DS-GA 1003 - Homework 4

Eric Niblock

March 20, 2021

Consider a binary classification setting with input space $\mathcal{X} = \mathbb{R}^d$, outcome space $\mathcal{Y}_\pm = \{-1, 1\}$, and a dataset $\mathcal{D} = \left((x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)})\right)$.

In the lecture we derived logistic regression using the Bernoulli response distribution. In this problem you will show that it is equivalent to ERM with logistic loss.

Consider a linear scoring function in the space $\mathcal{F}_{\textbf{score}} = \left\{x \mapsto x^T w \mid w \in \mathbb{R}^d\right\}$. A simple way to make predictions (similar to what we've seen with the perceptron algorithm) is to predict $\hat{y} = 1$ if $x^T w > 0$, or $\hat{y} = \textbf{sign}(x^T w)$. Accordingly, we consider margin-based loss functions that relate the loss with the margin, $yx^T w$. A positive margin means that $x^T w$ has the same sign as y, i.e. a correct prediction. Specifically, let's consider the logistic loss function $\ell_{\textbf{logistic}}(y, w) = \log\left(1 + \exp(-yw^T x)\right)$. This is a margin-based loss function that you have now encountered several times. Given the logistic loss, we can now minimize the empirical risk on our dataset $\mathcal{D}$ to obtain an estimate of the parameters, $\hat{w}$.

As discussed in the lecture, given that $p(y = 1 \mid x; w) = 1/(1 + \exp(-x^T w))$, we can estimate w by maximizing the likelihood, or equivalently, minimizing the negative log-likelihood ($\text{NLL}_\mathcal{D}(\text{w})$ in short) of the data.

1. **Show that the two approaches are equivalent, i.e. they will produce the same solution for w.**

We have that,

$$L(\mathcal{D}; w) = \prod_{i=1}^{n} \left(\frac{1}{1 + \exp(-y_i x_i^T w)}\right) \tag{1}$$

First we show it must be the case that,

$$\hat{w} = \underset{w \in \mathbb{R}^d}{argmax}\ L(\mathcal{D}; w) = \underset{w \in \mathbb{R}^d}{argmax}\ log(L(\mathcal{D}; w)) \tag{2}$$

We know that $x \to log(x)$ is a strictly increasing function. Let it be the case that,

$$a^* = \underset{w \in \mathbb{R}^d}{argmax}\ L(\mathcal{D}; w) \qquad b^* = \underset{w \in \mathbb{R}^d}{argmax}\ log(L(\mathcal{D}; w)) \tag{3}$$

Now, assume that $a^* \neq b^*$. If this is the case, then $L(\mathcal{D}; a^*) > L(\mathcal{D}; b^*)$, which follows from the definition of a^*. Furthermore, since $x \to log(x)$ is a strictly increasing function, it follows that $log(L(\mathcal{D}; a^*)) > log(L(\mathcal{D}; b^*))$. However, given the definition of b^*, we have that $log(L(\mathcal{D}; a^*)) < log(L(\mathcal{D}; b^*))$. Contradiction. Therefore it must be the case that $a^* = b^*$.

Now, we have that,

$$\begin{aligned} log(L(\mathcal{D}; w)) &= \sum_{i=1}^{n} log\left(\frac{1}{1 + \exp(-y_i x_i^T w)}\right) \\ &= -\sum_{i=1}^{n} log\left(1 + \exp(-y_i x_i^T w)\right) \end{aligned} \tag{4}$$

Maximizing a function $f(x)$ is the same as minimizing the function $-f(x)$, provided that $f(x)$ is concave (shown in Problem 4). Negating the above function yields,

$$-log(L(\mathcal{D}; w)) = \sum_{i=1}^{n} log\left(1 + \exp(-y_i x_i^T w)\right) \tag{5}$$

Which you will notice as simply the sum of $\ell_{\text{logistic}}(y, w)$. Thus we have shown that,

$$\hat{w} = \underset{w \in \mathbb{R}^d}{argmax}\ L(\mathcal{D}; w) = \underset{w \in \mathbb{R}^d}{argmax}\ log(L(\mathcal{D}; w)) = \underset{w \in \mathbb{R}^d}{argmin}\ -log(L(\mathcal{D}; w)) \tag{6}$$

In this problem, we will investigate the behavior of MLE for logistic regression when the data is linearly separable.

2. **Show that the decision boundary of logistic regression is given by $\{x\colon x^T w = 0\}$. Note that the set will not change if we multiply the weights by some constant c.**

We know that the decision boundary separates the classes given in the output space as $\{-1, 1\}$. Therefore, a point lying exactly on the decision boundary should yield $p(y = 1 \mid x; w) = p(y = -1 \mid x; w) = \frac{1}{2}$. Notice that for any $\{x\colon x^T w = 0\}$ we have,

$$p(y = 1 \mid x^T w = 0) = \frac{1}{1 + e^0} = \frac{1}{2} \tag{7}$$

$$p(y = -1 \mid x^T w = 0) = 1 - \frac{1}{1 + e^0} = \frac{1}{2} \tag{8}$$

Which is as expected. Furthermore, notice that multiplying by a constant c does not affect the decision boundary, since the exponent of e above would still be 0. Thus $\{x\colon x^T w = 0\}$ represents the decision boundary.

As further justification, take some $\epsilon > 0$ and consider the following two cases. We have that $p(y = 1 \mid x^T w = \epsilon) \in (0.5, 1)$, and hence would be classified as the $+1$. Furthermore $p(y = 1 \mid x^T w = -\epsilon) \in (0, 0.5)$, and hence would be classified as -1. Thus $x^T w = 0$ must represent the decision boundary, because small oscillations in either direction from $x^T w = 0$ lead to different classifications.

3. **Suppose the data is linearly separable and by gradient descent/ascent we have reached a decision boundary defined by $\hat{w}$ where all examples are classified correctly. Show that we can always increase the likelihood of the data by multiplying a scalar c on $\hat{w}$, which means that MLE is not well-defined in this case. (Hint: You can show this by taking the derivative of $L(c\hat{w})$ with respect to c, where L is the likelihood function.)**

We have the likelihood function given below,

$$L(\mathcal{D}; c\hat{w}) = \prod_{i=1}^{n} \left(\frac{1}{1 + \exp(-y_i x_i^T c\hat{w})} \right) \tag{9}$$

Instead we consider maximizing the log-likelihood,

$$log(L(\mathcal{D}; c\hat{w})) = \sum_{i=1}^{n} log\left(\frac{1}{1+\exp(-y_i x_i^T c\hat{w})}\right)$$
$$= -\sum_{i=1}^{n} log\left(1+\exp(-y_i x_i^T c\hat{w})\right) \tag{10}$$

And,

$$\frac{\partial log(L(\mathcal{D}; c\hat{w}))}{\partial c} = \sum_{i=1}^{n} \frac{y_i x_i^T \hat{w} \exp(-y_i x_i^T c\hat{w})}{1+\exp(-y_i x_i^T c\hat{w})} \tag{11}$$

Since we have assumed that all of the data is classified correctly, we have that $y_i x_i^T \hat{w} > 0$ for all i. Furthermore, exponential functions are always positive. Therefore the above partial derivative with respect to c is always positive, and thus multiplying $\hat{w}$ by a constant $c > 1$ increases the log-likelihood and therefore the likelihood.

As we've shown in above, when the data is linearly separable, MLE for logistic regression may end up with weights with very large magnitudes. Such a function is prone to overfitting. In this part, we will apply regularization to fix the problem. The ℓ_2 regularized logistic regression objective function can be defined as

$$\begin{aligned} J_{\textbf{logistic}}(w) &= \hat{R}_n(w) + \lambda\|w\|^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} \log\left(1+\exp\left(-y^{(i)}w^T x^{(i)}\right)\right) + \lambda\|w\|^2. \end{aligned}$$

4. **Prove that the objective function $J_{\textbf{logistic}}(w)$ is convex. You may use any facts mentioned in the convex optimization notes.**

First we rewrite the objective function as follows,

$$J_{\text{logistic}}(w) = \hat{R}_n(w) + \lambda\|w\|^2 = \sum_{i=1}^{n} f_i(d) + \lambda\|w\|^2 \tag{12}$$

Where $d = -y^{(i)}w^T x^{(i)}$ (so $d \in \mathbb{R}$) and,

$$f_i(d) = \log\left(1 + \exp\left(d\right)\right) \tag{13}$$

Now, $f : \mathbb{R} \to \mathbb{R}$ is convex if f has a second derivative greater than zero on all of $\mathbb{R}$. So,

$$\frac{df_i(d)}{dd} = \frac{e^d}{1+e^d} \tag{14}$$

$$\frac{d^2 f_i(d)}{dd^2} = \frac{e^d}{1+e^d} - \left(\frac{e^d}{1+e^d}\right)^2 \tag{15}$$

It is trivial that the second derivative is always positive, and thus $f_i(d)$ is convex. Furthermore, the sum of convex functions is convex, and any norm on $\mathbb{R}^n$ is convex. Thus,

$$J_{\text{logistic}}(w) = \hat{R}_n(w) + \lambda\|w\|^2 = \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \exp\left(-y^{(i)} w^T x^{(i)}\right)\right) + \lambda\|w\|^2 \tag{16}$$

Is convex provided $\lambda \geq 0$.

5. **Complete the `f_objective` function in the skeleton code, which computes the objective function for $J_{\text{logistic}}(w)$. (Hint: you may get numerical overflow when computing the exponential literally, e.g. try e^{1000} in Numpy. Make sure to read about the log-sum-exp trick and use the numpy function *logaddexp* to get accurate calculations and to prevent overflow.**

The following function computes the objective function $J_{\text{logistic}}(w)$,

```
def f_objective(theta, X, y, l2_param=1):
    '''
    Args:
        theta: 1D numpy array of size num_features
        X: 2D numpy array of size (num_instances, num_features)
        y: 1D numpy array of size num_instances
        l2_param: regularization parameter

    Returns:
        objective: scalar value of objective function
    '''
    use = -1*np.multiply(X@theta,y)
    obj = np.mean(np.logaddexp(0,use)) + l2_param*(np.linalg.norm(theta)**2)
    return(obj)
```

6. **Complete the `fit_logistic_regression_function` in the skeleton code using the `minimize` function from `scipy.optimize`. Use this function to train a model on the provided data. Make sure to take the appropriate preprocessing steps, such as standardizing the data and adding a column for the bias term.**

First, we imported the data and standardized it using the following function. Every feature was centered by subtracting the feature's mean, and scaled by dividing by each feature's respective standard deviation. Furthermore we added a feature of all ones in order to add a coefficient.

```
def standardize(X1, X2):

    for i in range(X1.shape[1]):
        X1[:,i] = X1[:,i] - np.mean(X1[:,i])
        X1[:,i] = X1[:,i]/ np.std(X1[:,i])
        X2[:,i] = X2[:,i] - np.mean(X1[:,i])
        X2[:,i] = X2[:,i]/ np.std(X1[:,i])

    b = np.ones((X1.shape[0],1))
    X1 = np.hstack((X1,b))
    b = np.ones((X2.shape[0],1))
    X2 = np.hstack((X2,b))
    return(X1,X2)
```

The following is our completed function which minimizes our objective function,

```
def fit_logistic_reg(X, y, objective_function, l2_param=1):
    '''
    Args:
        X: 2D numpy array of size (num_instances, num_features)
        y: 1D numpy array of size num_instances
        objective_function: function returning the value of the objective
        l2_param: regularization parameter

    Returns:
        optimal_theta: 1D numpy array of size num_features
    '''
```

```
        theta = np.zeros(X.shape[1])
        mini = sco.minimize(objective_function, theta, args = (X,y,l2_param))
        return(mini['x'])
```

We then trained the following model,

```
X_train = np.loadtxt('X_train_hw4.txt',delimiter=',')
y_train = np.loadtxt('y_train_hw4.txt',delimiter=',')
X_val = np.loadtxt('X_val_hw4.txt',delimiter=',')
y_val = np.loadtxt('y_val_hw4.txt',delimiter=',')
y_train[y_train==0] = -1
y_val[y_val==0] = -1

X_train, X_val = standardize(X_train, X_val)

sol = fit_logistic_reg(X_train, y_train, f_objective, l2_param=0.035)
```

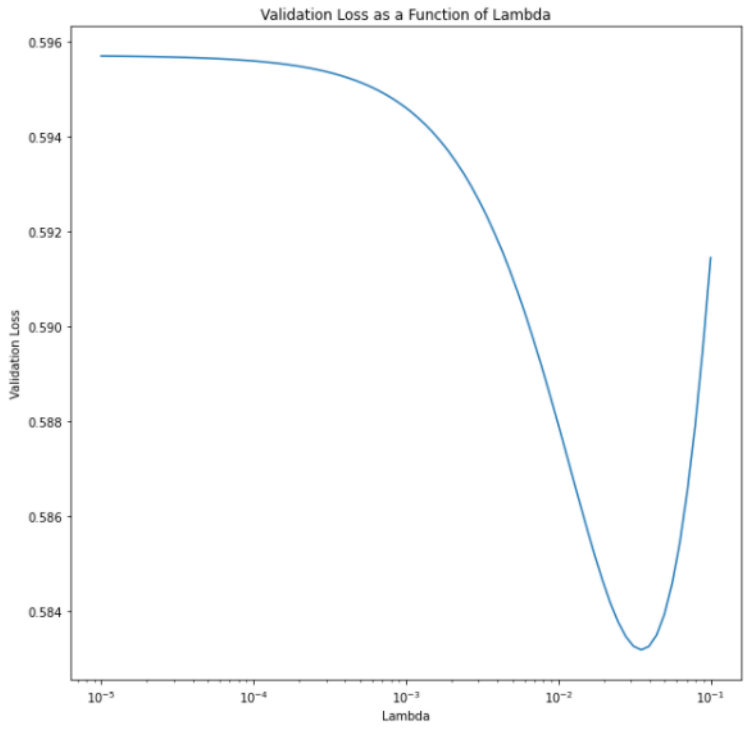
We determined this configuration was optimal in the following question.

7. Find the ℓ_2 regularization parameter that maximizes the log-likelihood on the validation set. Plot the log-likelihood for different values of the regularization parameter.

The following code was used to find the hyper-parameter which maximized the log-likelihood (or, equivalently, minimized the negative log-likelihood, which is shown here),

```
val_loss = []
for l2 in np.logspace(-5,-1,80):
    sol = fit_logistic_reg(X_train, y_train, f_objective, l2_param=l2)
    val_loss.append(f_objective_noreg(sol, X_val, y_val)) #No Reg Term
```

We then produced the following plot, with the minimum located at $\lambda = 0.035$,



8. **[Optional] It seems reasonable to interpret the prediction $f(x) = \phi(w^T x) = 1/(1 + e^{-w^T x})$ as the probability that $y = 1$, for a randomly drawn pair (x, y). Since we only have a finite sample (and we are regularizing, which will bias things a bit) there is a question of how well "calibrated" our predicted probabilities are. Roughly speaking, we say $f(x)$ is well calibrated if we look at all examples (x, y) for which $f(x) \approx 0.7$ and we find that close to 70% of those examples have $y = 1$, as predicted... and then we repeat that for all predicted probabilities in $(0, 1)$. To see how well-calibrated our predicted probabilities are, break the predictions on the validation set into groups based on the predicted probability (you can play with the size of the groups to get a result you think is informative). For each group, examine the percentage of positive labels. You can make a table or graph. Summarize the results. You may get some ideas and references from scikit-learn's discussion.**

The following code was used to divide the predicted examples based on probability, and then determine the percentage classified correctly by grouping,

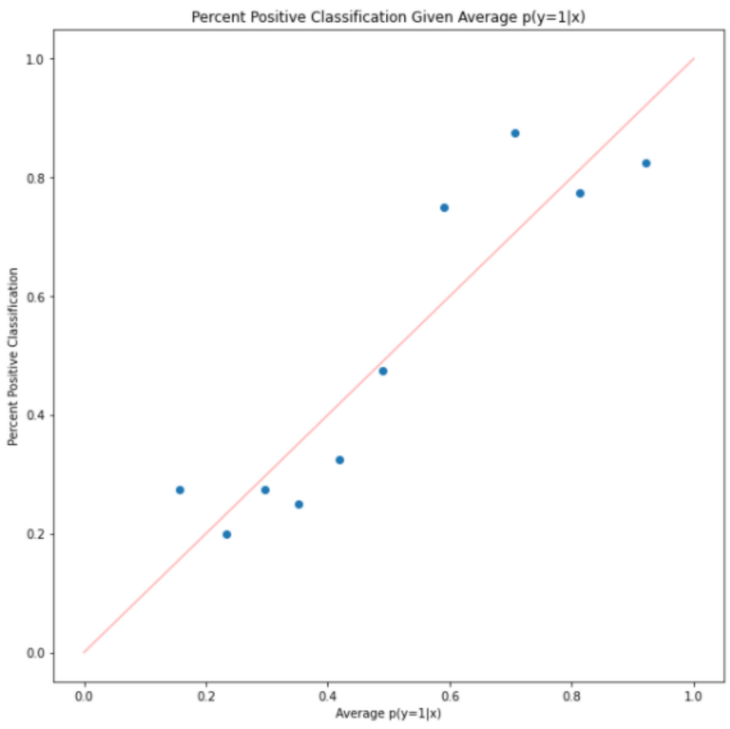
```
def calibration(X, sol, y,group_size):
    preds = 1/(1+np.exp(-1*X@sol))
    preds_sorted, y_sorted = zip(*sorted(zip(preds, y)))
    grouped = np.array_split(preds_sorted, group_size)
    group_y = np.array_split(y_sorted, group_size)

    all_acc = []
    for e in range(len(grouped)):
        acc = 0
        group = grouped[e]
        gy = group_y[e]
        for r in range(len(group)):

            if gy[r] == 1:
                acc+=1
        all_acc.append(acc/len(group))

    return([np.mean(g) for g in grouped],all_acc)
```

The results then produced the following plot for $\lambda = 0.035$,



We note that the model appears *roughly* calibrated. Though the results do appear to follow a roughly linear trend, further experimentation with different values of λ reveal that the trend is actually more sigmoidal.

Let's continue with logistic regression in the Bayesian setting, where we introduce a prior $p(w)$ on $w \in \mathbb{R}^d$.

9. **For the same dataset $\mathcal{D}$ described at the beginning of the Section, give an expression for the posterior density $p(w \mid \mathcal{D})$ in terms of the negative log-likelihood function $\mathrm{NLL}_{\mathcal{D}}(\mathrm{w})$ and the prior density $p(w)$ (up to a proportionality constant is fine).**

By Bayes Theorem, we have that,

$$p(w \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid w)p(w)}{p(\mathcal{D})} = \frac{\exp(-NLL_{\mathcal{D}}(w))p(w)}{p(\mathcal{D})} \tag{17}$$

Where $p(\mathcal{D})$ represents a proportionality or normalization constant.

10. **Suppose we take a prior on w of the form $w \sim \mathcal{N}(0, \Sigma)$, that is in the Gaussian family. Is this a conjugate prior to the likelihood given by logistic regression?**

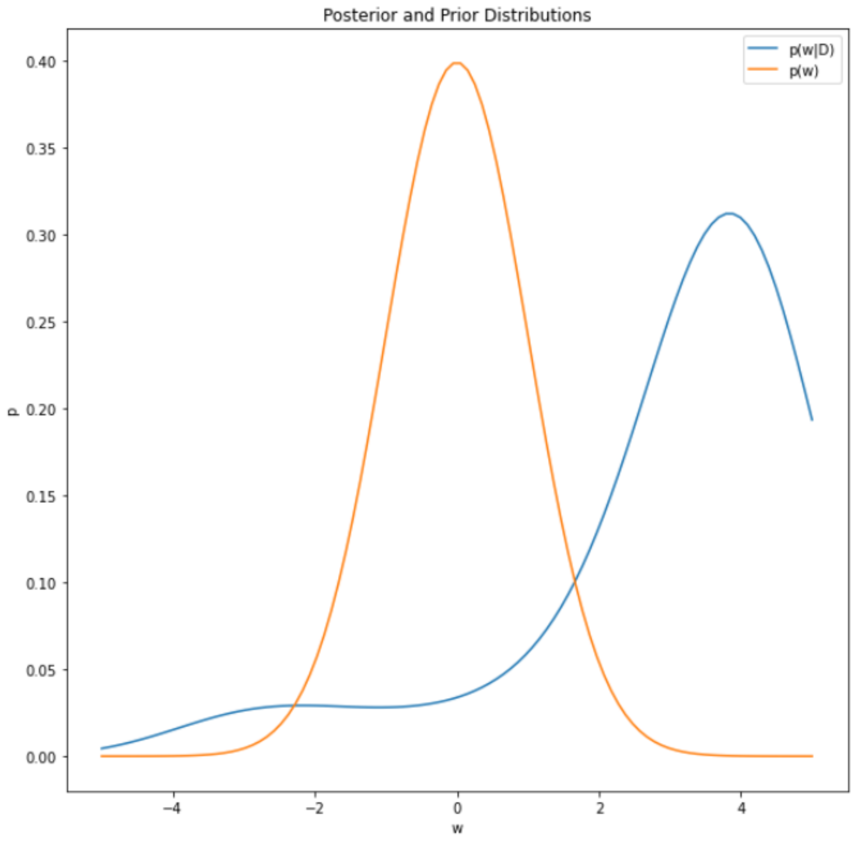
A closed form solution was attempted for this problem, though a counter-example was instead provided. The following code was used to calculate the necessary Gaussian distribution,

```
def gauss(w,sigma,u):
    d = len(w)
    det = 1/np.linalg.det(sigma)
    f = (1/(2*np.pi))**(d/2)
    m = w-u
    e = np.exp(-1*m.T@np.linalg.inv(sigma)@m/2)
    return(f*det*e)
```

We then used the following code to calculate the prior and posterior using random data with $\mu = 0$, $\Sigma = I$,

```
X = np.random.rand(20,1)
y = np.random.randint(2, size=20)
y[y==0] = -1
post = []
p = []

for w in np.linspace(-5,5,100):
    g = gauss(np.array([w]),np.array([[1]]),np.array([0]))
    p.append(g)
    post.append(g*np.exp(len(X)*f_objective_noreg(np.array([w]), X, y)))
```



As you can see the posterior is obviously not Gaussian. This confirms that we could not write the following expression,

$$p(w|\mathcal{D}) \propto e^{\frac{1}{2}(w-\mu)^T\Sigma^{-1}(w-\mu)} \tag{18}$$

And shows that the Gaussian family is not a conjugate prior to the likelihood given by logistic regression.

11. **Show that there exist a covariance matrix Σ such that MAP (maximum a posteriori) estimate for w after observing data $\mathcal{D}$ is the same as the minimizer of the regularized logistic regression function defined in Regularized Logistic Regression paragraph above, and give its value. [Hint: Consider minimizing the negative log posterior of w. Also, remember you can drop any terms from the objective function that don't depend on w. You may freely use results of previous problems.]**

We know that,

$$\hat{w} = \underset{w \in \mathbb{R}^d}{argmin}\ \hat{R}_n(w) + \lambda \|w\|^2 \tag{19}$$

Then we should also have that,

$$\begin{aligned} w^* &= \underset{w \in \mathbb{R}^d}{argmax}\ p(w \mid \mathcal{D}) \\ &= \underset{w \in \mathbb{R}^d}{argmax}\ \exp(-NLL_{\mathcal{D}}(w))p(w) \\ &= \underset{w \in \mathbb{R}^d}{argmin}\ -log[\exp(-NLL_{\mathcal{D}}(w))p(w)] \\ &= \underset{w \in \mathbb{R}^d}{argmin}\ NLL_{\mathcal{D}}(w) - log(p(w)) \end{aligned} \tag{20}$$

And since we know the prior is given by $w \sim \mathcal{N}(0, \Sigma)$, and that $NLL_{\mathcal{D}}(w) = n\hat{R}_n(w)$ we see that,

$$\begin{aligned} w^* &= \underset{w \in \mathbb{R}^d}{argmin}\ NLL_{\mathcal{D}}(w) + \frac{1}{2} w^T \Sigma^{-1} w \\ &= \underset{w \in \mathbb{R}^d}{argmin}\ \hat{R}_n(w) + \frac{1}{2n} w^T \Sigma^{-1} w \end{aligned} \tag{21}$$

So, equating the two regularization terms yields,

$$\frac{1}{2n} w^T \Sigma^{-1} w = \lambda w^T w \tag{22}$$

$$w^T \Sigma^{-1} w = w^T (2n\lambda) w \tag{23}$$

$$\Sigma = \frac{1}{2n\lambda} I \tag{24}$$

12. **In the Bayesian approach, the prior should reflect your beliefs about the parameters before seeing the data and, in particular, should be independent on the eventual size of your dataset. Imagine choosing a prior distribution $w \sim \mathcal{N}(0, I)$. For a dataset $\mathcal{D}$ of size n, how should you choose λ in our regularized logistic regression objective function so that the ERM is equal to the mode of the posterior distribution of w (i.e. is equal to the MAP estimator).**

In order for the previously derived expression to hold for the given $\Sigma = I$, we must have,

$$I = \frac{1}{2n\lambda} I \tag{25}$$

And therefore,

$$\frac{1}{2n\lambda} = 1 \tag{26}$$

$$\lambda = \frac{1}{2n} \tag{27}$$

Consider flipping a biased coin where $p(z = H \mid \theta_1) = \theta_1$. However, we cannot directly observe the result z. Instead, someone reports the result to us, which we denoted by x. Further, there is a chance that the result is reported incorrectly *if it's a head.* Specifically, we have $p(x = H \mid z = H, \theta_2) = \theta_2$ and $p(x = T \mid z = T) = 1$.

13. Show that $p(x = H \mid \theta_1, \theta_2) = \theta_1\theta_2$.

The only way in which we are reported a head is when an actual head is flipped (with probability $p(z = H \mid \theta_1) = \theta_1$) and when the report is transmitted factually (with probability $p(x = H \mid z = H, \theta_2) = \theta_2$). Thus,

$$p(x = H \mid \theta_1, \theta_2) = p(z = H \mid \theta_1)p(x = H \mid z = H, \theta_2) = \theta_1\theta_2 \tag{28}$$

14. Given a set of reported results $\mathcal{D}_r$ of size N_r, where the number of heads is n_h and the number of tails is n_t, what is the likelihood of $\mathcal{D}_r$ as a function of θ_1 and θ_2.

Since there are only two possible results from our sample space, we know that if $p(x = H \mid \theta_1, \theta_2) = \theta_1\theta_2$ then $p(x = T \mid \theta_1, \theta_2) = 1 - \theta_1\theta_2$. Therefore, the likelihood of $\mathcal{D}_r$ is given by,

$$L(\mathcal{D}_r; \theta_1, \theta_2) = (\theta_1\theta_2)^{n_h}(1 - \theta_1\theta_2)^{n_t} \tag{29}$$

15. Can we estimate θ_1 and θ_2 using MLE? Explain your judgment.

We cannot estimate θ_1 and θ_2 using MLE (in a way that provides useful information). Imagine instead we had a likelihood function with $\theta = \theta_1\theta_2$, then we simply have,

$$L(\mathcal{D}_r; \theta) = \theta^{n_h}(1-\theta)^{n_t} \tag{30}$$

Where $\theta_{MLE} = \frac{n_h}{n_h+n_t}$ (simply the MLE for Bernoulli experiments). Thus, we also have,

$$\theta_1\theta_2 = \frac{n_h}{n_h+n_t} \tag{31}$$

Such an equation generally has infinite solutions, and thus we find no reliable information about either θ_1 or θ_2. However, we can find $(\theta_1\theta_2)_{MLE}$.

16. We additionally obtained a set of clean results $\mathcal{D}_c$ of size N_c, where x is directly observed without the reporter in the middle. Given that there are c_h heads and c_t tails, estimate θ_1 and θ_2 by MLE taking the two data sets into account. Note that the likelihood is $L(\theta_1, \theta_2) = p(\mathcal{D}_r, \mathcal{D}_c \mid \theta_1, \theta_2)$.

Assuming conditional independence we have that,

$$p(\mathcal{D}_r, \mathcal{D}_c \mid \theta_1, \theta_2) = p(\mathcal{D}_r \mid \theta_1, \theta_2)p(\mathcal{D}_c \mid \theta_1, \theta_2) \tag{32}$$

So then we have,

$$\begin{aligned} L(\theta_1, \theta_2) &= p(\mathcal{D}_r \mid \theta_1, \theta_2)p(\mathcal{D}_c \mid \theta_1, \theta_2) \\ &= (\theta_1\theta_2)^{n_h}(1-\theta_1\theta_2)^{n_t}\theta_1^{c_h}(1-\theta_1)^{c_t} \end{aligned} \tag{33}$$

And furthermore,

$$log(L(\theta_1,\theta_2)) = n_h log(\theta_1) + n_h log(\theta_2) + n_t log(1-\theta_1\theta_2) + c_h log(\theta_1) + c_t log(1-\theta_1) \quad (34)$$

First, we find $\theta_{2,MLE}$ as follows,

$$\frac{\partial log(L(\theta_1,\theta_2))}{\partial\theta_2} = \frac{n_h}{\theta_2} - \frac{n_t\theta_1}{1-\theta_1\theta_2} \quad (35)$$

Which setting equal to zero and solving implies,

$$\theta_{2,MLE} = \frac{n_h}{\theta_{1,MLE}(n_h+n_t)} \quad (36)$$

Now we attempt to find $\theta_{1,MLE}$ through the same method of differentiation, though we replace all of the θ_2 with $\theta_{2,MLE}$ found above.

$$\frac{\partial log(L(\theta_1,\theta_2))}{\partial\theta_1} = \frac{n_h}{\theta_1} - \frac{n_t\theta_2}{1-\theta_1\theta_2} + \frac{c_h}{\theta_1} - \frac{c_t}{1-\theta_1} \quad (37)$$

Now we substitute the $\theta_{2,MLE}$ and find after simplifying that,

$$\theta_{1,MLE} = \frac{c_h}{c_h+c_t} \quad (38)$$

And therefore,

$$\theta_{2,MLE} = \frac{n_h(c_h+c_t)}{c_h(n_h+n_t)} \quad (39)$$

17. Since the clean results are expensive, we only have a small number of those and we are worried that we may overfit the data. To mitigate overfitting we can use a prior distribution on θ_1 if available. Let's imagine that an oracle gave us the prior $p(\theta_1) = \text{Beta}(h,t)$. Derive the MAP estimates for θ_1 and θ_2.

Assuming the prior given by the beta distribution, we have that,

$$p(\theta_1|\mathcal{D}_c) = \frac{p(\mathcal{D}_c|\theta_1)p(\theta)}{p(\mathcal{D}_c)} \tag{40}$$

$$p(\theta_1|\mathcal{D}_c) \propto \theta_1^{c_h}(1-\theta_1)^{c_t}\theta_1^{h-1}(1-\theta_1)^{t-1} \tag{41}$$

$$p(\theta_1|\mathcal{D}_c) \propto \theta_1^{c_h+h-1}(1-\theta_1)^{c_t+t-1} = Beta(c_h+h, c_t+t) \tag{42}$$

Then we have that $\theta_{1,MAP}$ is given by the mode of $p(\theta_1|\mathcal{D}_c)$. The mode of $Beta(c_h+h, c_t+t)$ is given by,

$$\frac{\alpha-1}{\alpha+\beta-1} = \frac{c_h+h-1}{c_h+c_t+h+t-2} \tag{43}$$

So,

$$\theta_{1,MAP} = \frac{c_h+h-1}{c_h+c_t+h+t-2} \tag{44}$$

Furthermore, we use the following result from above,

$$\theta_{2,MLE} = \frac{n_h}{\theta_1(n_h+n_t)} \tag{45}$$

Yielding,

$$\theta_{2,MLE} = \frac{n_h(c_h+c_t+h+t-2)}{(c_h+h-1)(n_h+n_t)} \tag{46}$$

Thus we have found $\theta_{1,MAP}$ and $\theta_{2,MLE}$.