}^{N} \frac{\partial}{\partial w}\left(h(x^{(i)}) x^{(i)}\right)$

Since $h(x^{(i)}) x^{(i)} = \begin{bmatrix} h(x^{(i)}) x_1^{(i)} \\ \vdots \\ h(x^{(i)}) x_m^{(i)} \end{bmatrix}$ ($\leftarrow m \rightarrow$)

$$\Rightarrow \frac{\partial}{\partial w}\left(h(x^{(i)}) x^{(i)}\right) = \begin{bmatrix} \frac{\partial (h(x^{(i)}) x_1^{(i)})}{\partial w} \\ \vdots \\ \frac{\partial (h(x^{(i)}) x_m^{(i)})}{\partial w} \end{bmatrix} = \begin{bmatrix} x_1^{(i)} \frac{\partial h(x^{(i)})}{\partial w} \\ \vdots \\ x_m^{(i)} \frac{\partial h(x^{(i)})}{\partial w} \end{bmatrix}$$

$$= x^{(i)} \frac{\partial h(x^{(i)})}{\partial w}$$

its transpose: $\frac{\partial}{\partial w}(h(x^{(i)})x^{(i)})^T = \frac{\partial h(x^{(i)})^T}{\partial w} x^{(i)T} = \nabla_w h^{(i)} x^{(i)T}$

$= h^{(i)}(1-h^{(i)}) x^{(i)} x^{(i)T}$

$\Rightarrow H = H^T = -\sum_{i=1}^{N} \frac{\partial}{\partial w}(h(x^{(i)})x^{(i)})^T$

$= -\sum_{i=1}^{N} h(x^{(i)})(1-h(x^{(i)})) x^{(i)} x^{(i)T}$

(vectorized version:

$z = Xw, \; h = \sigma(z) = \frac{1}{1+e^{-z}}$

$\nabla L(w) = X^T(h-y)$

let $R := \begin{pmatrix} h^{(1)}(1-h^{(1)}) & & \\ & \ddots & \\ & & h^{(N)}(1-h^{(N)}) \end{pmatrix}$

$\Rightarrow H = X^T R X$)

(b) Prove the H is negative semi-definite

[6 points] Show that H is negative semi-definite and thus ℓ is concave and has no local maxima other than the global one. That is, show that

$$\mathbf{z}^\top H \mathbf{z} \leq 0$$

for any vector $\mathbf{z}$. [*Hint: You might want to start by showing the fact that* $\sum_i \sum_j z_i x_i x_j z_j = (\mathbf{x}^\top \mathbf{z})^2$. *Note that* $(\mathbf{x}^\top \mathbf{z})^2 \geq 0$.]

1(b) Goal: to show $z^T H z \le 0 \quad \forall z \in \mathbb{R}^n$

Pf Let $z \in \mathbb{R}^n$

$$z^T H z = z^T \left(-\sum_{i=1}^{N} h(x^{(i)})(1-h(x^{(i)}))\, x^{(i)} x^{(i)T} \right) z$$

$$= -\sum_{i=1}^{N} h(x^{(i)})(1-h(x^{(i)}))\, z^T x^{(i)} x^{(i)T} z \quad \text{by linearity of quadratic forms}$$

$$z^T x^{(i)} x^{(i)T} z = \underbrace{z^T x^{(i)}}_{\text{scalar}} \left(z^T x^{(i)}\right)^T = \left(z^T x^{(i)}\right)^2 \ge 0$$

and note that $h(x^{(i)}) \in (0,1)$ by the property of sigmoid function

$$\Rightarrow z^T H z = -\sum_{i=1}^{N} \underbrace{h(x^{(i)})}_{(0,1)} \underbrace{(1-h(x^{(i)}))}_{(0,1)} \underbrace{\left(z^T (x^{(i)})\right)^2}_{\ge 0} \le 0$$

This completes the proof that H is negative semi-definite (thus L is concave, any local maxima is global) □

(c) updae rule by Newton's method

(c) **[2 points]** Using the H you calculated in part (a), write down the update rule implied by Newton's method for optimizing $\ell(\mathbf{w})$. [*Hint: it could be a single-line equation:* $\mathbf{w} = YOUR\_ANSWER$.]

1(c) update rule by Newton's method:

$$w := w - H^{-1}(w) \nabla L(w)$$

$$= w + \left(\sum_{i=1}^{N} h(x^{(i)})(1-h(x^{(i)}))\, x^{(i)} x^{(i)T} \right)^{-1} \left(\sum_{i=1}^{N} \left(y^{(i)} - h(x^{(i)})\right) x^{(i)} \right)$$

(d) **[8 points]** (Autograder) Now use the update rule in (c) (and not a library function) to implement Newton's method and apply it to binary classification problem, following the guide in `logistic_regression.ipynb`. Your `ipynb` file **SHOULD** include all the outputs.

(e) coefficients from the code

[**2 points**] What are the coefficients **w**, including the intercept term, resulting from your code? Please provide your answer in your **writeup**.

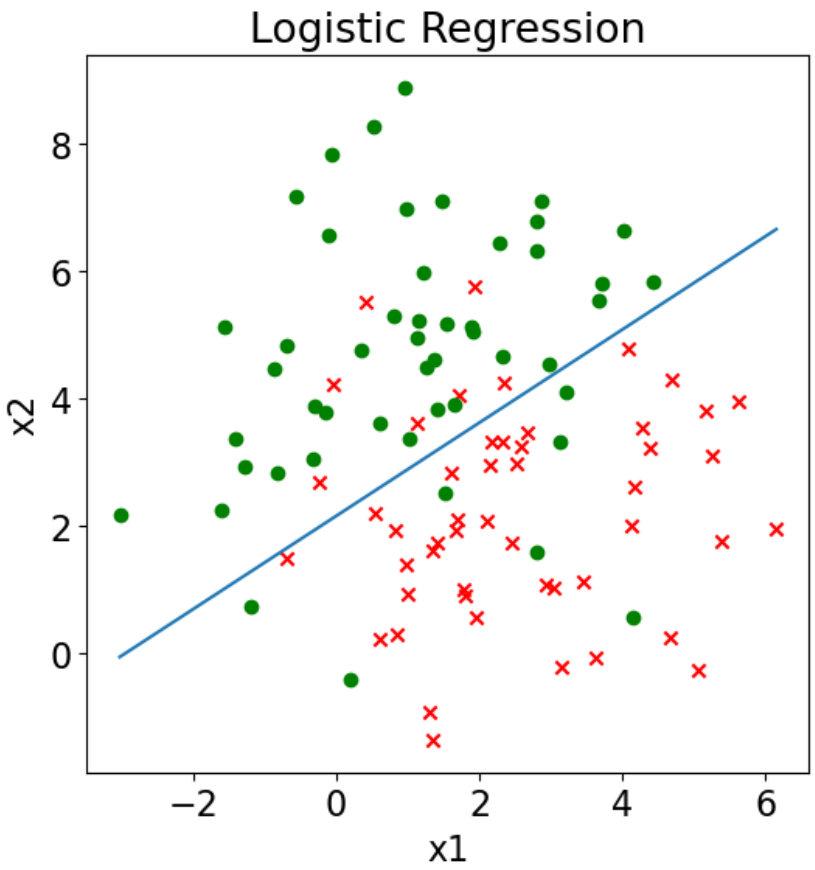
Answer:

```
(3,) [-1.84922892 -0.62814188  0.85846843]
Difference:  1.6653345369377348e-15
Good! The two methods match in distance!
```

(f) plot from the code

Please share the final plot from `logistic_regression.ipynb` in your **writeup**.

Answer:



2 [27 points] Softmax Regression via Gradient Ascent

Gradient ascent is an algorithm used to find parameters that maximize a certain expression (contrary to gradient descent, which is used to minimize an expression). For some function $f(\mathbf{w})$, gradient ascent finds $\mathbf{w}^* = \arg\max_{\mathbf{w}} f(\mathbf{w})$ according to the following pseudo-code:

Algorithm 1 Gradient Ascent

$\mathbf{w}^* \leftarrow$ *random*
repeat
    $\mathbf{w}^* \leftarrow \mathbf{w}^* + \alpha \nabla_{\mathbf{w}} f(\mathbf{w}^*)$
until *convergence*
return $\mathbf{w}^*$

Softmax regression is a multiclass classification algorithm. Given a labeled dataset $D = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), ..., (\mathbf{x}^{(N)}, y^{(N)})\}$, where $y^{(i)} \in \{1, 2, ..., K\}$ (total K classes), softmax regression computes the probability that an example $\mathbf{x}$ belongs to a class k:

$$p(y = k|\mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}_k^\top \phi(\mathbf{x}))}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \phi(\mathbf{x}))}$$

where $\mathbf{w}_k$ is a weight vector for class k. The above expression is over-parametrized, meaning that there is more than one unique $\{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_K\}$ that gives identical probability measures for $p(y = k|\mathbf{x}, \mathbf{w})$. An unique solution can be obtained using only $K-1$ weight vectors $\mathbf{w} = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_{K-1}\}$ and fixing $\mathbf{w}_K = 0$:

$$p(y = k|\mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}_k^\top \phi(\mathbf{x}))}{1 + \sum_{j=1}^{K-1} \exp(\mathbf{w}_j^\top \phi(\mathbf{x}))}; \;\; \forall k = \{1, 2, ..., K-1\}$$

$$p(y = K|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(\mathbf{w}_j^\top \phi(\mathbf{x}))}$$

We define the likelihood of the ith training example $p(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w})$ as:

$$p(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}) = \prod_{k=1}^{K} \Big[p(y^{(i)} = k|\mathbf{x}^{(i)}, \mathbf{w})\Big]^{\mathbb{I}\{y^{(i)}=k\}}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function. We define the likelihood as:

$$L(\mathbf{w}) = \prod_{i=1}^{N} p(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}) = \prod_{i=1}^{N}\prod_{k=1}^{K} \Big[p(y^{(i)} = k|\mathbf{x}^{(i)}, \mathbf{w})\Big]^{\mathbb{I}\{y^{(i)}=k\}}$$

Finally, we define the log-likelihood as:

$$l(\mathbf{w}) = \log L(\mathbf{w}) = \sum_{i=1}^{N}\sum_{k=1}^{K} \log(\Big[p(y^{(i)} = k|\mathbf{x}^{(i)}, \mathbf{w})\Big]^{\mathbb{I}\{y^{(i)}=k\}})$$

(a) Derive the gradient ascent update rule for log-likelihood

[11 points] Derive the gradient ascent update rule for the log-likelihood of the training data. In other words, derive the expression $\nabla_{\mathbf{w}_m} l(\mathbf{w})$ for $m = 1, ..., K-1$. Show that:

$$\nabla_{\mathbf{w}_m} l(\mathbf{w}) = \sum_{i=1}^{N} \phi(\mathbf{x}^{(i)}) \left[\mathbb{I}\{y^{(i)} = m\} - \frac{\exp(\mathbf{w}_m^\top \phi(\mathbf{x}^{(i)}))}{1 + \sum_{j=1}^{K-1} \exp(\mathbf{w}_j^\top \phi(\mathbf{x}^{(i)}))}\right]$$

$$= \sum_{i=1}^{N} \phi(\mathbf{x}^{(i)}) \Big[\mathbb{I}\{y^{(i)} = m\} - p(y^{(i)} = m|\mathbf{x}^{(i)}, \mathbf{w})\Big]$$

Remember that in this notation, $\mathbf{w}_k$ is a weight vector for class k, **not** the kth element of $\mathbf{w}$.

[Hints: $\log a^b = b\log(a)$. Further, it helps to consider the two cases separately; a case for $y^{(i)} = k = m$, and another for $y^{(i)} = k \neq m$, which is equivalent to using Kronecker delta δ_{km}].

2(a) Proof

$$\log p(y^{(i)}=k \mid x^{(i)}, w)^{\mathbb{I}\{y^{(i)}=k\}}$$

$$= \mathbb{I}\{y^{(i)}=k\} \log p(y^{(i)}=k \mid x^{(i)}, w)$$

$$\log p(y^{(i)}=k \mid x^{(i)}, w) = \log \exp(w_k^T \varphi(x)) - \log\left(\sum_{j=1}^{K} \exp(w_j^T \varphi(x^{(i)}))\right)$$

$$= w_k^T \varphi(x) - \log \sum_{j=1}^{k} \exp(w_j^T \varphi(x^{(i)}))$$

note $\nabla_{w_m}(w_k^T \varphi(x^{(i)})) \neq 0$ iff $m=k$, so

$$\nabla_m(w_k^T \varphi(x^{(i)})) = \delta_{km} \varphi(x^{(i)})$$

and $\nabla_{w_m} \log\left(\sum_{j=1}^{k} \exp(w_j^T \varphi(x^{(i)}))\right) = \frac{1}{\sum_{j=1}^{K} \exp(\dots)} \frac{\partial}{\partial w_m} \sum_{j=1}^{k} \exp(\dots)$

$$= \frac{1}{\sum_{j=1}^{K} \exp(\dots)} \exp(w_m^T \varphi(x^{(i)})) \varphi(x^{(i)})$$

$$= p(y=m \mid x^{(i)}, w) \varphi(x^{(i)})$$

$$\Rightarrow \nabla_{w_m} \log p(y^{(i)}=k|x^{(i)},w) = \delta_{km}\varphi(x^{(i)}) - p(y=m|x^{(i)},w)\varphi(x^{(i)})$$

($\delta_{km} = \mathbb{I}\{y^{(i)}=m\}$ here)

$$= \varphi(x^{(i)})\left(\mathbb{I}\{y^{(i)}=m\} - p(y=m|x^{(i)},w)\right)$$

$$\Rightarrow \nabla_{w_m} \log p(y^{(i)}=k|x^{(i)},w)^{\mathbb{I}\{y^{(i)}=k\}} = \begin{cases} 0, & m \neq k \\ \mathbb{I}\{y^{(i)}=m\} - p(y=m|x^{(i)},w)\varphi(x^{(i)}), & m=k \end{cases}$$

$$\Rightarrow \nabla_{w_m} l(w) = \sum_{i=1}^{N}\sum_{k=1}^{K} \nabla_{w_m} \log p(y^{(i)}=k|x^{(i)},w)^{\mathbb{I}\{y^{(i)}=k\}}$$

$$= \sum_{i=1}^{N} \nabla_{w_m} \log p(y^{(i)}=m|x^{(i)},w)$$

$$= \sum_{i=1}^{N} \varphi(x^{(i)})\left[\mathbb{I}\{y^{(i)}=m\} - p(y=m|x^{(i)},w)\right]$$

$$\left(= \sum_{i=1}^{N} \varphi(x^{(i)})\left[\mathbb{I}\{y^{(i)}=m\} - \frac{\exp(w_m^T\varphi(x^{(i)}))}{1+\sum_{j=1}^{K-1}\exp(w_j^T\varphi(x^{(i)}))}\right]\right)\square$$

(b) **[14 points]** (Autograder) Using the gradient computed in part (a), implement gradient ascent for softmax regression, following the guide in `softmax_regression.ipynb`. You are required to implement your code in `softmax_regression.py`. Make sure to include all the outputs in your `softmax_regression.ipynb` file. Recall that softmax regression classifies an example $\mathbf{x}$ as:

$$y = \arg\max_{y'} p(y'|\mathbf{x}, \mathbf{w})$$

(c) report accuracy

[2 points] When you train your classifier on the training data in (b), what is the accuracy on the test data? Please report your accuracy in your **writeup**.

```
(3, 4)
The accuracy of Softmax Regression - our implementation: 94.00%
```

3 Gaussian Discriminate Analysis

Suppose we are given a dataset $\{(\mathbf{x}^{(i)}, y^{(i)}); i = 1, ..., N\}$ consisting of N independent examples, where $\mathbf{x}^{(i)} \in \mathbb{R}^M$ are M-dimensional vectors, and $y^{(i)} \in \{0, 1\}$. We will model the joint distribution of $(\mathbf{x}^{(i)}, y^{(i)})$ using:

$$p(y^{(i)}) = \phi^{y^{(i)}}(1-\phi)^{1-y^{(i)}}$$

$$p(\mathbf{x}^{(i)}|y^{(i)}=0) = \frac{1}{(2\pi)^{\frac{M}{2}}|\Sigma|^{\frac{1}{2}}} \exp(-\frac{1}{2}(\mathbf{x}^{(i)}-\mu_0)^\top \Sigma^{-1}(\mathbf{x}^{(i)}-\mu_0))$$

$$p(\mathbf{x}^{(i)}|y^{(i)}=1) = \frac{1}{(2\pi)^{\frac{M}{2}}|\Sigma|^{\frac{1}{2}}} \exp(-\frac{1}{2}(\mathbf{x}^{(i)}-\mu_1)^\top \Sigma^{-1}(\mathbf{x}^{(i)}-\mu_1))$$

Here, the parameters of our model are ϕ, Σ, μ_0, and μ_1. (Note that while there are two different mean vectors μ_0 and μ_1, there is only one covariance matrix Σ.)

(a) $p(y = 1|x; \phi, \Sigma, \mu_0, \mu_1)$ is a form of logistic function

[4 points] Suppose we have already fit ϕ, Σ, μ_0, and μ_1, and now want to make a prediction at some new query point $\mathbf{x}$. Show that the posterior distribution of the label (y) at $\mathbf{x}$ takes the form of a logistic function, and can be written as

$$p(y = 1|\mathbf{x}; \phi, \Sigma, \mu_0, \mu_1) = \frac{1}{1 + \exp(-\mathbf{w}^\top \widehat{\mathbf{x}})}$$

where $\widehat{\mathbf{x}}$ to be $M+1$ dimensional vectors ($\widehat{\mathbf{x}}^{(i)} \in \mathbb{R}^{M+1}$) by adding an extra coordinate $\widehat{\mathbf{x}}_0 = 1$ from the original $\mathbf{x}$ and $\mathbf{w}$ is a function of ϕ, Σ, μ_0, and μ_1.

By Bayes, $p(y=1|x) = \frac{p(x|y=1)p(y=1)}{p(x|y=1)p(y=1)+p(x|y=0)p(y=0)}$

$$= \frac{1}{1+\frac{p(x|y=0)p(y=0)}{p(x|y=1)p(y=1)}}$$

$$\frac{p(x|y=0)p(y=0)}{p(x|y=1)p(y=1)} = \frac{1-\varphi}{\varphi}\exp\Big\{-\frac{1}{2}(x^{(i)}-\mu_1)^T\Sigma^{-1}(x^{(i)}-\mu_0) + \frac{1}{2}(x^{(i)}-\mu_1)^T\Sigma^{-1}(x^{(i)}-\mu_1)\Big\}$$

$$= \frac{1-\varphi}{\varphi}\exp\Big\{-\frac{1}{2}\big(x^T\Sigma^{-1}x - 2\mu_0^T\Sigma^{-1}x + \mu_0^T\Sigma^{-1}\mu_0\big) + \frac{1}{2}\big(x^T\Sigma^{-1}x - 2\mu_1^T\Sigma^{-1}x + \mu_1^T\Sigma^{-1}\mu_1\big)\Big\}$$

$$= \frac{1-\varphi}{\varphi}\exp\Big\{(\mu_0^T-\mu_1^T)\Sigma^{-1}x - \frac{1}{2}(\mu_0^T\Sigma^{-1}\mu_0 - \mu_1^T\Sigma^{-1}\mu_1)\Big\}$$

$$\Rightarrow \log\frac{p(x|y=0)p(y=0)}{p(x|y=1)p(y=1)} = \log\frac{1-\varphi}{\varphi} + (\mu_0-\mu_1)^T\Sigma^{-1}x + \frac{1}{2}(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0)$$

Define: $w_0 := -\log\frac{1-\varphi}{\varphi} - \frac{1}{2}(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0) \in \mathbb{R}$

$w_1 := -\Sigma^{-1}(\mu_1-\mu_0) \in \mathbb{R}^n$

And let $\hat{x} := \begin{bmatrix} x \\ 1 \end{bmatrix}$, $w := \begin{bmatrix} w_1 \\ w_0 \end{bmatrix}$

$$\Rightarrow \log\frac{p(x|y=0)p(y=0)}{p(x|y=1)p(y=1)} = -w^T\hat{x}$$

Then $p(y=1|x) = \frac{1}{1+\exp(-w^T\hat{x})}$

(b) MLE of ϕ, μ_0, μ_1

[14 points] The maximum likelihood estimates of ϕ, μ_0, and μ_1 are given by

$$\phi_{\text{ML}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\{y^{(i)} = 1\}$$

$$\mu_{0_{\text{ML}}} = \frac{\sum_{i=1}^{N} \mathbb{I}\{y^{(i)} = 0\}\mathbf{x}^{(i)}}{\sum_{i=1}^{N} \mathbb{I}\{y^{(i)} = 0\}}$$

$$\mu_{1_{\text{ML}}} = \frac{\sum_{i=1}^{N} \mathbb{I}\{y^{(i)} = 1\}\mathbf{x}^{(i)}}{\sum_{i=1}^{N} \mathbb{I}\{y^{(i)} = 1\}}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function. The log-likelihood of the data is

$$\ell(\phi, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^{N} p(\mathbf{x}^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma)$$

$$= \log \prod_{i=1}^{N} p(\mathbf{x}^{(i)} | y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi)$$

By maximizing ℓ with respect to the three parameters (ϕ, μ_0, and μ_1), **prove that** the maximum likelihood estimates of ϕ, μ_0, and μ_1 are indeed the ones described above. (You may assume that there is at least one positive and one negative example, so that the denominators in the definitions of μ_0 and μ_1 above are non-zero.)

$$l(\varphi, \mu_0, \mu_1, \Sigma) = \log p(x, y; \varphi, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^{N} p\left(x^{(i)}, y^{(i)}; \varphi, \mu_0, \mu_1, \Sigma\right)$$

$$= \sum_{i=1}^{N} \log\left[p\left(y^{(i)}; \varphi\right) p\left(x | y^{(i)}; \mu_0, \mu_1, \Sigma\right)\right]$$

$$= \sum_{i=1}^{N} \log p\left(y^{(i)}; \varphi\right) + \sum_{i=1}^{N} \log p\left(x^{(i)} | y^{(i)}; \mu_0, \mu_1, \Sigma\right)$$

To solve φ_{ML}:

$$\nabla_\varphi l = \nabla_\varphi \sum_{i=1}^{N} \log p(y^{(i)};\varphi) \quad \text{+0}$$

$$= \sum_{i=1}^{N} \nabla_\varphi \log\left(\varphi^{y^{(i)}}(1-\varphi)^{1-y^{(i)}}\right)$$

$$= \sum_{i=1}^{N} \nabla_\varphi \left(y^{(i)}\log\varphi + (1-y^{(i)})\log(1-\varphi)\right) = \sum_{i=1}^{N} \frac{y^{(i)}}{\varphi} - \frac{1-y^{(i)}}{1-\varphi}$$

$$= \frac{1}{\varphi}\underbrace{\sum_{i=1}^{N} y^{(i)}}_{\text{set it as } N_1} - \frac{1}{1-\varphi}\underbrace{\sum_{i=1}^{N}(1-y^{(i)})}_{\text{set it as } N_0}$$

Note $N_0 + N_1 = \sum_{i=1}^{N} 1 = N$

Set $\nabla_\varphi l := 0 \Rightarrow \frac{1}{\varphi} N_1 = \frac{1}{1-\varphi} N_0 \Rightarrow N_1 - N_0\varphi = N_1\varphi$

$$\Rightarrow \varphi_{ML} = \frac{N_1}{N_0+N_1} = \frac{1}{N}\sum_{i=1}^{N} y^{(i)} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\{y^{(i)}=1\}$$

To solve $\mu_{0_{ML}}$:

since μ_0 only contribute to $p(x^{(i)}|y^{(i)}=0)$ for each i,

$$\arg\max_{\mu_0} l = \arg\max_{\mu_0} \sum_{i:y^{(i)}=0} \log p(x^{(i)}|y^{(i)}=0;\mu_0,\mu_1,\Sigma)$$

$$= \arg\max_{\mu_0} \sum_{i:y^{(i)}=0} \log \frac{1}{2\pi M^{\frac{M}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x^{(i)}-\mu_y)^T\Sigma^{-1}(x^{(i)}-\mu_0)\right)$$

$$= \arg\max_{\mu_0} \sum_{i:y^{(i)}=0} -\frac{1}{2}(x^{(i)}-\mu_y)^T\Sigma^{-1}(x^{(i)}-\mu_0) \quad ①$$

We define ① as l_{μ_0}, and take

$$\frac{\partial l_{\mu_0}}{\partial \mu_0} = -\frac{1}{2}\sum_{i:y^{(i)}=0} -2\Sigma^{-1}(x^{(i)}-\mu_0) = \sum_{i:y^{(i)}=0} \Sigma^{-1}(x^{(i)}-\mu_0) := 0$$

$$\Rightarrow \Sigma^{-1}\sum_{i:y^{(i)}=0}(x^{(i)}-\mu_0) = 0$$

Since we assume Σ^{-1} is invertible, we have

$$\sum_{i:y^{(i)}=0}(x^{(i)}-\mu_0) = 0$$

$$\Rightarrow \mu_0 \sum_{i:y^{(i)}=0} 1 = \sum_{i:y^{(i)}=0} x^{(i)} \quad \text{i.e. } \mu_0 \sum_{i=1}^{N} \mathbb{I}\{y^{(i)}=0\} = \sum_{i=1}^{N} x^{(i)}\mathbb{I}\{y^{(i)}=0\}$$

$$\Rightarrow \mu_{0_{ML}} = \frac{\sum_{i=1}^{N} \mathbb{I}\{y^{(i)}=0\}\, x^{(i)}}{\sum_{i=1}^{N} \mathbb{I}\{y^{(i)}=0\}}$$

To solve $\mu_{1_{ML}}$: similar to solving $\mu_{0_{ML}}$

Since μ_0 only contribute to $p(x^{(i)}|y^{(i)}=1)$ for each i,

$$\underset{\mu_1}{\operatorname{argmax}}\, l = \underset{\mu_1}{\operatorname{argmax}} \sum_{i:y^{(i)}=1} \log p(x^{(i)}|y^{(i)}=1; \mu_0, \mu_1, \Sigma)$$

$$= \underset{\mu_1}{\operatorname{argmax}} \sum_{i:y^{(i)}=1} \log \frac{1}{2\pi M^{\frac{M}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x^{(i)}-\mu_1)^T \Sigma^{-1}(x^{(i)}-\mu_1)\right)$$

$$= \underset{\mu_1}{\operatorname{argmax}} \sum_{i:y^{(i)}=1} -\frac{1}{2}(x^{(i)}-\mu_1)^T \Sigma^{-1}(x^{(i)}-\mu_1) \quad ①$$

We define ① as l_{μ_1}, and take

$$\frac{\partial l_{\mu_1}}{\partial \mu_1} = -\frac{1}{2} \sum_{i:y^{(i)}=1} -2\Sigma^{-1}(x^{(i)}-\mu_0) = \sum_{i:y^{(i)}=0} \Sigma^{-1}(x^{(i)}-\mu_0) := 0$$

$$\Rightarrow \Sigma^{-1} \sum_{i:y^{(i)}=1} (x^{(i)}-\mu_1) = 0$$

$$\Rightarrow \sum_{i:y^{(i)}=0} (x^{(i)}-\mu_0) = 0 \quad \text{Since we assume } \Sigma^{-1} \text{ is invertible,}$$

$$\Rightarrow \mu_1 \sum_{i:y^{(i)}=1} 1 = \sum_{i:y^{(i)}=1} x^{(i)} \quad \text{i.e. } \mu_1 \sum_{i=1}^{N} \mathbb{I}\{y^{(i)}=1\} = \sum_{i=1}^{N} x^{(i)} \mathbb{I}\{y^{(i)}=1\}$$

$$\Rightarrow \mu_{1_{ML}} = \frac{\sum_{i=1}^{N} \mathbb{I}\{y^{(i)}=1\} x^{(i)}}{\sum_{i=1}^{N} \mathbb{I}\{y^{(i)}=1\}}$$

(c) MLE of Σ when $M = 1$

[3 points] When M (the dimension of $\mathbf{x}^{(i)}$) is 1, then $\Sigma = [\sigma^2]$ becomes a scalar, and its determinant is $|\Sigma| = \sigma^2$. Moreover, all data points become scalars $x^{(i)} \in \mathbb{R}$. By maximizing ℓ in (b) with respect to Σ, prove the maximum likelihood estimate for Σ would become

$$\Sigma_{\text{ML}} = \frac{1}{N}\sum_{i=1}^{N}(x^{(i)} - \mu_{y^{(i)}})^2$$

(c) To solve for Σ_{ML} when $M=1$, i.e. $\sigma^2_{ML} \in \mathbb{R}$:

when $M=1$, $p(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(x^{(i)} - \mu_{y^{(i)}})^2\right)$

$$\begin{aligned}
\underset{\sigma^2}{\text{argmax}}\, l &= \underset{\sigma^2}{\text{argmax}} \sum_{i=1}^{N} \log p(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma) \\
&\quad \text{since } \sum_{i=1}^{N} \log p(y^{(i)}; \varphi) \text{ does not has relation with } \sigma^2 \\
&= \underset{\sigma^2}{\text{argmax}} \sum_{i=1}^{N}\left[\left(\log\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{1}{2\sigma^2}(x^{(i)} - \mu_{y^{(i)}})^2\right] \\
&= \underset{\sigma^2}{\text{argmax}} \sum_{i=1}^{N}\left[-\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(x^{(i)} - \mu_{y^{(i)}})^2\right] \\
&\quad \log(2\pi\sigma^2) = \log 2\pi + \log\sigma^2 \\
&= \underset{\sigma^2}{\text{argmax}}\left[-\frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x^{(i)} - \mu_{y^{(i)}})^2\right] \quad (*)
\end{aligned}$$

write $(*)$ as l_{σ^2},

set $\frac{\partial l_{\sigma^2}}{\partial(\sigma^2)} = -\frac{N}{2}\frac{1}{\sigma^2} + \frac{1}{2(\sigma^2)^2}\sum_{i=1}^{N}(x^{(i)} - \mu_{y^{(i)}})^2 := 0$

$\times 2(\sigma^2)^2$ on both sides $\Rightarrow$ $-N\sigma^2 + \sum_{i=1}^{N}(x^{(i)} - \mu_{y^{(i)}})^2 = 0$

$$\Rightarrow \sigma^2_{ML} = \frac{1}{N}\sum_{i=1}^{N}(x^{(i)} - \mu_{y^{(i)}})^2$$

(d) MLE of Σ when $M > 1$

[4 points] Repeat the above for a general $M > 1$, and show that on maximizing ℓ in (b) with respect to Σ, the maximum likelihood estimate for Σ becomes

$$\Sigma_{\text{ML}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)} - \mu_{y^{(i)}})(\mathbf{x}^{(i)} - \mu_{y^{(i)}})^\top.$$

Hint: For a matrix X, you may use the following gradient expression without proof:

$$\nabla_X \log|X| = (X^{-1})^\top = X^{-1} \quad \textit{if } X \textit{ is symmetric}$$

$$\nabla_X \left(a^\top X^{-1} a\right) = -X^{-1}aa^\top X^{-1}$$

(c) To solve for Σ_{ML} when $M>1$,

$$\underset{\Sigma}{\text{argmax}}\, l = \underset{\sigma^2}{\text{argmax}} \sum_{i=1}^{N} \log p(x^{(i)}/y^{(i)}; \mu_0, \mu_1, \Sigma)$$

since $\sum_{i=1}^{N} \log p(y^{(i)}; \varphi)$ does not has relation with Σ

$$= \underset{\Sigma}{\text{argmax}} \sum_{i=1}^{N} \left[\left(\log \frac{1}{(2\pi)^{\frac{M}{2}} |\Sigma|^{\frac{1}{2}}} - \frac{1}{2}(x^{(i)} - \mu_{y^{(i)}})^T \Sigma^{-1} (x^{(i)} - \mu_{y^{(i)}}) \right) \right]$$

$$= \underset{\Sigma}{\text{argmax}} \sum_{i=1}^{N} \left(-\log(2\pi)^{\frac{M}{2}} - \frac{1}{2}\log|\Sigma| \right) - \frac{1}{2}\sum_{i=1}^{N} (x^{(i)} - \mu_{y^{(i)}})^T \Sigma^{-1} (x^{(i)} - \mu_{y^{(i)}})$$

const; $(x^{(i)} - \mu_{y^{(i)}})^T$ as a_i^T; $(x^{(i)} - \mu_{y^{(i)}})$ as a_i

writing $(x^{(i)} - \mu_{y^{(i)}})$ as a_i

$$= \underset{\Sigma}{\text{argmax}} -\frac{N}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{N} a_i^T \Sigma^{-1} a_i$$

①

write ① as l_Σ

$$\Rightarrow \nabla_\Sigma l_\Sigma = -\frac{N}{2}(\Sigma^{-1})^T - \frac{1}{2}\sum_{i=1}^{N}(-\Sigma^{-1}a_i a_i^T \Sigma^{-1})$$

$$= -\frac{N}{2}\Sigma^{-1} + \frac{1}{2}\sum_{i=1}^{N}\Sigma^{-1}a_i a_i^T \Sigma^{-1}$$

Set $\nabla_\Sigma l_\Sigma := 0 \Rightarrow N\Sigma^{-1} - \sum_{i=1}^{N}\Sigma^{-1}a_i a_i^T \Sigma^{-1} = 0$

Left, right multiply Σ on both sides

$$N\Sigma - \sum_{i=1}^{N}\Sigma\Sigma^{-1}a_i a_i^T \Sigma^{-1}\Sigma = 0$$

$$N\Sigma = \sum_{i=1}^{N} a_i a_i^T$$

$$\Rightarrow \Sigma_{ML} = \frac{1}{N}\sum_{i=1}^{N} a_i a_i^T$$

$$= \frac{1}{N}\sum_{i=1}^{N}(x^{(i)} - \mu_{y^{(i)}})^T (x^{(i)} - \mu_{y^{(i)}})$$

4 Naive Bayes for Classifying SPAM

(a) Naive Bayes with Bayesian Smoothing

[8 points] Naive Bayes with Bayesian Smoothing

Recall that Naive Bayes can be solved with MLE, in which we count the occurences of each feature (or word). Adding Laplace smoothing, we get:

$$P(C_i) = \phi_i = \frac{N^{C_i}}{\sum_{i'} N^{C_{i'}}} \tag{1}$$

$$P(x_j|C_i) = \mu_j^i = \frac{N_j^{C_i} + \alpha}{\sum_{j'} N_{j'}^{C_i} + \alpha M} \tag{2}$$

where M is the dimension of $\mathbf{x}$ (total number of features or words). $N_j^{C_i}$ is the count of the occurrences of x_j with class C_i. $\alpha > 0$ is the Laplace smoothing hyperparameter. We also denote K as the number of classes.

Show that Laplace smoothing is equivalent to solving the MAP estimate of Naive Bayes, where we have a prior on the values of $\boldsymbol{\mu}$ which follow a symmetric Dirichlet distribution:

$$P(\boldsymbol{\mu}) = \frac{1}{Z} \prod_{i=1}^{K} \prod_{j=1}^{M} (\mu_j^i)^{\alpha} \tag{3}$$

where Z is some normalizing constant.

[Hint: You may use the Naive Bayes likelihood and MLE derivations from lecture without proof.]

(a) To find μ_{MAP}, by def is to find μ maximizing the following problem

$$\max_{\mu = \{\mu_1^i, \ldots, \mu_M^i\}_{i=1}^K} p(\mu|x,y)$$

$$\text{s.t} \sum_{j=1}^{M} \mu_j^i = 1, \quad \forall 1 \le i \le K$$

Note:

$$\operatorname*{argmax}_{\mu} p(\mu|x,y) = \operatorname*{argmax}_{\mu} p(x,y|\mu)p(\mu)$$

$$= \operatorname*{argmax}_{\mu} \log p(x,y|\mu) + \log p(\mu)$$

By our deduction in lecture

$$p(x,y|\mu) = \prod_{i=1}^{K} \prod_{j=1}^{M} (\mu_i^j)^{N_{C_i}^j}$$

(we generalize it from two classes to multiple classes)

So $$\log p(x,y|\mu) = \sum_{i=1}^{K} \sum_{j=1}^{M} N_{C_i}^j \log \mu_i^j$$

And by our assumption of the distribution of μ,

$$\log p(\mu) = \sum_{i=1}^{K} \sum_{j=1}^{M} \alpha \log \mu_i^j - \log Z$$

const

Thus it is equivalent to find μ that optimizes:

$$\max_{\mu} \sum_{i=1}^{K} \sum_{j=1}^{M} (N_{ci}^j + \alpha) \log \mu_i^j$$

$$\text{s.t.} \sum_{j=1}^{M} \mu_i^j = 1 \quad \forall i$$

Note: by checking Hessian $\frac{\partial^2 f}{\partial \mu_i^j \partial \mu_i^k} = \begin{cases} \frac{-(N_{ci}^j + \alpha)}{(\mu_i^j)^2}, & j = k \\ 0, & j \neq k \end{cases}$

$H_f(\mu_i) = \begin{bmatrix} \cdots & & \\ & \cdots & \\ & & \cdots \end{bmatrix}$ is negative definite

$\Rightarrow$ the objective is diffble and concave

$\Rightarrow$ to maximize it $\Leftrightarrow$ to minimize its negative that is convex and diffble

$\Rightarrow$ by Lagrange's multiplier method we can get the global maximum.

Using the Lagrange multiplier method, we construct the Lagrang function:

$$\mathcal{L}(\mu, \lambda) = \sum_{i=1}^{K} \sum_{j=1}^{M} (N_{ci} + \alpha) \log \mu_i^j + \sum_{i=1}^{K} \lambda_i \left(1 - \sum_{j=1}^{M} \mu_i^j\right)$$

Then $\frac{\partial \mathcal{L}}{\partial \mu_i^j} = \frac{N_{ci}^j + \alpha}{\mu_i^j} - \lambda_i := 0 \Rightarrow \mu_i^j = \frac{N_{ci}^j + \alpha}{\lambda_i}$

Then we solve λ_i for each i:

Since $\sum_{j=1}^{M} \mu_i^j = 1 \Rightarrow \sum_{j=1}^{M} \frac{N_{ci}^j + \alpha}{\lambda_i} = 1 \Rightarrow \lambda_i = \sum_{j=1}^{M} (N_{ci}^j + \alpha)$

Then we conclude: $\mu_i^j = \frac{N_{ci}^j + \alpha}{\sum_{j=1}^{M} (N_{ci}^j + \alpha)} = \frac{N_{ci}^j + \alpha}{\sum_{j=1}^{M} N_{ci}^j + \alpha M}$

which is exactly the same as what we obtained by Laplace smoothing.

(b) SPAM classifier

(b) In this subproblem, we will use the naive Bayes algorithm to build a SPAM classifier which we covered in our class. Our classifier will distinguish between "real" (non-spam) emails and spam emails. For this problem, we obtained a set of spam emails and a set of non-spam newsgroup messages. Using only the subject line and body of each message, the classifier will learn to distinguish between the two groups. In this problem, we're going to call the features *tokens*.

i. **[9 points]** (Autograder) Implement a naive Bayes classifier for spam classification, using the multinomial event model and Laplace smoothing. `naive_bayes_spam.ipynb` will work you through implementing this classifier. In this task, we will train your parameters with the training dataset **MATRIX.TRAIN**. Then, use these parameters to classify the test dataset **MATRIX.TEST** and compute the accuracy using the **evaluate** function. Implement your code in `naive_bayes_spam.py` and submit your `naive_bayes_spam.py` implementation with `naive_bayes_spam.ipynb`. Please include all the outputs in your `naive_bayes_spam.ipynb` file.

- For each email (\mathbf{x}), we compute:

$$\log P(\text{spam} \mid \mathbf{x}) \approx \log \phi + \sum_j \big(\mathbf{x}[j] \cdot \log(\mu_{\text{spam}}[j])\big) \tag{1}$$

$$\log P(\text{nonspam} \mid \mathbf{x}) \approx \log(1-\phi) + \sum_j \big(\mathbf{x}[j] \cdot \log(\mu_{\text{nonspam}}[j])\big) \tag{2}$$

- We label the email as spam (`1`) if $logP(\text{spam} \mid \mathbf{x}) > \log P(\text{nonspam} \mid \mathbf{x})$, otherwise non-spam (`0`).

ii. **[2 points]** Some tokens may be particularly indicative of an email being spam or non-spam. One way to measure how indicative a token i is for the SPAM class by looking at:

$$\log\left(\frac{p(x_j = i|y=1)}{p(x_j = i|y=0)}\right) = \log\left(\frac{P(\text{token}_i|\text{email is SPAM})}{P(\text{token}_i|\text{email is NOTSPAM})}\right)$$

Using the parameters you obtained in part (a), find the 5 tokens that are most indicative of the SPAM class (i.e., 5 tokens that have the highest positive value on the measure above). What are the first five tokens from your implementation. Please provide them in your **writeup**.

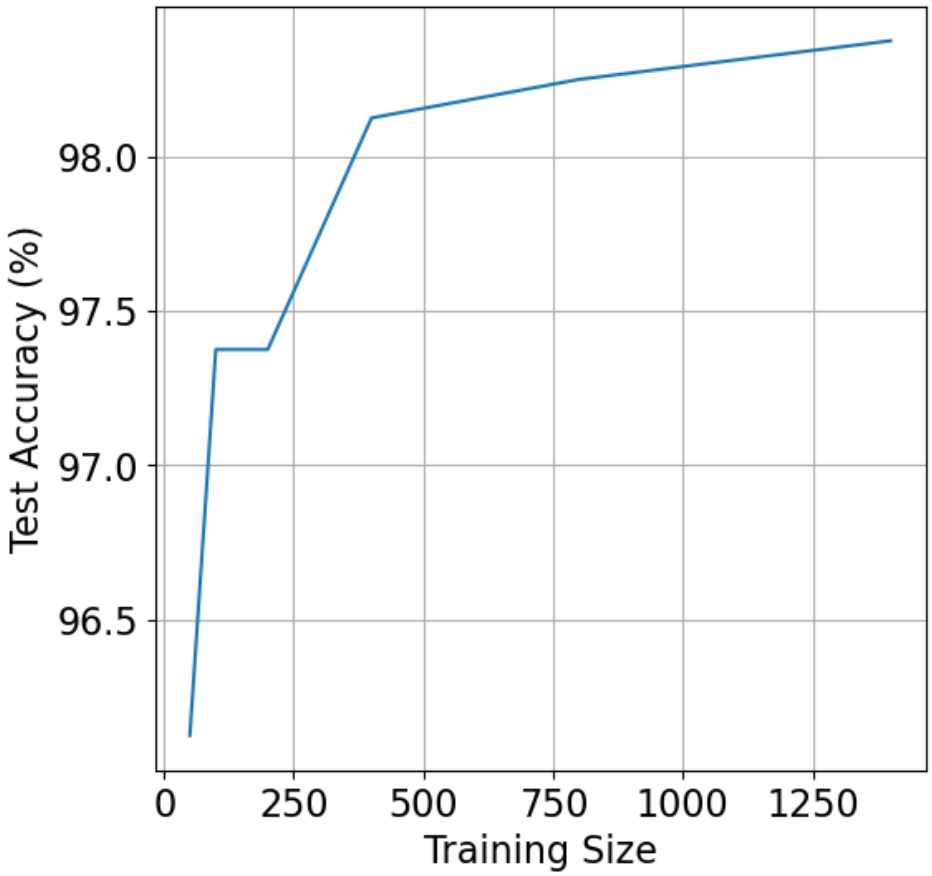
```
Top 5 most indicative tokens are: ['httpaddr' 'spam' 'unsubscrib' 'ebai' 'valet'].
```

iii. **[2 points]** Repeat part (a), but with different naive Bayes classifiers each trained with different training sets **MATRIX.TRAIN.***. Evaluate the classifiers with **MATRIX.TEST** and report the classification accuracy for each classifier in your **writeup**.

report is as below:

```
(50, 1448)
Accuracy for 50 mail data (data/q4_data/MATRIX.TRAIN.50): 96.1250%
(100, 1448)
Accuracy for 100 mail data (data/q4_data/MATRIX.TRAIN.100): 97.3750%
(200, 1448)
Accuracy for 200 mail data (data/q4_data/MATRIX.TRAIN.200): 97.3750%
(400, 1448)
Accuracy for 400 mail data (data/q4_data/MATRIX.TRAIN.400): 98.1250%
(800, 1448)
Accuracy for 800 mail data (data/q4_data/MATRIX.TRAIN.800): 98.2500%
(1400, 1448)
Accuracy for 1400 mail data (data/q4_data/MATRIX.TRAIN.1400): 98.3750%
```

iv. [**2 points**] Please provide the final training data size-accuracy plot from part (c) in your **writeup**.



v. [**2 points**] Which training set size gives you the best classification accuracy? Please provide your own analysis in your **writeup**.

Answer: 1400 mail data training set provides the best classification accuracy of 98.375%. By looking at our optimal solution of $\mu_i^j = \frac{N_{C_i}^j + \alpha}{\sum_{j'=1}^{M}(N_{C_i}^{j'} + \alpha)}$ and $\phi_i = \frac{N_{C_i}}{\sum_{i'} N_{C_{i'}}}$, we can see that the classification is a direct reflection of how frequent a word in one email occurs in spam/nonspam emails. So more data gives more generality.