COMP 652: ASSIGNMENT 1

CARLOS G. OLIVER
260425853

1. Regression

1.1. **Q1 (c):Objective for logistic regression with L_2 regulariztion.** The logistic regression objective is the cross entropy error function between true outputs y_i and predictions $h(x_i)$ with the additional regularization term proportional to some constant λ and the L_2 norm of the weight vector $\mathbf{w}$.

$$J_{\mathbf{w}} = -\left(\sum_{i=1}^{m} y_i \log(h(\mathbf{x_i})) + (1 - y_i) \log(1 - h(\mathbf{x_i})) \right) + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \tag{1}$$

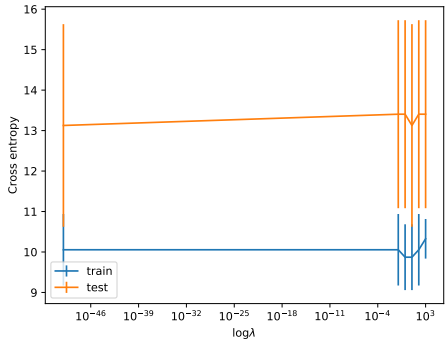
1.2. **Q1 (d): Regularization.** For all subsequent experiments, I present summary statistics of K-fold cross validation with shuffled input data. We compute a value of K so as to have approximately a training-test split ratio of $80 - 20\%$. This resulted in 4 splits. The bars one each point represent one standard deviation from the mean over the 4 splits. It is important to note that for the dataset in this assignment, the parameter space (314) is much larger than the number of instances (94) so we expect our models to display high variability. And indeed this is what we see when plotting the variance seen in K-fold cross validation **Fig. 1**. Despite the strong variability, we are still able to note key trends in the effect of regularization strength where higher λ appears to minimize cross entropy. The effect of lambda on the weight vector can also be noted in **Fig. 1c, 1d** where stronger regularization reduces the norm of the weight vector (although never to zero as we are in L2 regularization) and the values of each parameters can also be seen to converge to zero as we increase regularization strength. Overall, given the high variability in the data, it will be interesting to attempt experiments with even higher regularization strength.

1.3. **Q1 (e): Gaussian Basis Functions.** Next, we apply a univariate Gaussian basis function $\phi_j(x_i)$ to each value in input vectors x to compute a new feature mapping. Where

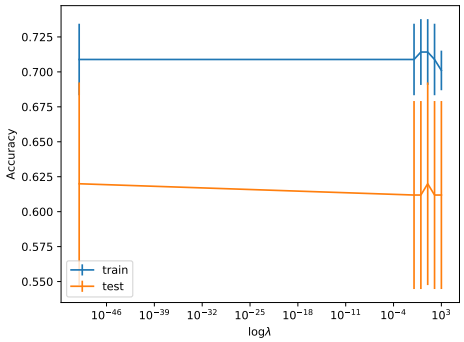
$$\phi(x)_j = exp\left[\frac{(x_i - \mu_k)^2}{2\sigma^2} \right] \tag{2}$$

We apply this basis with a fixed σ chosen from the set $\{0.1, 0.5, 1, 5, 10\}$ at each iteration. For each variable we compute 5 basis functions $(\phi_1(x), ...\phi_5(x))$ with $\mu \in \{-10, -5, 0, 5, 10\}$

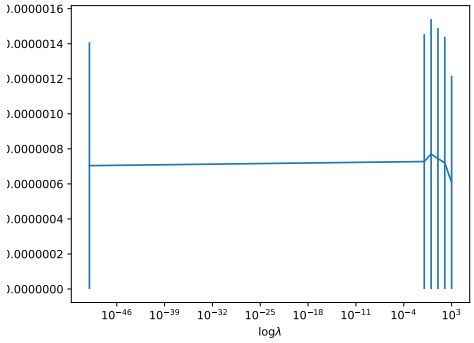
Date: February 6, 2017.



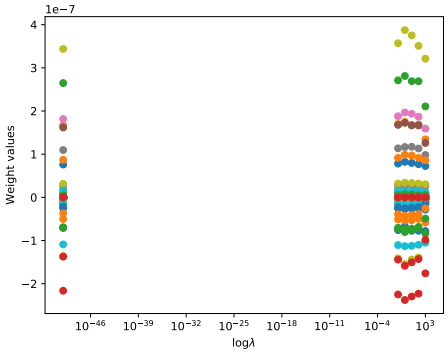
(A) Cross entropy error over regularization strength



(B) Accuracy over regularization strength



(C) Norm of weight vector over regularization strength



(D) Weight vector values over regularization strength

FIGURE 1. Effect of regularization strength on logistic regression fitting

resulting in 5 feature vectors for each original feature vector in X. This results in a new input matrix Φ.

1.4. **Q1 (f): Effect of σ on regression.** In principle, a larger σ should have an effect on the variance and bias of the model. A smaller sigma will result in strong penalties for points that are somewhat far from the center and thus increase the variance of the model to new data points. Interestingly, we note a strong increase in accuracy over all values of σ see **Fig. 2**. This is likely due to the fact that the basis function is allowing us to better capture the complexity in the data, and indeed we see that the training accuracy reaches 100%.

1.5. **Q1 (g): Basis function and regularization.** Now we add in all the basis functions for each values of μ and σ and perform logistic regression on this expanded feature space

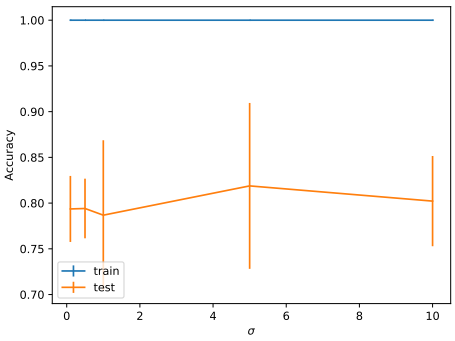


FIGURE 2. Fixed mean basis regression

Fig. 3. We note that although cross entropy appears to also have decreased from the original input space, we notice that stronger regularization is instead producing higher cross entropy. Again, we are able to observe the decrease in the L2 norm of weight vectors across all σ basis functions **Fig. 5**.

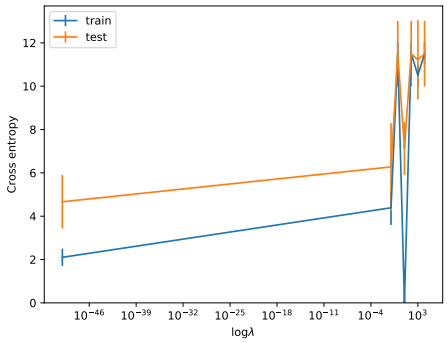


FIGURE 3. Average cross entropy with all basis functions

1.6. **Q1 (h): Gaussian basis function capturing relationship between inputs.** Instead of using a univariate Gaussian basis function to transform each input variable, we could use multivariate Gaussians to capture the relationship between inputs as covariance. We can use a similar basis as before, but introduce a covariance parameter matrix Σ that can be estimated from the data. As we would be increasing the complexity of our model, we are likely to reduce the bias but make the model more sensitive to variations in the data.

1.7. **Q1 (i): Adaptive gaussian center placement.** Previously, we have been externally fixing the centers μ_i of the Gaussian basis functions. We can instead use the data to

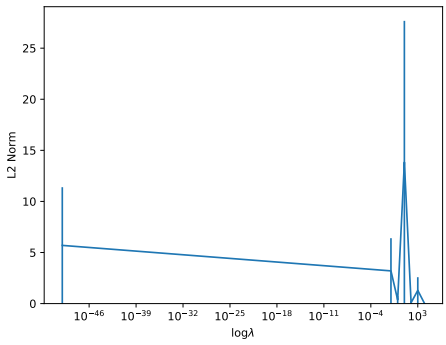


FIGURE 4. L2 norm of weight vector

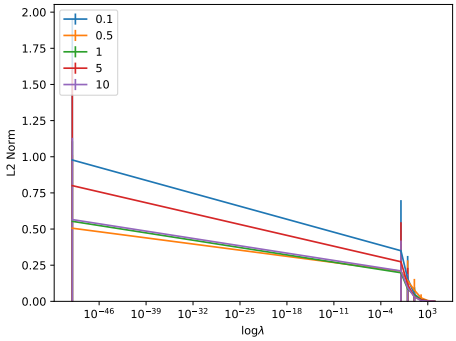


FIGURE 5. Full feature mapping σ L2 norms

adaptively compute the placement of these centers. This would effectively be a clustering task, where each cluster would define the center of a Gaussian basis. With a fixed σ, the only parameter we need to estimate is the center vector μ for each center k. Our basis functions would then take the form:

$$\phi(\mathbf{x}) = exp\left(\frac{\|\mathbf{x} - \mu\|_2^2}{2\sigma_k^2}\right) \tag{3}$$

We can iterate over all possible values of k and keep the best performing model.

input : X, λ, **y**, $[\sigma_0, ..., \sigma_j]$
output: ω
$minTestErr = \infty$;
$bestModel \leftarrow Null$;
for k *in* K **do**
  $\mu_1, ..., \mu_j \leftarrow$ `KMeans(`X, k`)`;
  $X^{'} \leftarrow$ `GaussianBasis(`$X, [\mu_1, ..., \mu_k]$`)`;
  $model, \mathbf{w} \leftarrow$ `LogisticRegression(`$X^{'}, y, \lambda$`)`;
  $currentTestErr \leftarrow$ `accuracy(`$model$, X, y`)`;
  if $currentTestErr < minTestErr$ **then**
    $bestModel \leftarrow model$
end
return $bestModel$

1.8. **Q1 (j): Convergence of algorithm.** We can expect this algorithm to converge to a global minimum with a proper optimization scheme, as the parametrization of logistic regression remains untouched and we can still used gradient based methods to solve the convex problem. The K-means procedure is simply a method for obtaining a feature mapping that is adaptive to the data and is an instance of expectation maximization algorithms.

1.9. **Q2 (a): Dual view of logistic regression.** The hypothesis $h(\mathbf{x})$ in logistic regression takes the form:

$$h(\mathbf{x}) = P(Y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{\mathbf{w^T x}}} \tag{4}$$

If we wish to obtain the probability distribution of y condition on $\mathbf{x}$ and $\mathbf{w}$ we get:

$$p(y|\mathbf{x}, \mathbf{w}) = \sigma(y\mathbf{w^T x}) = \frac{1}{1 + e^{-y\mathbf{w^T x}}} \tag{5}$$

We can then use this form to find the parameters that maximize the log likelihood of the data given a choice of model. (The following derivation is based on work published by Thomas P. Minka in 2007 [1].)

$$l(\mathbf{w}) = \sum_{i=1}^{m} \log \sigma(-y_i \mathbf{w^T x}) - \frac{\lambda}{2}\mathbf{w^T w} \tag{6}$$

In order to derive a dual expression of this objective, we aim to find a tight linear upper bound parametrized by a new set of parameters α_i. Such a function will let us reverse the order of the optimization while maintaining the same optima. In our case, if we wish to maximize $l(\mathbf{w})$ we can instead minimize some function $l(\mathbf{w}, \alpha)$ that is an upper bound to the first. We can then find the minimal value of $l(\mathbf{w}, \alpha)$ that maximizes $l(\mathbf{w})$.

We use the function:

$$H(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha), \quad \alpha \in [0,1] \tag{7}$$

Plotting $H(x)$ we see that the function is quadratic.

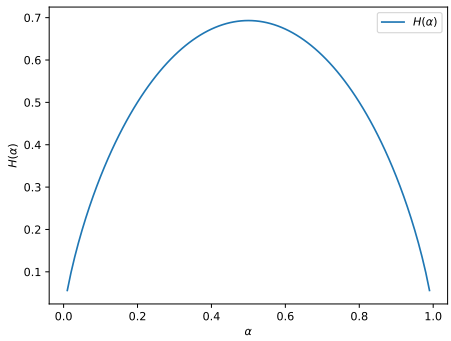


FIGURE 6. Parameter function

Using $H(x)$ we can bound $\log(\sigma(x))$ above as follows:

$$\log \sigma(x) \leq \alpha x - H(x) \tag{8}$$

Plotting these functions with $\alpha = \{0, 0.5, 1\}$ we see that indeed, $\log \sigma(x)$ is tightly bounded above.

We can then apply these constraints to the original log likelihood function optimization problem:

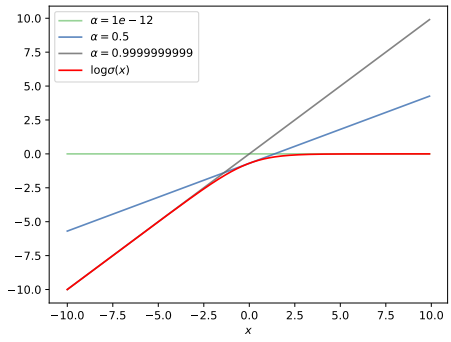


FIGURE 7. Upper bounds on log likelihood function

$$l(\mathbf{w}) = \sum_{i}^{m} \log \sigma(y_i \mathbf{w}^\mathbf{T} \mathbf{x}) - \frac{\lambda}{2} \mathbf{w}^\mathbf{T} \mathbf{w} \leq l(\mathbf{w}, \alpha) \tag{9}$$

$$\text{Where} \quad l(\mathbf{w}, \alpha) = \sum_{i}^{m} \alpha_i y_i \mathbf{w}^\mathbf{T} \mathbf{x_i} - H(\alpha_i) - \frac{\lambda}{2} \mathbf{w}^\mathbf{T} \mathbf{w} \tag{10}$$

We can now write the problem as a minimization problem over α

$$\min_{\alpha} \max_{\mathbf{w}} l(\mathbf{w}, \alpha) \tag{11}$$

Taking the derivative of $l(\mathbf{w}, \alpha)$ with respect to $\mathbf{w}$ and setting to zero we get an expression for $\mathbf{w}$

$$\mathbf{w}(\alpha) = -\lambda \sum_{i}^{m} \alpha_i y_i \mathbf{x_i} \tag{12}$$

Which we can plug into the original $l(\mathbf{w}, \alpha)$ and now optimize with respect to α to obtain

$$J(\alpha) = \frac{1}{2\lambda} \sum_{i,j}^{m,n} \alpha_i \alpha_j y_i y_j \mathbf{x_j^T} \cdot \mathbf{x_i} - \sum_{i}^{m} H(\alpha_i) \tag{13}$$

This is the dual view of logistic regression, where the objective is expressed in terms of weighted dot products between input instances and not in terms of weight vector $\mathbf{w}$. In this form, we can apply any valid kernel to the dot product $\mathbf{x_i^T} \cdot \mathbf{x_j}$ as $K(\mathbf{x_i}, \mathbf{x_j})$. It is important to note that $\alpha \in \mathbb{R}^m$ where m is the number of input instances, and $\mathbf{w} \in \mathbb{R}^n$ where n is the number of features. When using kernel methods n can grow exponentially, making the dual form more computationally tractable when the number of instances is low compared to the number of features.

In order to solve for the optimal α Minka proposes to use coordinate wise Newton method instead of Newton's method to avoid inverting the Hessian.

$$g_i = \frac{dJ(\alpha)}{d\alpha_i} = y_i \mathbf{w}(\alpha)^T \mathbf{x_i} + \log\left(\frac{\alpha_i}{1 - \alpha_i}\right) \tag{14}$$

Using the first derivative we can iteratively step in the direction of the gradient over each α_i.

$$\alpha_i^{new} = \alpha_i - \frac{g_i}{\lambda^{-1} \mathbf{x_i}^T \mathbf{x_i} + \frac{1}{\alpha_i(1-\alpha_i)}} \tag{15}$$

Having obtained the vector α through this procedure, we can make predictions using the original logistic model with a kernel.

$$P(Y = 1|x, \alpha) = \sigma\left(\sum_{i}^{m} \alpha_i K(\mathbf{x}, \mathbf{x_i})\right) \tag{16}$$

1.10. **Q2 (b): Implementation of kernelized logistic regression.** Please see the `Python` script `q2.py` for a full implementation of this method.

1.11. **Q2 (c): Kernelized logistic regression vs. non-kernel logistic regression results.** I am not including results for the performance of my implementation of kernelized logistic regression as it is clear that something is not working correctly (the full code is regardless, still included in the `Src` directory). All accuracies irrespective of test or training set are around 0.3 throughout k-fold cross validation (data not shown, see `Data` folder for data files). With more time I will continue working to fix any errors there may be in the code. However, it is also possible that the method for solving for the α using the coordinate ascent I implemented is not converging adequately. I tried updating α a total of 10 and 25 times with no significant increase in performance. So it is possible that many more updates are required, however this was beginning to take up a significant amount of computation time. The author in [1] proposes a prioritized update scheme that might help the algorithm converge faster.

Assuming the implementation had worked as expected, I can speculate on the performance as a function of the width of the kernel. Certainly a higher degree polynomial kernel will be more prone to overfitting which would become apparent in the increased performance on the training set as compared with the test set. The degree at which this would occur would have to be determined by cross validation. Such methods are especially well suited or the kind of dataset presented in this assignment where the feature space is much larger than the number of observations.

k-fold	Train	Test
1	0.680	0.5
2	0.691	0.625
3	0.747	0.645
4	0.715	0.677

TABLE 1. Regular Logistic Regression Accuracy $\lambda = 1$

1.12. **Q2 (d): Advantages and disadvantages of kernelized logistic regression (KLR).** Again, since I was unable to produce interpretable data for KLR, these responses are theoretical.

Advantages:

- The addition of kernels allows for a priori field knowledge to be coded into the models and produce more accurate models.
- Non-linear patterns can be modelled at relatively low computational costs

Disadvantages:

- The computational cost associated with kernel methods scales with input size, in this example input size was quite low so computation was not very expensive. However, already with 25 updates on α the slowdown was noticeable. Many data sets might include much larger data instances and computation will be much more costly.

1.13. **Q3: (a) Kernels.** *Statement:* $K(\mathbf{x}, \mathbf{z}) = aK_1(\mathbf{x}, \mathbf{z}) + bK_2(\mathbf{x}, \mathbf{z})$ is a kernel given that K_1 and K_2 are kernels and $a, b \in \mathbb{R}$ and $a, b > 0$.

Proof. If K_1 and K_2 are kernels then we can write them as:

$$K_1(\mathbf{x}, \mathbf{z}) = \phi^1(\mathbf{x})^T \phi^1(\mathbf{z}) \tag{17}$$

$$K_2(\mathbf{x}, \mathbf{z}) = \phi^2(\mathbf{x})^T \phi^2(\mathbf{z}) \tag{18}$$

We define two new feature mappings $\phi^a(\mathbf{x})$ and $\phi^b(\mathbf{x})$ which are scalar products of the original $\phi(\mathbf{x})$.

$$\phi^a(\mathbf{x}) = (\sqrt{a}x_1, \sqrt{a}x_2 ... \sqrt{a}x_n) \tag{19}$$

$$\phi^b(\mathbf{x}) = (\sqrt{b}x_1, \sqrt{b}x_2 ... \sqrt{b}x_n) \tag{20}$$

Taking their dot products we can construct a new kernel $K^a(\mathbf{x}, \mathbf{z})$

$$K^a(\mathbf{x}, \mathbf{z}) = \phi^a(\mathbf{x})^T \cdot \phi^a(\mathbf{z}) = (\sqrt{a}\sqrt{a}x_1 z_1 + \sqrt{a}\sqrt{a}x_2 z_2 + ... \sqrt{a}\sqrt{a}x_n z_n) = aK^2(\mathbf{x}, \mathbf{z}) \tag{21}$$

We can do the same for $\phi^b(\mathbf{x})$.

By conctatenating feature maps $\phi^a(\mathbf{x})$ and $\phi^b(\mathbf{x})$ into $\phi(\mathbf{x})$ we can express $aK_1 + bK_2$ as a dot product of this new feature map.

$$\phi(\mathbf{x})^T \cdot \phi(\mathbf{z}) = (\sqrt{a}\phi^a(x_1)\sqrt{a}\phi^a(x_1) + \sqrt{a}\phi^a(x_1)\sqrt{a}\phi^a(x_1) + ... + \sqrt{b}\phi^b(x_1)\sqrt{b}\phi^b(x_1) + \sqrt{b}\phi^b(x_1)\sqrt{b}\phi^b(x_1) + ...) \tag{22}$$

Which reduces to

$$K(\mathbf{x}, \mathbf{z}) = \phi^a(\mathbf{z})^T \cdot \phi^b(\mathbf{z}) \tag{23}$$

□

1.14. **Q3 (b) Kernels.** *Statement:* $K(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z})K_2(\mathbf{x}, \mathbf{z})$

Proof. This proof is based on the proof for the product of kernels in the Bishop textbook Ch. 6. [2].

We can expand each kernel into dot products of their respective feature vectors ϕ^1 and ϕ^2

$$\begin{aligned} K(\mathbf{x}, \mathbf{z}) &= (\phi^1(\mathbf{x})^T \cdot \phi^1(\mathbf{z}))(\phi^2(\mathbf{x})^T \cdot \phi^2(\mathbf{z})) \\ &= \sum_{i,j} \phi_i^1(\mathbf{x})\phi_j^2(\mathbf{x})\phi_i^1(\mathbf{z})\phi_j^2(\mathbf{z}) \end{aligned} \tag{24}$$

To show that this is a kernel, we need to write it as the dot product of two feature mappings. We define a new feature map $\phi'_{i,j}(\mathbf{x}) = \phi_i^1(\mathbf{x})\phi_j^2(\mathbf{x})$ that yields a feature for each pair i, j in $\mathbf{x}$. We now plug this expression back in to (24) to obtain:

$$K(\mathbf{x}, \mathbf{z}) = \phi'(\mathbf{x})\phi'(\mathbf{z}) \tag{25}$$

□

References

[1] Thomas P Minka. A comparison of numerical optimizers for logistic regression. *Unpublished draft*, 2003.
[2] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.