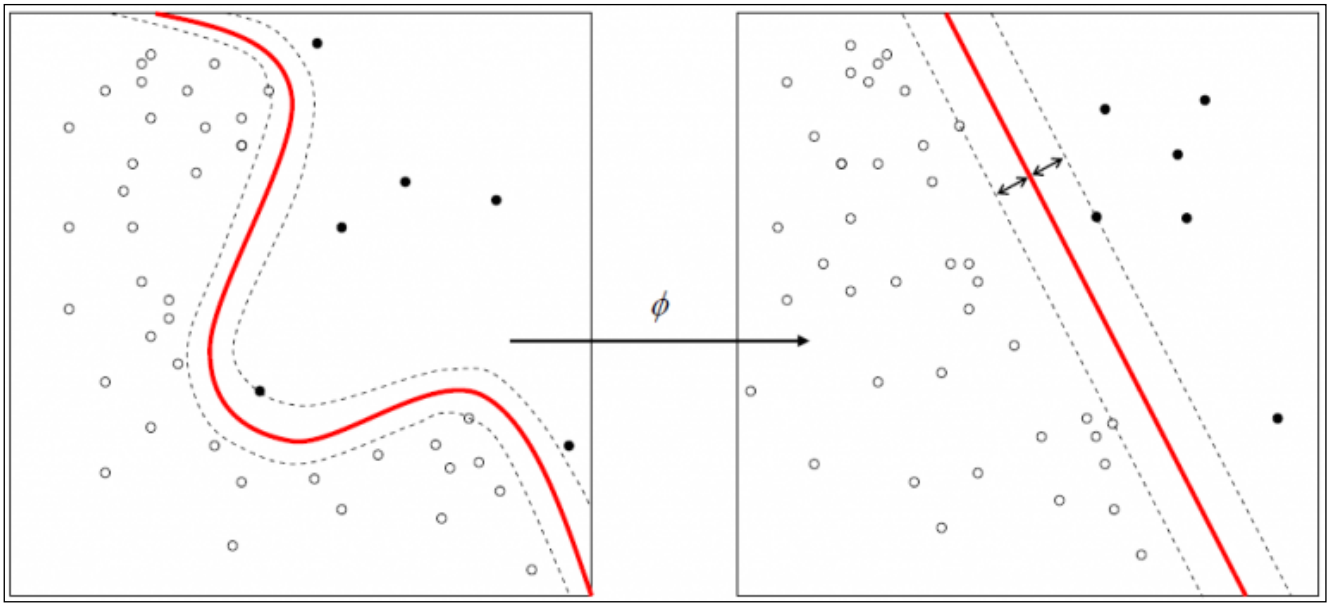
Machine Learning

Homework 1

Nathaniel Xu 徐星成 (2019280359)



1.1 Lagrange Optimization

1.1 Use Lagrange Multipliers to optimize

$$\min \; f(\underline{x}) = x_1^2 + x_2^2 - 1$$

$$\text{s.t} : \begin{cases} g(\underline{x}) = x_1 + x_2 - 1 = 0 \\ h(\underline{x}) = x_1 - 2x_2 \geq 0 \end{cases}$$

- The Lagrangian function is:

$$L(\underline{x}, \mu, \lambda) = x_1^2 + x_2^2 - 1 - \mu(x_1 - 2x_2) - \lambda(x_1 + x_2 - 1)$$

- Find the saddle points:

$$\max_{\mu,\lambda} \min_{\underline{x}} \; L(\underline{x}, \mu, \lambda)$$

Minimize L

$$\frac{\partial L}{\partial x_1} = 2x_1 - \mu - \lambda = 0 \quad \Rightarrow \quad x_1 = \frac{1}{2}\mu + \frac{1}{2}\lambda$$

$$\frac{\partial L}{\partial x_2} = 2x_2 + 2\mu - \lambda = 0 \quad \Rightarrow \quad x_2 = -\mu + \frac{1}{2}\lambda$$

- Express the Lagrangian as a function of KKT multipliers.

$$L(\mu, \lambda) = \frac{1}{4}\mu^2 + \frac{1}{2}\mu\lambda + \frac{1}{4}\lambda^2 + \mu^2 - \mu\lambda + \frac{1}{4}\lambda^2$$
$$- \mu\left(\frac{1}{2}\mu + \frac{1}{2}\lambda + 2\mu - \lambda\right)$$
$$- \lambda\left(\frac{1}{2}\mu + \frac{1}{2}\lambda - \mu + \frac{1}{2}\lambda - 1\right)$$

$$\Rightarrow \quad L(\mu, \lambda) = -\frac{5}{4}\mu^2 + \frac{1}{2}\mu\lambda - \frac{1}{2}\lambda^2 + \lambda$$

Maximize L

$$\frac{\partial L}{\partial \mu} = -\frac{5}{2}\mu + \frac{1}{2}\lambda = 0 \quad \Rightarrow \quad 5\mu = \lambda$$

$$\frac{\partial L}{\partial \lambda} = \frac{1}{2}\mu - \lambda + 1 = 0 \quad \Rightarrow \quad \mu = 2\lambda - 2$$

Saddle Points

$$\mu = \frac{2}{9}, \quad \lambda = \frac{10}{9}$$

$$x_1 = \frac{2}{3}, \quad x_2 = \frac{1}{3}$$

→ Now we check KKT conditions...

Check KKT conditions

$$\underline{x} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \end{bmatrix}^T$$

$$\mu = \frac{2}{9}$$

$$\lambda = \frac{10}{9}$$

Since $\underline{x} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \end{bmatrix}^T$ satisfy the KKT conditions the optimal solution is

$$x_1 = \frac{2}{3}$$

$$x_2 = \frac{1}{3}$$

1) Primal Feasibility

$$g(x) = x_1 + x_2 - 1$$

$$= \frac{2}{3} + \frac{1}{3} - 1 = 0 \quad \checkmark$$

$$h(x) = x_1 - 2x_2$$

$$= \frac{2}{3} - 2 \cdot \frac{1}{3}$$

$$= 0 \geq 0 \quad \checkmark$$

2) Dual Feasibility

$$\mu = \frac{2}{9} \geq 0 \quad \checkmark$$

3) Complementary Slackness

$$\mu h(x) = \frac{2}{9}(x_1 - 2x_2)$$

$$= \frac{2}{9}\left(\frac{2}{3} - 2 \cdot \frac{1}{3}\right)$$

$$= 0 = 0 \quad \checkmark$$

1.2 Stochastic Processes

1.2 We can use property of Markov Chains to solve for the expected value of this sequence of tosses.

$$\underbrace{HTT\cdots T}_{K}$$

There are $K+2$ states of interest

$$\{\phi, H, HT, HTT \cdots, \underbrace{HT\cdots T}_{K}\}$$

Let E_S be the expected value from a given state S.

Then we have the system of equations

① $E_{\underbrace{HT\cdots T}_{K-1}} = E_{HT_{K-1}} = \frac{1}{2}\cdot 1 + \frac{1}{2}(E_H+1)$

② $E_{HT_{K-2}} = \frac{1}{2}(E_{HT_{K-1}}+1) + \frac{1}{2}(E_H+1)$

③ $E_{HT_{K-3}} = \frac{1}{2}(E_{HT_{K-2}}+1) + \frac{1}{2}(E_H+1)$

$\vdots$

$E_{HT} = \frac{1}{2}(E_{HT_2}+1) + \frac{1}{2}(E_H+1)$

$E_H = \frac{1}{2}(E_{HT}+1) + \frac{1}{2}(E_H+1)$

$E_\phi = \frac{1}{2}(E_H+1) + \frac{1}{2}(E_\phi+1)$

we can use these equations to solve for E_H.

$\frac{1}{2}\times$① → ②

$\frac{1}{2}\times$② → ②

$\vdots$

by constantly substituting sequential equations we get...

$$\Rightarrow E_H = \frac{1}{2}(E_H+2) + \frac{1}{2^2}(E_H+2) + \cdots + \frac{1}{2^K}(E_H+2)$$

$$= (E_H+2)\left(\frac{1}{2}+\frac{1}{2^2}+\cdots+\frac{1}{2^K}\right)$$

$$= (E_H+2)\left(1-\frac{1}{2^K}\right)$$

$$\Rightarrow \boxed{E_H = 2^{K+1}-2}$$

Then we can solve for E_ϕ

$$E_\phi = \frac{1}{2}(2^{K+1}-2+1) + \frac{1}{2}(E_\phi+1)$$

$$\Rightarrow \boxed{E_\phi = 2^{K+1}}$$

The expected number of tosses to get $\underbrace{HT\cdots T}_{K}$ is 2^{K+1}

2. SVM

2. The primal problem of this linear SVM is already given.

$$\min_{\underline{w}, b, \underline{\xi}, \hat{\underline{\xi}}} \ \frac{1}{2}\|\underline{w}\|^2 + C\sum_i (\xi_i + \hat{\xi}_i)$$

$$\text{s.t.} \begin{cases} \underline{w}^T \underline{x}_i + b + \xi_i + \varphi - y_i \geq 0 \quad (\text{where } \varphi > 0 \text{ is a fixed constant.}) \\ -\underline{w}^T \underline{x}_i - b + \hat{\xi}_i + \varphi + y_i \geq 0 \\ \xi_i \geq 0 \\ \hat{\xi}_i \geq 0 \end{cases}$$

We can use the Lagrange Multipliers to solve this optimization problem.

↳ Use 4 vectors of lagrange multipliers for each of the 4 constraint functions.

$$L(\underline{w}, b, \underline{\xi}, \hat{\underline{\xi}}, \underline{\alpha}, \underline{\beta}, \underline{\gamma}, \underline{\sigma}) \quad \leftarrow \text{4 lagrange multipliers.}$$

$$= \frac{1}{2}\|\underline{w}\|^2 + C\sum(\xi_i + \hat{\xi}_i) - \sum \alpha_i (\underline{w}^T \underline{x}_i + b + \xi_i + \varphi - y_i) - \sum \beta_i (-\underline{w}^T \underline{x}_i - b + \hat{\xi}_i + \varphi + y_i) - \sum \gamma_i \xi_i - \sum \sigma_i \hat{\xi}_i$$

The dual solution are the saddle points of the Lagrangian.

$$\Rightarrow \arg\max_{\underline{\alpha}, \underline{\beta}, \underline{\gamma}, \underline{\sigma} \geq 0} \ \min_{\underline{w}, b, \underline{\xi}, \hat{\underline{\xi}}} L(\cdot)$$

Minimize Lagrangian:

$$\frac{\partial L}{\partial \underline{w}} = \underline{w} - \sum \alpha_i \underline{x}_i + \sum \beta_i \underline{x}_i = 0 \Rightarrow \boxed{\underline{w} = \sum (\alpha_i - \beta_i) \underline{x}_i}$$

$$\frac{\partial L}{\partial b} = -\sum \alpha_i + \sum \beta_i = 0 \Rightarrow \boxed{\sum (\alpha_i - \beta_i) = 0}$$

$$\frac{\partial L}{\partial \underline{\xi}} = \left[\frac{\partial L}{\partial \xi_i}\right] = [C - \alpha_i - \gamma_i] = [0] \Rightarrow \boxed{\alpha_i + \gamma_i = C}$$

$$\frac{\partial L}{\partial \hat{\underline{\xi}}} = \left[\frac{\partial L}{\partial \hat{\xi}_i}\right] = [C - \beta_i - \sigma_i] = [0] \Rightarrow \boxed{\beta_i + \sigma_i = C}$$

We can now form the dual problem by writing the the Lagrangian as a function of its lagrange multipliers.

$$L(\cdot) = \frac{1}{2}\|\underline{w}\|^2 + C\sum(\xi_i+\hat{\xi}_i) - \sum\alpha_i\underline{w}^T\underline{x}_i - \sum\alpha_i b - \sum\alpha_i\xi_i - \sum\alpha_i\varphi + \sum\alpha_i y_i$$
$$+\sum\beta_i\underline{w}^T\underline{x}_i + \sum\beta_i b - \sum\beta_i\hat{\xi}_i - \sum\beta_i\varphi - \sum\beta_i y_i$$
$$-\sum\gamma_i\xi_i$$
$$-\sum\sigma_i\hat{\xi}_i$$

$$= \frac{1}{2}\|\underline{w}\|^2 + C\sum(\xi_i+\hat{\xi}_i)$$
$$-\underline{w}^T\underbrace{\sum(\alpha_i-\beta_i)\underline{x}_i}_{\underline{w}} - b\underbrace{\sum(\alpha_i-\beta_i)}_{0} - \varphi\sum(\alpha_i+\beta_i) + \sum(\alpha_i-\beta_i)y_i$$
$$-\sum\alpha_i\xi_i - \sum\beta_i\hat{\xi}_i - \sum(\underbrace{C-\alpha_i}_{\gamma_i})\xi_i - \sum(\underbrace{C-\beta_i}_{\sigma_i})\hat{\xi}_i$$

$$= \frac{1}{2}\underline{w}^T\underline{w} - \underline{w}^T\underline{w} - \varphi\sum(\alpha_i+\beta_i) + \sum(\alpha_i-\beta_i)y_i$$

$$= -\frac{1}{2}\underline{w}^T\underline{w} - \varphi\sum_i(\alpha_i+\beta_i) + \sum(\alpha_i-\beta_i)y_i$$

Therefore the Dual Optimization problem is...

$$\underset{\underline{\alpha},\underline{\beta}}{\text{maximize}}: \quad -\frac{1}{2}\sum_i\sum_j(\alpha_i-\beta_i)(\alpha_j-\beta_j)\underline{x}_i^T\underline{x}_j - \varphi\sum_i(\alpha_i+\beta_i) + \sum_i(\alpha_i-\beta_i)y_i$$

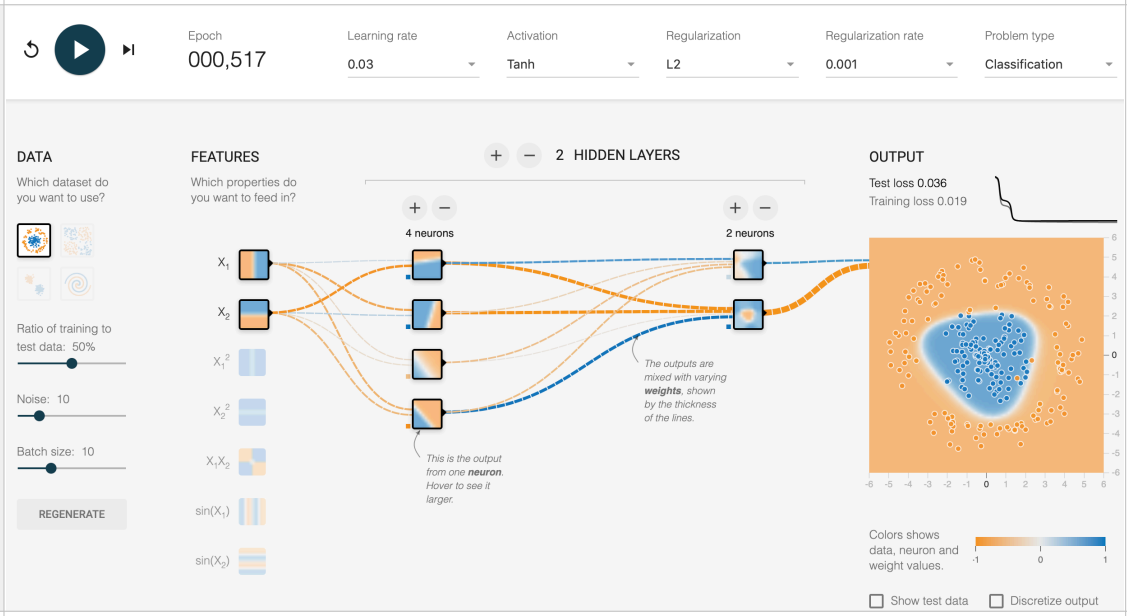
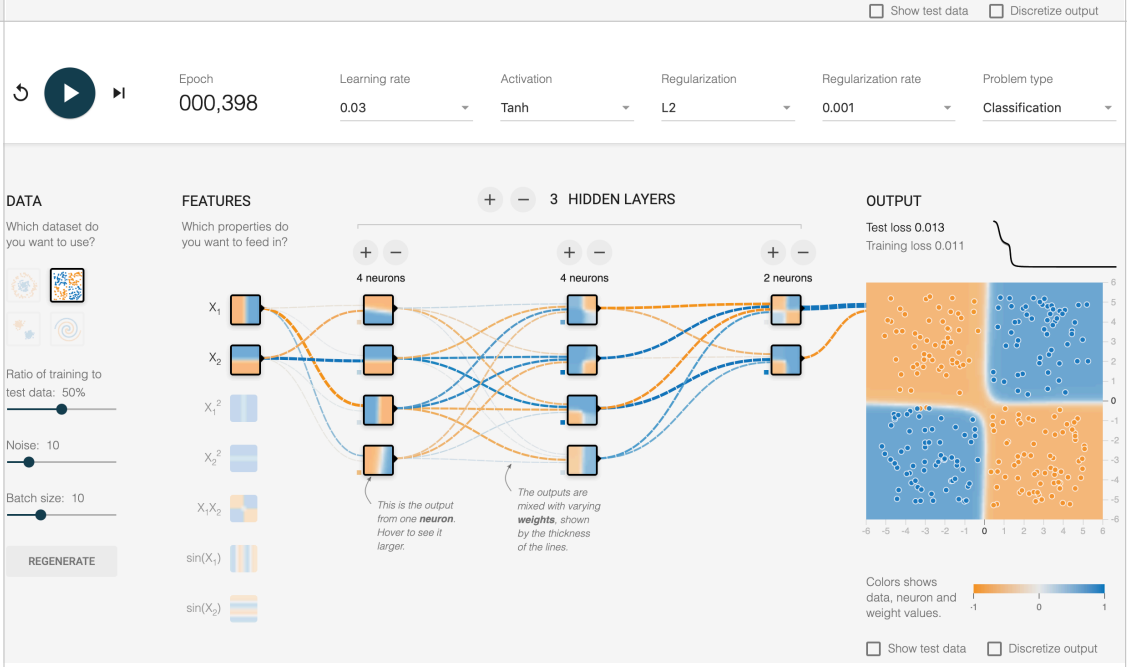
Subject to:
$$\left.\begin{aligned}&\alpha_i,\beta_i,\gamma_i,\sigma_i \geq 0\\&\textstyle\sum(\alpha_i-\beta_i)=0\\&\alpha_i+\gamma_i=C\\&\beta_i+\sigma_i=C\end{aligned}\right\} \Rightarrow \left\{\begin{aligned}&0\leq\alpha_i\,\beta_i\leq C\\&\textstyle\sum(\alpha_i-\beta_i)=0\end{aligned}\right. //$$

3. Deep Neural Networks

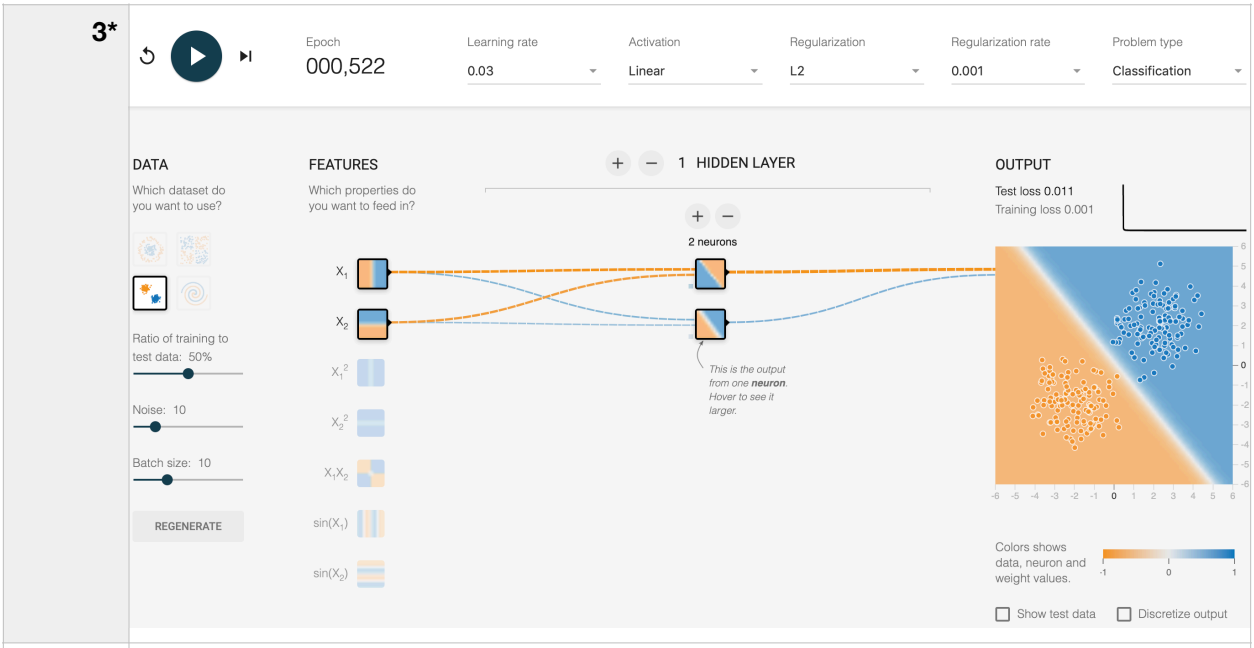
1. Identify the best classification configuration for each of the 4 datasets.

For each classification problem I have set the following static parameters:
- Learning Rate = 0.03
- Regularization = L2 norm with coefficient 0.001

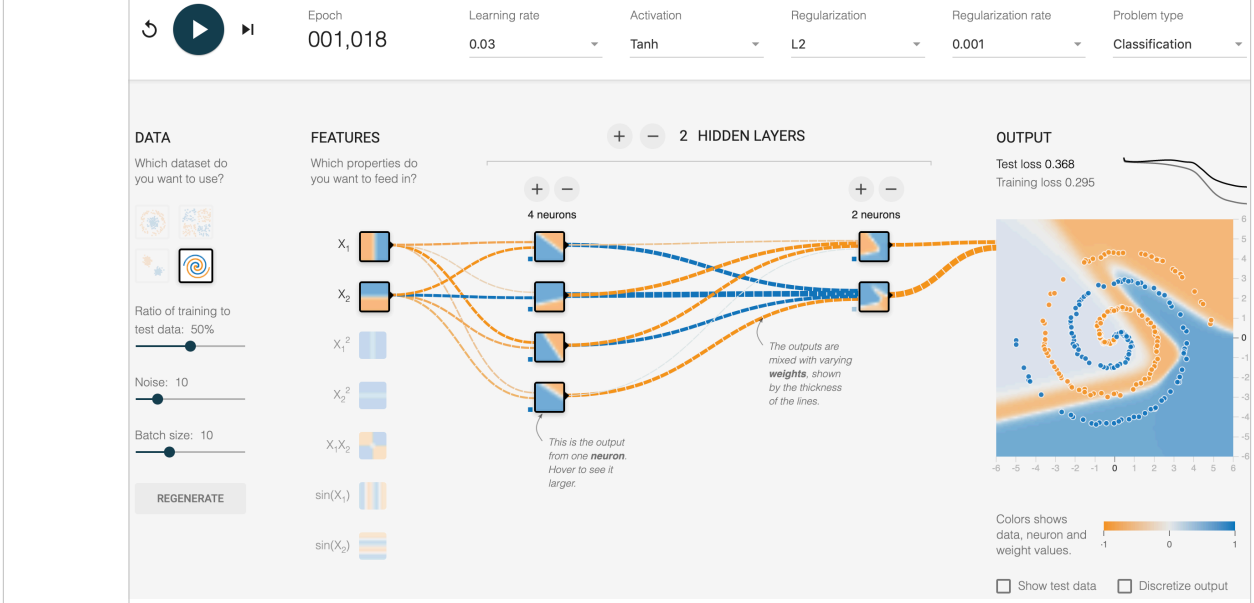
The remaining parameters have been configured specific to each dataset:
- Activation Function
- Number of Layers

Dataset	Best Configuration
1	
2	

3*



4

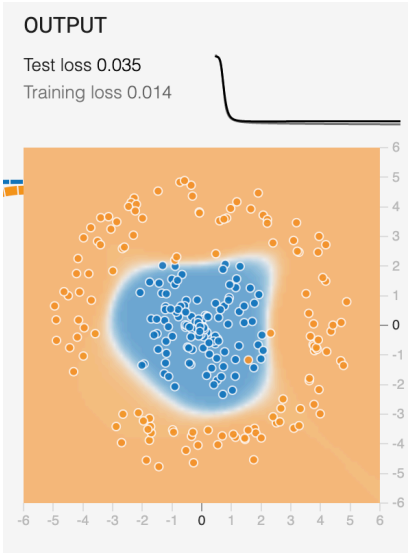


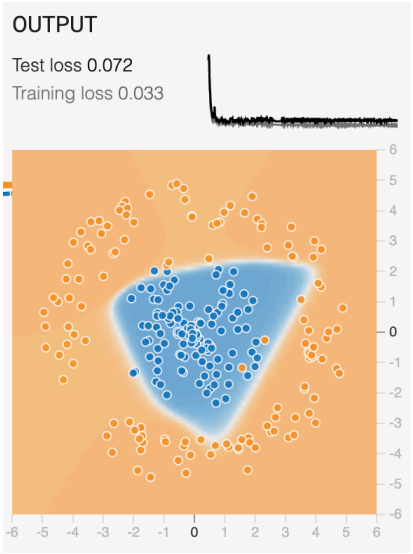
* In fact dataset 3 does not require any hidden layers to get an accurate classification. The activation function does not affect the accuracy either, however it may affect the convergence rate. I select a linear activation because it is the least computational and the data can be separated by a linear separator.

2. How does the Learning Rate, Activation Function, Number of Hidden Layers, and Regularization affect the performance and convergence rate?

Learning Rate

The learning rate may affect both the accuracy and convergence rate. With small learning rates the step sizes are minute and so oscillations around a local minimum are avoided, however the convergence rate is slower. With large learning rates the convergence rate is faster, however the larger step sizes may cause oscillations between a local minimum.

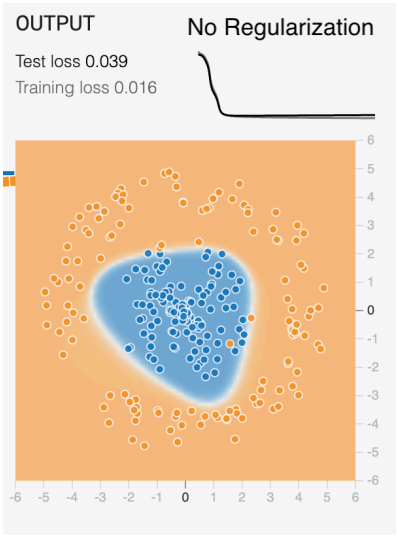


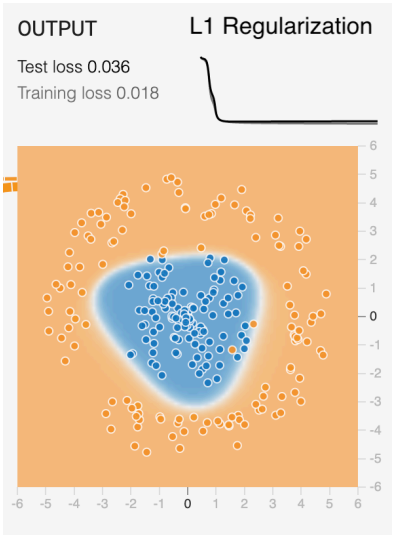


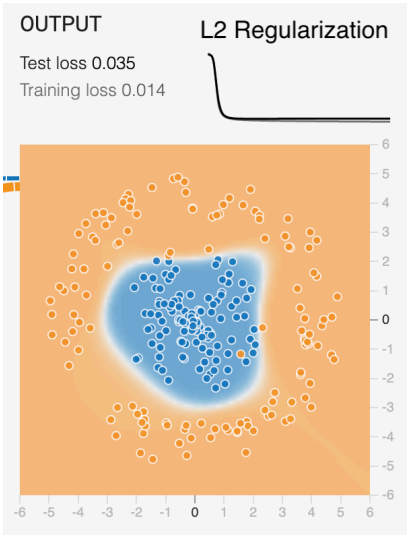
The loss graphs on the left correspond to a learning rate of 0.03, while the loss graphs on the right correspond to a learning rate of 0.3. Notice the oscillations in the loss graph with learning rate = 0.3. These are a result of gradient descent oscillating between local minimums.

Regularization

Regularization will affect the accuracy of the test data by preventing overfitting. The only difference between L1 and L2 regularization is the penalty term of the loss function. L2 regularization adds a squared magnitude penalty on the weights, while L1 regularization adds an absolute value of magnitude penalty.



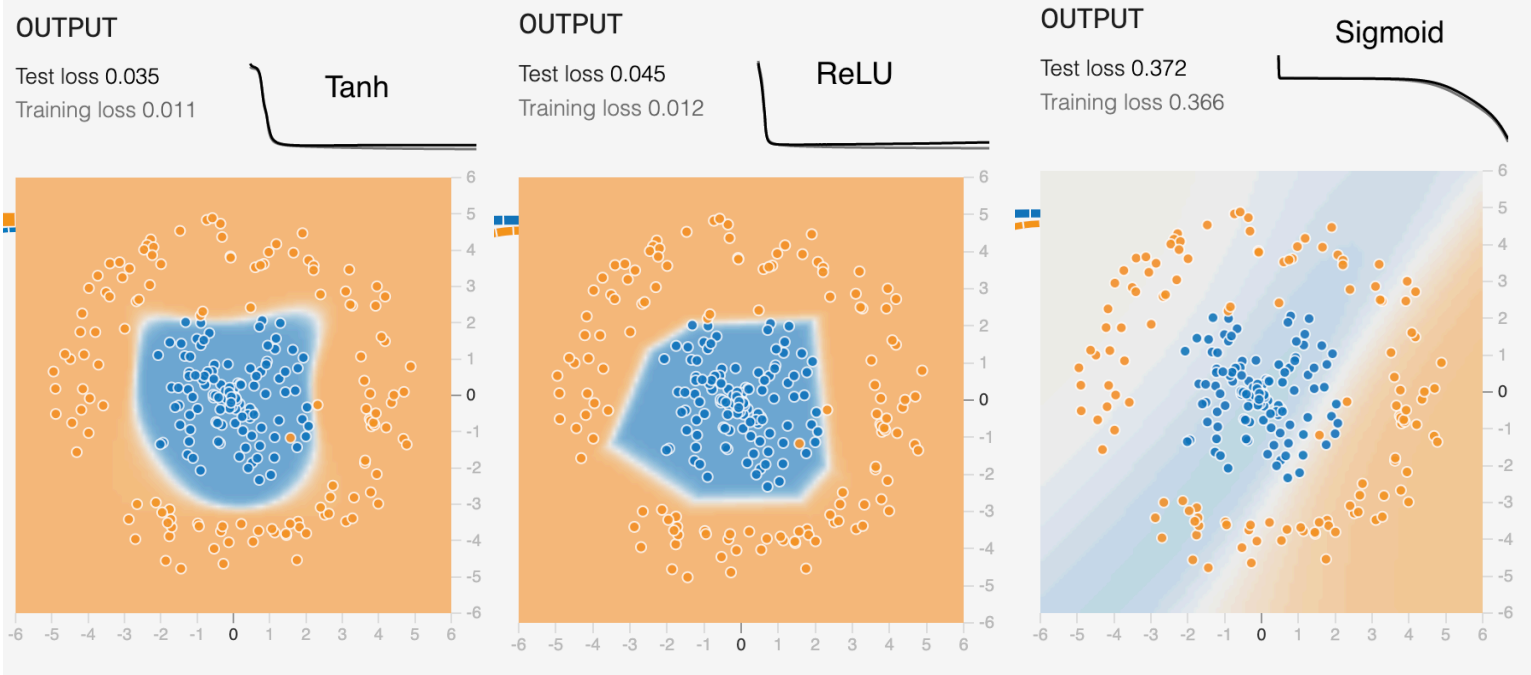




In the case of our data, an L2 regularization (coefficient=0.001) provides the least test loss. The unregularized model has the most test loss.

Activation Function

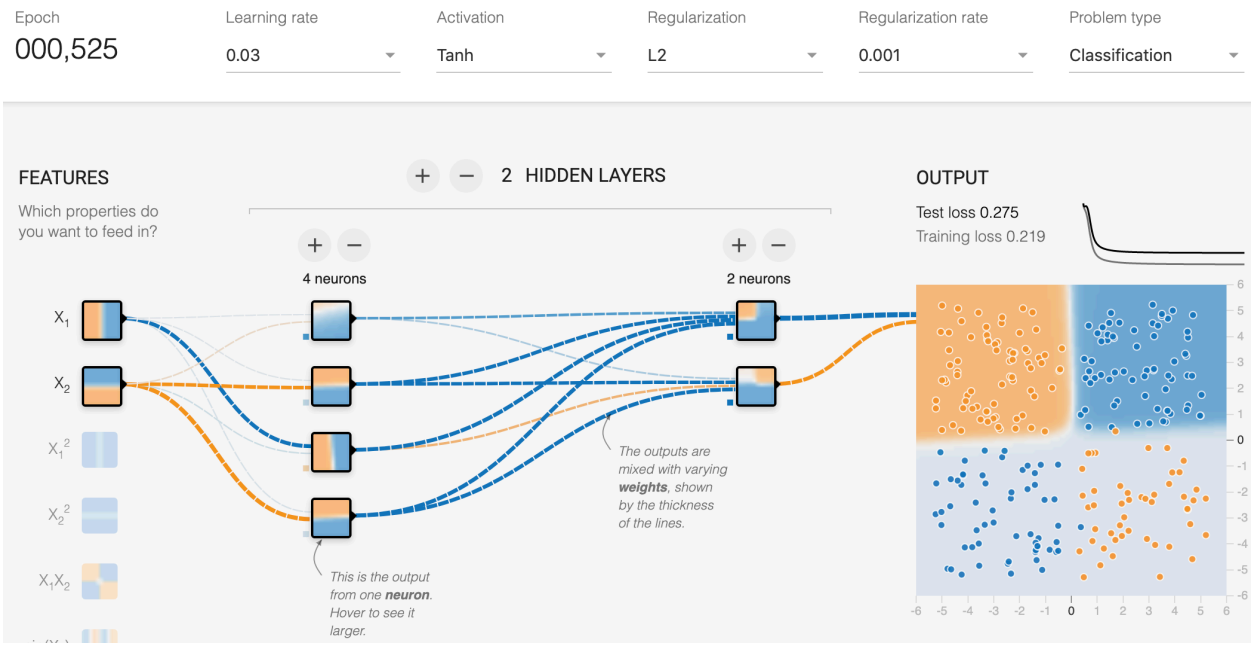
The activation function may affect both the accuracy and convergence rate. Different activation functions are more suited for different models. The results after learning the model for dataset 1 using different activation functions are shown below:

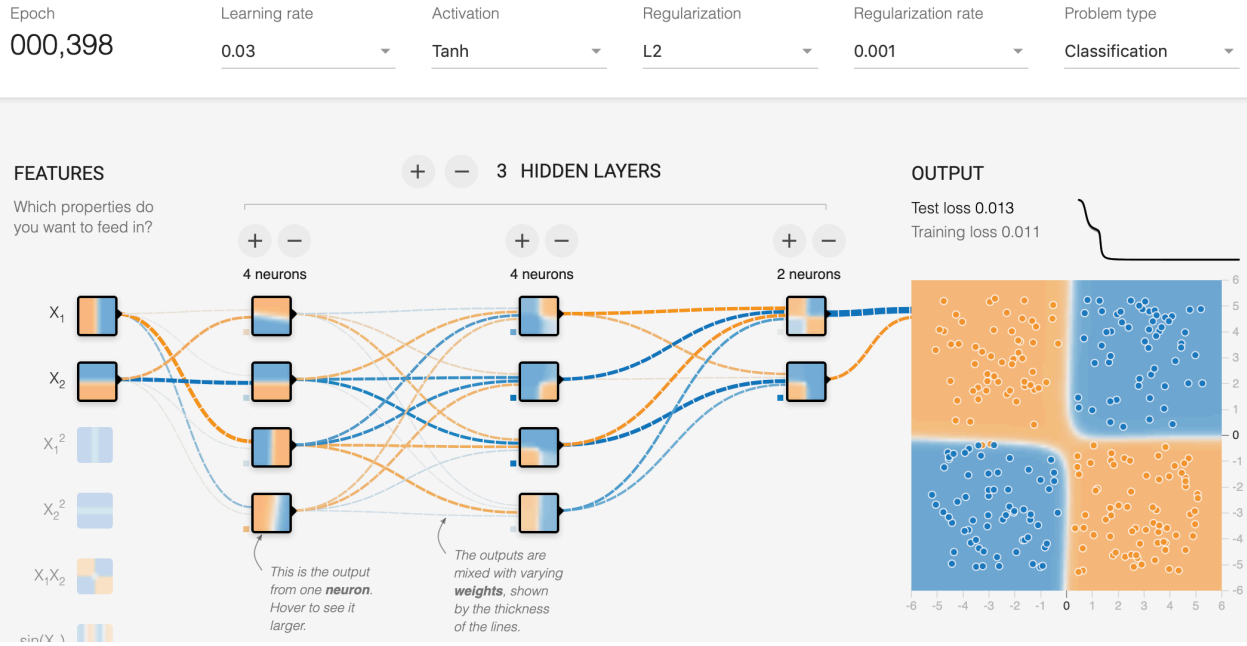


We see that different activation functions produce different classification boundaries, and therefore result in differing accuracies. Different activation functions also have differing convergence rates. All 3 models were learned using 500 epochs, however the sigmoid model had not yet converged whereas the other 2 models had.

Number of Hidden Layers

The number of hidden layers affects both the accuracy and convergence rate. The convergence rate is affected because more layers adds complexity during forward/back propagation. The accuracy is affected because while more layers generally lead to better training accuracy, it is also more susceptible to overfitting.





In the case of this XOR data, the 2-layer NN performed much worse than the 3-layer NN. This is because the 3-layer NN was able to learn a more complex separator than the 2-layer NN was able to.

4. IRLS Derivation for Logistic Regression

4. IRLS for LOGISTIC REGRESSION.

Let us derive the update step for logistic regression with an L2-regularization.

$$L(\underline{w}) = \sum_i [y_i \underline{w}^T \underline{x}_i - \log(1 + e^{\underline{w}^T \underline{x}_i})] - \frac{\lambda}{2} \|\underline{w}\|^2$$

$$\nabla_{\underline{w}} L(\underline{w}) = \sum (y_i - \mu_i) \underline{x}_i - \lambda \underline{w}$$

$$\boxed{\nabla_{\underline{w}} L(\underline{w}) = X^T (y - \underline{\mu}) - \lambda \underline{w}}$$

where $\mu_i = \varphi(\underline{w}^T \underline{x}_i) = \frac{1}{1 + e^{-\underline{w}^T \underline{x}_i}}$

$$X = \begin{bmatrix} - \underline{x}_1^T - \\ \vdots \\ - \underline{x}_n^T - \end{bmatrix}$$

The Hessian is then calculated...

$$\nabla_{\underline{w}}^2 L(\underline{w}) = \nabla_{\underline{w}} [\nabla_{\underline{w}} L(\underline{w})]^T$$

$$= \nabla_{\underline{w}} \left[\sum (y_i - \mu_i) \underline{x}_i^T - \lambda \underline{w}^T \right]$$

$$= - \sum \mu_i (1 - \mu_i) \underline{x}_i \underline{x}_i^T - \lambda$$

$$\boxed{H = \nabla_{\underline{w}}^2 L(\underline{w}) = -X^T R X - \lambda I}$$

where $R = \text{diag}(\underline{\mu}(1 - \underline{\mu}))$, $R_{ii} = \mu_i (1 - \mu_i)$

The Newton Update is then

$$\underline{w}_{t+1} \leftarrow \underline{w}_t - H_t^{-1} \nabla_{\underline{w}} L(\underline{w}_t)$$

$$\boxed{\underline{w}_{t+1} \leftarrow \underline{w}_t + (X^T R_t X + \lambda I)^{-1} \left[X^T (y - \underline{\mu}_t) - \lambda \underline{w}_t \right]}$$

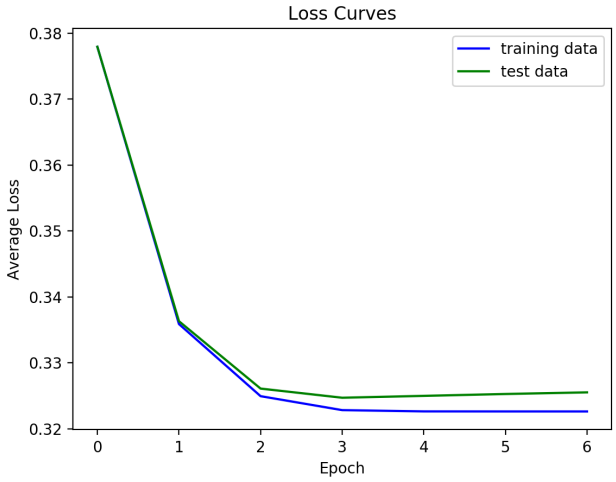
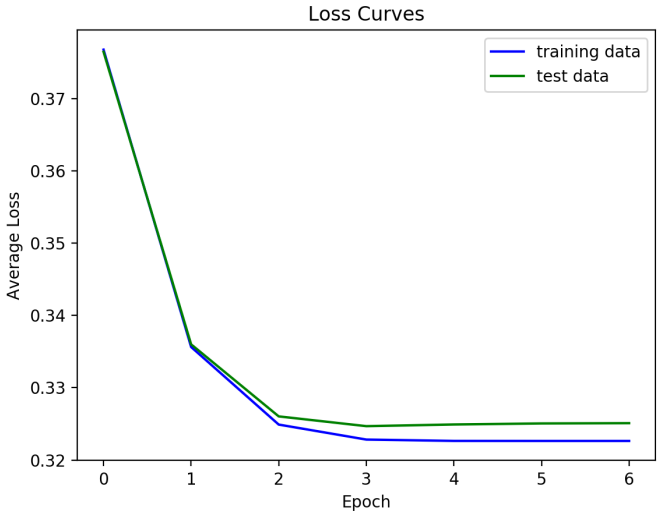
The problem with this update step is that R eventually converges to the 0 matrix if we approach a perfect model $\underline{w}^*$

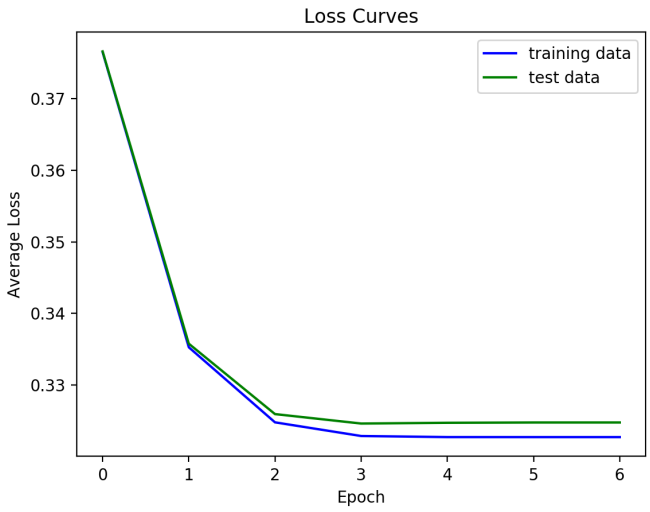
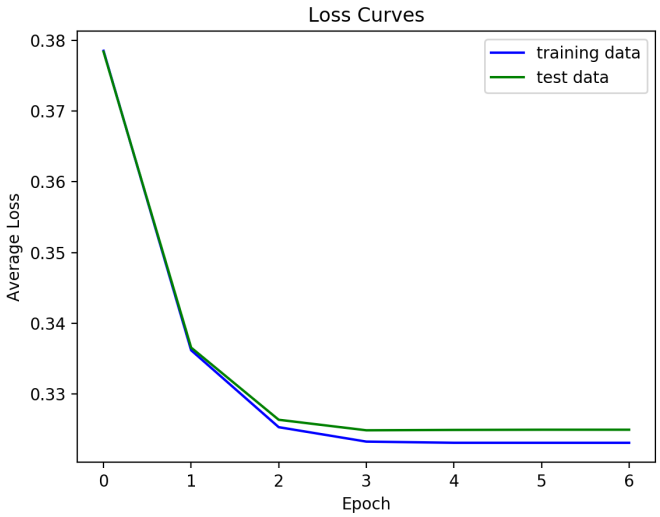
↳ This causes overflow/underflow that truncates very small R_{ii} values to 0 (ie. $1e^{-60} \rightarrow 0$), which in turn causes the Hessian to be singular and non-invertable.

A workaround to this issue is by taking the pseudo-inverse rather than the inverse to get appropriate update steps.

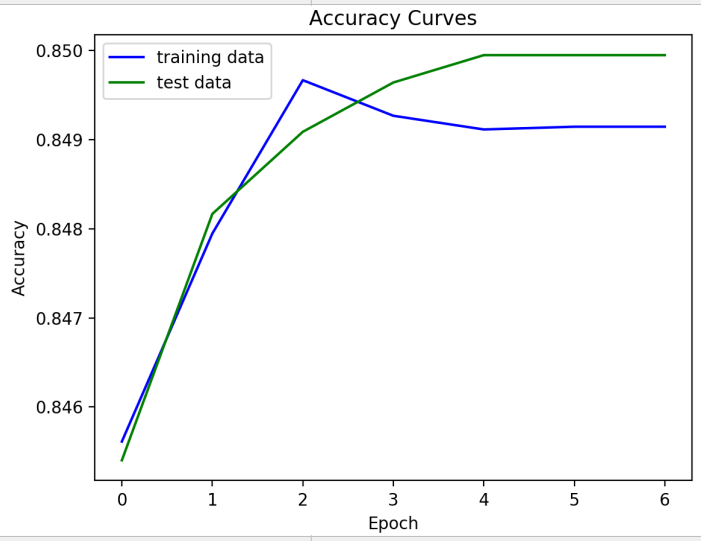
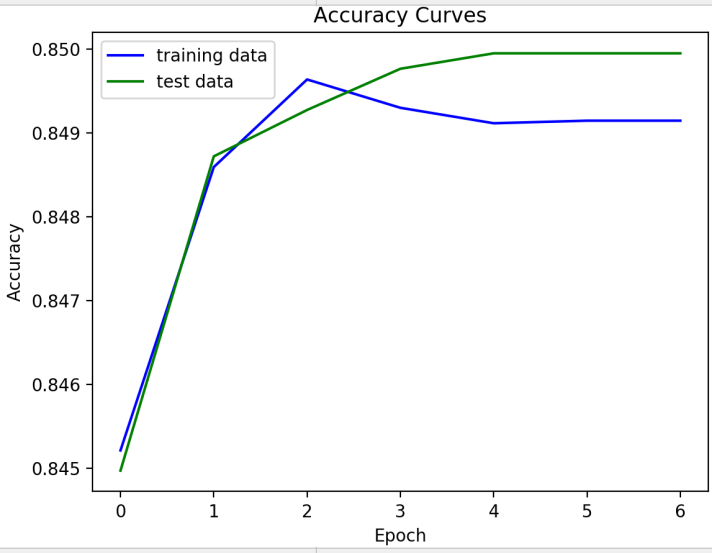
4. IRLS for Logistic Regression

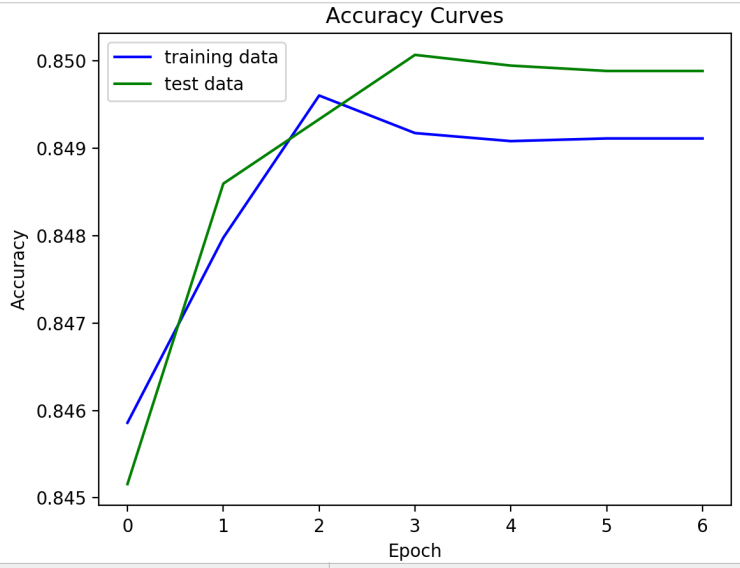
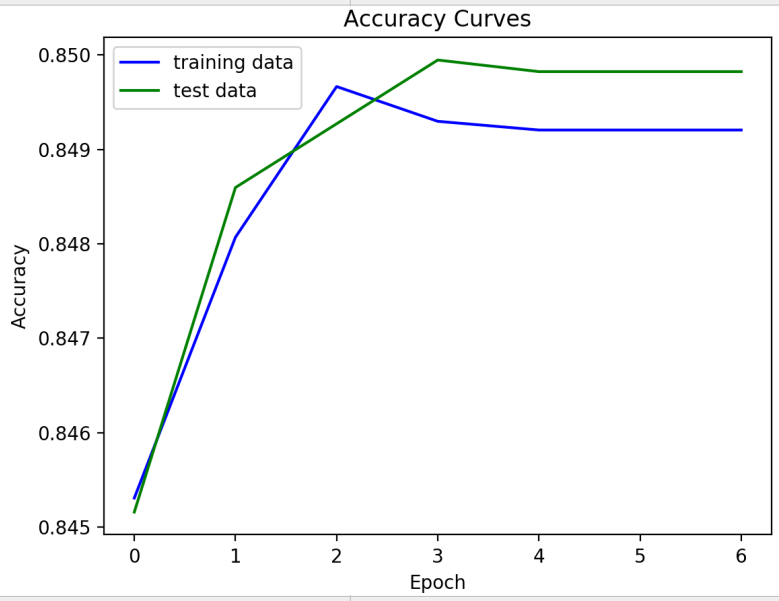
The tables below show the performance of the logistic regression model trained using IRLS. The model was cross validated on a range of regularization values.
The loss is taken as the average regularized negative log likelihood, and the accuracy is the percentage of correct classifications.

Loss		
	Training Data	**Test Data**
λ = 0		
	Training Data `0.32262165298659623`	Test Data `0.3255160134119026`
λ = 0.01		
	Training Data `0.32264071790717624`	Test Data `0.325100147085358`

λ = 0.1		
	Training Data `0.32274217581254006`	Test Data `0.3247828506234361`
λ = 0.5		
	Training Data `0.32305598581775247`	Test Data `0.3249073538941429`

Accuracy

	Training Data	Test Data
λ = 0		
	Training Data `0.8491446822886275`	Test Data `0.8499477919046742`
λ = 0.01		
	Training Data `0.8491446822886275`	Test Data `0.8499477919046742`

λ = 0.1		
	Training Data 0.8491139707011456	Test Data 0.8498863706160555
λ = 0.5		
	Training Data 0.8492061054635914	Test Data 0.8498249493274369

Performance Analysis

Looking at the loss curves, it is evident that the addition of the L2-regularization helped prevent slight overfitting of the training data. For $\lambda=0$ the test loss seems to be slightly increasing after the training loss converged. This suggests that we are slightly overfitting our training data. With increased regularization values, this phenomenon was mitigated.

In terms of which regularization value performed the best, we notice that $\lambda=0$, 0.01 produced the highest test accuracies, and $\lambda=0.1$ produced the lowest test loss. However, these parameters only account for slight performance differences. In all cases, the test accuracy was higher than the training accuracy. This information suggests that a regularization parameter in the range [0.01, 0.1] would best suit this model.

In terms of the performance of the IRLS algorithm, both training and test data converged after just 4 iterations. The number of iterations is significantly less that regular gradient descent methods, however the runtime is still comparable to that of gradient descent. This is because of the complexity of taking the (pseudo) inverse of the Hessian. Perhaps quasi-newton methods are faster in this regard.