2 From scores to conditional probabilities

2.1

Given, $\pi(x) = p\,(y = 1 \mid x)$

$$\begin{aligned}
\mathbb{E}_y\left[\ell\left(yf(x)\right) \mid x\right] &= \ell\left(f(x)\right) p\left(y = 1 \mid x\right) + \ell\left(-f(x)\right) p\left(y = -1 \mid x\right) \\
\Rightarrow \mathbb{E}_y\left[\ell\left(yf(x)\right) \mid x\right] &= \ell\left(f(x)\right) \pi(x) + \ell\left(-f(x)\right)\left(1 - \pi(x)\right) \\
\Rightarrow \mathbb{E}_y\left[\ell\left(yf(x)\right) \mid x\right] &= \pi(x)[\ell\left(f(x)\right) - \ell\left(-f(x)\right)] + \ell\left(-f(x)\right)
\end{aligned}$$

2.2

Given, $\ell\left(yf(x)\right) = e^{-yf(x)}$,
We use the formula from the last question:

$$\mathbb{E}_y[\ell(yf(x))|x] = \pi(x)[\ell(f(x)) - \ell(-f(x))] + \ell(-f(x))$$
$$\Rightarrow \mathbb{E}_y[\ell(yf(x))|x] = \pi(x)[e^{-f(x)} - e^{f(x)}] + e^{f(x)}$$

To find the bayes prediction function, we differentiate and equate to zero:

$$\frac{d}{df(x)}[\pi(x)(e^{-f(x)} - e^{f(x)}) + e^{f(x)}] = 0$$
$$\Rightarrow -\pi(x)[e^{-f(x)} + e^{f(x)}] + e^{f(x)}] = 0$$
$$\Rightarrow e^{2f(x)} = \frac{\pi(x)}{1-\pi(x)}$$

Then,

$$f^*(x) = \frac{1}{2} ln\left(\frac{\pi(x)}{1-\pi(x)}\right)$$

And shuffling the above equality gives us:

$$\pi(x) = \frac{1}{1+e^{-2f^*(x)}}.$$

2.3

Now, given, $\ell\left(y, f(x)\right) = \ln\left(1 + e^{-yf(x)}\right)$,
We repeat the process from the last question:

$$\mathbb{E}_y[\ell(yf(x))|x] = \pi(x)[\ell(f(x)) - \ell(-f(x))] + \ell(-f(x))$$

We denote f(x) as y^*

$$\Rightarrow \mathbb{E}_y[\ell(yf(x))|x] = \pi(x)[ln(1 - e^{-y^*}) - ln(1 + e^{y^*})] + ln(1 + e^{y^*})$$

Simplifying,

$$= \pi(x)e^{-y^*} + ln(1 + e^{y^*})$$
$$= -\pi(x)y^* + ln(1 + e^{y^*})$$

Now, differentiating w.r.t y^* and equating to zero

$$\frac{d}{dy^*}\left[-\pi(x)y^* + ln(1 + e^{y^*})\right] = 0$$

$$\Rightarrow \pi(x) = \frac{e^{y^*}}{(1 + e^{y^*})}$$

Then the Bayes function is:

$$f^*(x) = \ln\left(\frac{\pi(x)}{1 - \pi(x)}\right)$$

and the conditional probability is:

$$\pi(x) = \frac{1}{1 + e^{-f^*(x)}}.$$

3 Logistic Regression

3.1

To show $n\hat{R}_n(w) = NLL(w)$ for all $w \in \mathbf{R}^d$, we begin with $NLL(w)$:

$$NLL(w) = \sum_{i=1}^{n}[-y_i' \log \phi(w^T x_i)] + (y_i' - 1)\log(1 - \phi(w^T x_i))$$

$$= \sum_{i=1}^{n}[y_i' \log(1 + e^{w^T x_i)}] + (y_i' - 1)[\log(e^{w^T x_i}) - \log(1 + e^{w^T x_i})]$$

$$= \sum_{i=1}^{n} y_i' \log(1 + e^{w^T x_i}) - (1 - y_i')w^T x_i$$

Now, we can do a case wise analysis. Then $\forall i$

1. When $y_i = 1$, $y_i' = 1$, and we get:

$$y_i' \log(1 + e^{w^T x_i}) - (1 - y_i')w^T x_i = \log(1 + e^{-w^T x_i})$$

2. When $y_i = -1$, $y_i' = 0$, and we get:

$$y_i' \log(1 + e^{w^T x_i}) - (1 - y_i')w^T x_i = \log(1 + e^{w^T x_i})$$

Since the only thing switching is the sign of the power of e, we can write it as:

$$\sum_{i=1}^{n} y_i' \log(1 + e^{w^T x_i}) - (1 - y_i')w^T x_i = \sum_{i=1}^{n} \log(1 + e^{-y_i w^T x_i})$$

which is exatcly $n\hat{R}_n(w)$. So, $n\hat{R}_n(w) = NLL(w)$

3.2

3.2.1

To prove:

$$\text{LogSumExp}(x_1, \ldots, x_n) = x^* + \log \left[e^{x_1 - x^*} + \cdots + e^{x_n - x^*} \right].$$

We start from the L.H.S:

$$\text{LogSumExp}(x_1, \ldots, x_n) = \log \left(e^{x_1} + \cdots + e^{x_n} \right).$$

Suppose x^* is the max value as given, then, we take e^{x^*} common:

$$\text{LogSumExp}(x_1, \ldots, x_n) = \log \left(e^{x^*} [e^{x_1 - x^*} + \cdots + e^{x_n - x^*}] \right).$$

Then, by laws of logarithm:

$$\text{LogSumExp}(x_1, \ldots, x_n) = x^* + \log \left[e^{x_1 - x^*} + \cdots + e^{x_n - x^*} \right].$$

3.2.2

Since x^* is the maxium, we have $\forall i$:

$$x_i - x^* \leq 0$$

taking exponent on both side, and adding natural lowewr bound of exponent:

$$0 \leq e^{x_i - x^*} \leq 1$$

3.2.3

$\log\left[e^{x_1-x^*}+\cdots+e^{x_n-x^*}\right]$ will never be "-inf" because for the term that had x^* as the power we are left with $e^{x^*-x^*}=1$, implying:

$$e^{x_1-x^*}+\cdots+e^{x_n-x^*}\geq 1$$

So the value inside log will never be closer to zero, and in fact always greater than 1, meaning log will spit out positive values.

3.2.4

$$\log(1 + e^{-s}) = \log\left(\frac{1 + e^{-s}}{e^{-s}}\right) = \log(1 + e^{s}) - s$$

Then:

$$\log(1 + e^{-s}) = \text{LogSumExp}(0, s) - s$$

3.3

3.3.1

We use the theorem that for a continuous function, second derivative always greater or equal to zero implies convexity. Since norm is convex and sum of convex function is convex, it suffices to check the term inside the sum is convex. SO:

$$Z = \log(1 + \exp(-y_i w^T x_i)) \Rightarrow \frac{dZ}{dw} = \frac{-y_i x_i}{1 + exp(-y_i w^T x_i)}$$

$$\Rightarrow \frac{d^2 Z}{dw^2} = \frac{(y_i x^i)^2}{(1 + exp(-y_i w^T x_i))^2} \geq 0$$

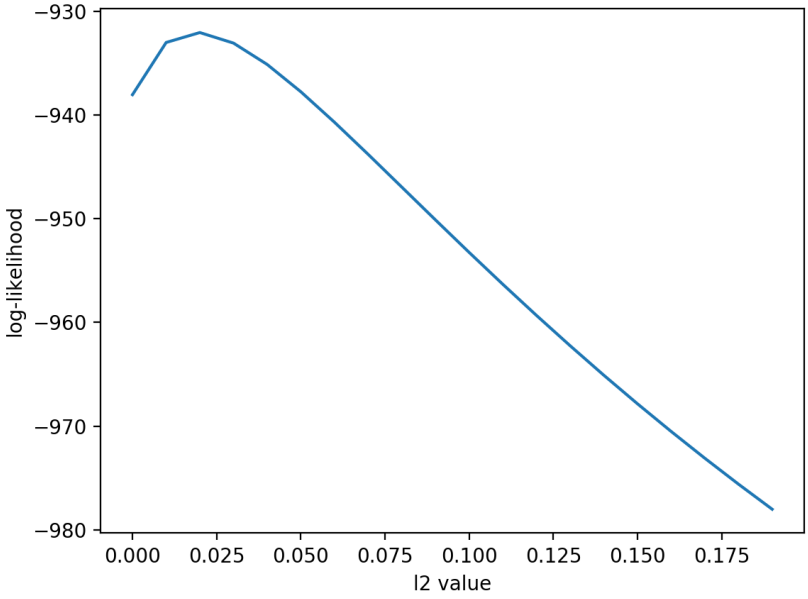
Since the function is twice diferentiable and second derivative non- negative, the function is convex.

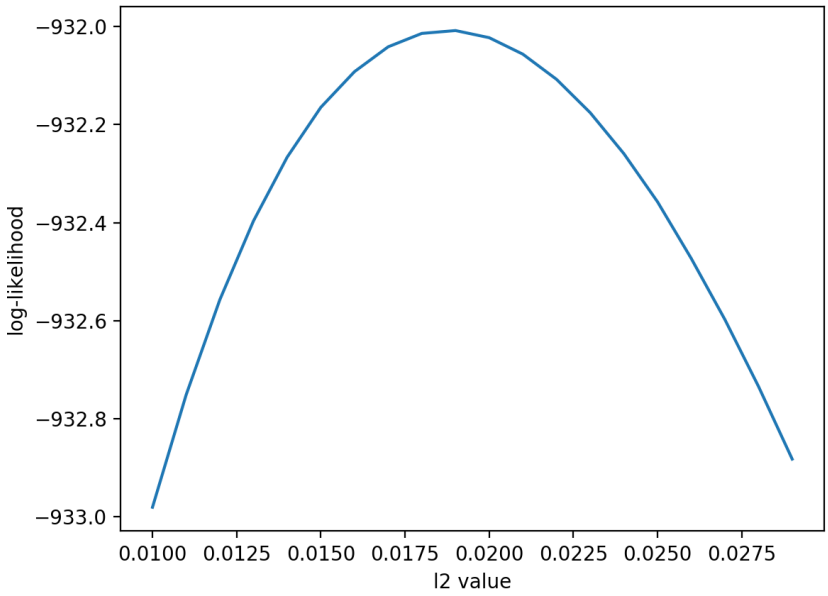
3.3.2

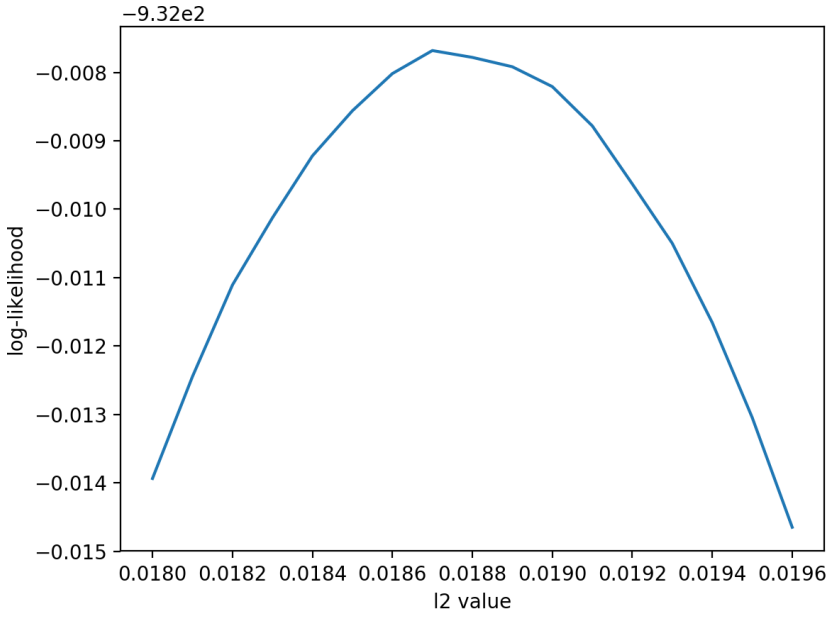
```
def f_objective(theta, X, y, l2_param):
    '''
    Args:
        theta: 1D numpy array of size num_features
        X: 2D numpy array of size (num_instances, num_features)
        y: 1D numpy array of size num_instances
        l2_param: regularization parameter

    Returns:
        objective: scalar value of objective function
    '''
    n = len(y)
    margin = 0
    obj = 0
    for i in range(n):
        if y[i] == 0:
            y[i] = -1

    for i in range(n):
        margin = -y[i]*(np.transpose(theta).dot(X[i,:]))
        obj += np.logaddexp(0, margin)
    obj = (obj/n) + l2_param*(np.transpose(theta).dot(theta))

    return obj
```

3.3.3

The log-likelihood is maximized at $\lambda = 0.0187$. The progressively zoomed-in graphs follow:

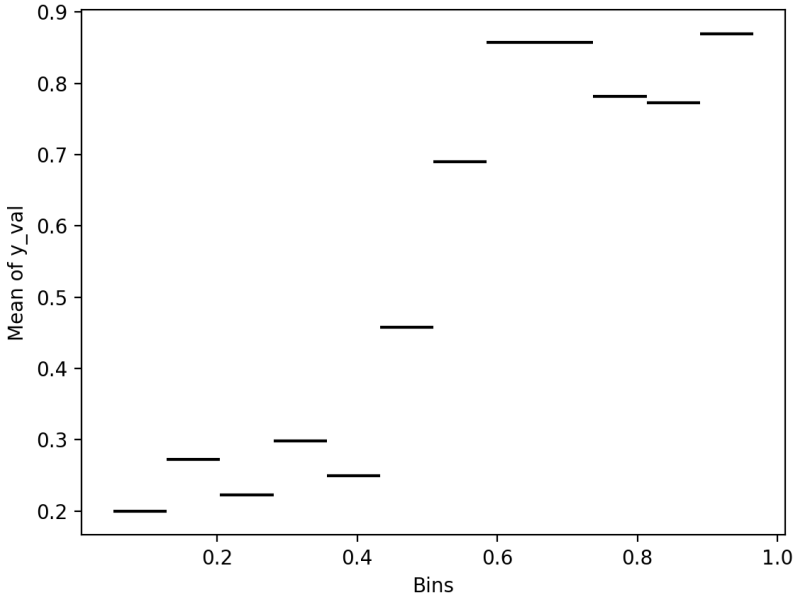





−9.32e2
−0.008
−0.009
−0.010
−0.011
−0.012
−0.013
−0.014
−0.015
log-likelihood
0.0180
0.0182
0.0184
0.0186
0.0188
0.0190
0.0192
0.0194
0.0196
l2 value

3.3.4

In the plot mean of y value in each bin is roughly increasing with the probability prediction we get, which is what we'd want.



4 Bayesian logistic regression with Gaussian prior

4.1

Here,

$$p(w \mid \mathcal{D}') \propto e^{-NLL(w)} p(w)$$

4.2

We start by finding 'w' that is the minimizer of regularized logistic regression. We differentiate and equate to 0:

$$\frac{dJ}{dw} = \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i x_i}{1 + exp(-y_i W^T x_i)} + 2\lambda W = 0$$

Consider this eq(I), and the minimzer satisfies this. Now, MAP estimate minimizes the negative log posterior from the last question. So, we are to minimize:

$$NLL - logP(w) = n\hat{R}_n(w) - logP(w)$$

Differentiating w.r.t w and equating to zero and multipling by 1/n:

$$\frac{1}{n}\sum_{i=1}^{n} \frac{-y_i x_i}{1 + exp(-y_i W^T x_i)} - \frac{P(w)'}{nP(w)} = 0$$

for the second term, we need to write it in normal distribution and simplify, which I did, but it's way to cumbersome to write here, so I'll write the end result directly:

$$\frac{1}{n}\sum_{i=1}^{n} \frac{-y_i x_i}{1 + exp(-y_i W^T x_i)} - \frac{-\Sigma^{-1} W}{n} = 0$$

where Σ is the prior variance on w as given. Then equating with eq(I), we get:

$$\frac{\Sigma^{-1} W}{n} = 2\lambda W$$

$$\Rightarrow \Sigma = \frac{I}{2n\lambda}$$

4.3

Using the final equality from the last question, and for the supposing $\Sigma = I$ gives us:

$$I = \frac{I}{2n\lambda} \Rightarrow \lambda = \frac{1}{2n}$$

5 Bayesian linear regression - Implementation

5.1

```
def likelihood_func(w, X, y_train, likelihood_var):
    '''
    Implement likelihood_func. This function returns the data likelihood
    given f(y_train | X; w) ~ Normal(Xw, likelihood_var).

    Args:
        w: Weights
        X: Training design matrix with first col all ones (np.matrix)
        y_train: Training response vector (np.matrix)
        likelihood_var: likelihood variance

    Returns:
        likelihood: Data likelihood (float)
    '''

    #TO DO

    #change to matrix
    X = np.matrix(X)
    y = np.matrix(y_train)

    n = len(y)

    for i in range(n):
        y_hat = np.transpose(w).dot(np.transpose(X[i,:]))
        temp = y[i]*(y_hat - np.logaddexp(0,y_hat))
        temp += (1-y[i])*(y_hat + (y_hat - np.logaddexp(0,y_hat)))

    likelihood = math.exp(temp)

    return likelihood
```

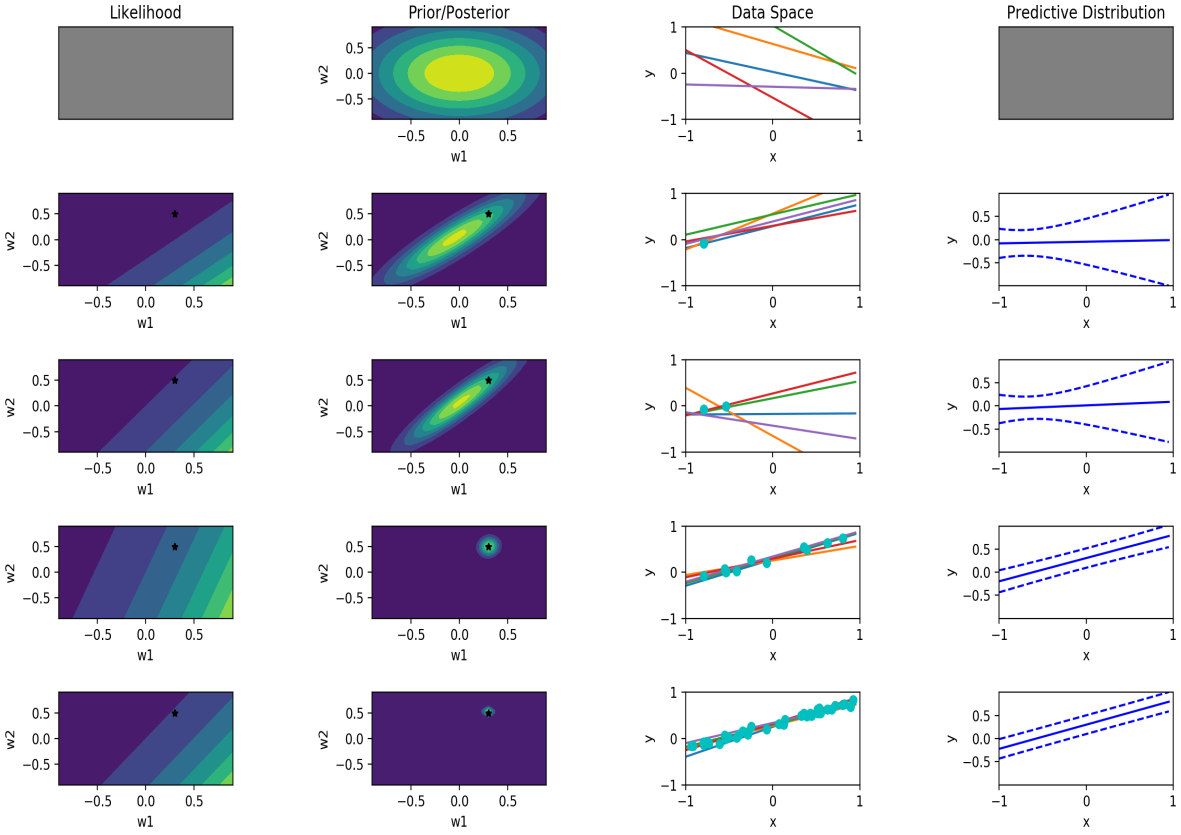
5.2

```
def get_posterior_params(X, y_train, prior, likelihood_var = 0.2**2):
    '''
    Implement get_posterior_params. This function returns the posterior
    mean vector \mu_p and posterior covariance matrix \Sigma_p for
    Bayesian regression (normal likelihood and prior).

    Note support_code.make_plots takes this completed function as an argument.

    Args:
        X: Training design matrix with first col all ones (np.matrix)
        y_train: Training response vector (np.matrix)
        prior: Prior parameters; dict with 'mean' (prior mean np.matrix)
               and 'var' (prior covariance np.matrix)
        likelihood_var: likelihood variance- default (0.2**2) per the lecture slides

    Returns:
        post_mean: Posterior mean (np.matrix)
        post_var: Posterior mean (np.matrix)
    '''

    # TO DO

    #sigma^2 is likelihood variance
    prior_mean = prior['mean']
    prior_var = prior['var']

    #TO FIND POST-MEAN
    in_matrix = X.T.dot(X) + likelihood_var*prior_var.getI()
    out_matrix = (in_matrix.getI()).dot(X.T)
    post_mean = out_matrix.dot(y_train)

    #to find post-var
    in_matrix = (1/likelihood_var)*(X.T).dot(X) + prior_var.getI()
    post_var = in_matrix.getI()


    return post_mean, post_var
```

5.3

```
def get_predictive_params(X_new, post_mean, post_var, likelihood_var = 0.2**2):
    '''
    Implement get_predictive_params. This function returns the predictive
    distribution parameters (mean and variance) given the posterior mean
    and covariance matrix (returned from get_posterior_params) and the
    likelihood variance (default value from lecture).

    Args:
        X_new: New observation (np.matrix object)
        post_mean, post_var: Returned from get_posterior_params
        likelihood_var: likelihood variance (0.2**2) per the lecture slides

    Returns:
        - pred_mean: Mean of predictive distribution
        - pred_var: Variance of predictive distribution
    '''

    # TO DO

    pred_mean = (post_mean.T).dot(X_new)
    #pred-variance
    in_matrix = (X_new.T).dot(post_var)
    pred_var = in_matrix.dot(X_new) + likelihood_var

    return pred_mean, pred_var
```

5.4


Likelihood
Prior/Posterior
Data Space
Predictive Distribution
w1
w2
x
y

5.5

As the sample size increases, our posterior distribution is much narrower and much certain. We also get a much better predictive function as the amount of data increases or uncertainty decreases. With decreasing value of sigma, the posterior distribution clusters around the true parameters, since we get increasingly confident priors. Similarly, the fluctuation of the predictive distribution from the mean prediction is also much narrower, and this more confident.

5.6

I use sklearn to run the ridge regression and upon suitable λ it our Bayesian code and ridge regression gave the same prediction function. I used $\lambda = 2 * (0.2^2)$ because we know: $\Sigma = \frac{\sigma^2}{\lambda} I$ for ridge regression to be the as the MAP.

```
sigmas_to_test = [1/2]
    for sigma_squared in sigmas_to_test:
        prior = {"mean":np.matrix([[0], [0]]),
                 "var":matlib.eye(2) * sigma_squared}

        post_mean, post_var = get_posterior_params(xtrain, ytrain, prior)
        print(post_mean)
        #ridge regression
        clf = Ridge(alpha=2*0.04)
        clf.fit(np.matrix(xtrain), np.matrix(ytrain))
        print(clf.coef_)
```