Model Answers Exercises May 3rd

Exercise 1

For the first exercise of the day, it is proposed that you come back to the demo software: https://playground.tensorflow.org, and we will use the spiral dataset again.

On Wednesday we ignored the regularization parameters. Today, we are going to evaluate the role this parameter can play. The input data – or features - will be the two coordinates X_1 and X_2. The final map will not be displayed in "Discrete Output Mode" in order to better understand how the output of the neural network varies.

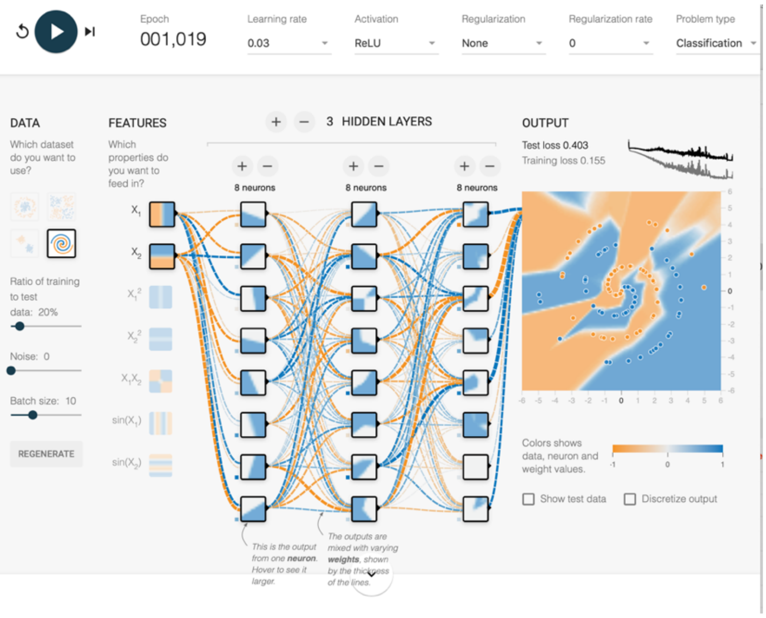
We will keep the following hyper-parameters fixed:

- **Number of hidden layers (3)**
- **Number of neurons per layer (8)**
- **There is a bias term in each neuron calculation.**
- **Learning rate: 0.03**
- **ReLU activation**

1. **How many neural networks parameters are fitted with these hyperparameters?**
   The total number of parameters (there are bias terms) is:
   (2+1)x8 + (8+1)x8 + (8+1)x8 + (8+1)x1 = 177

2. **If we take the ratio of training to test equal to 20%, how many training data points do we have? Do you expect over-fitting?**
   Since we have a total of 500 points, the training set is 20% of this, that is 100 points. Considering that we have almost twice more parameters than the number of data points, we expect over-fitting.

3. **Try one run to see what happens with the 20% ratio of training data if we calculate the neural network with no regularization. What do you observe?**
   We observe much chaotic behaviour, related to over-fitting (see image below). Over-fitting is confirmed by the big difference between the training and the test error.



Epoch 001,019

Learning rate 0.03

Activation ReLU

Regularization None

Regularization rate 0

Problem type Classification

Test loss 0.403

Training loss 0.155

Ratio of training to test data: 20%

Noise: 0

Batch size: 10

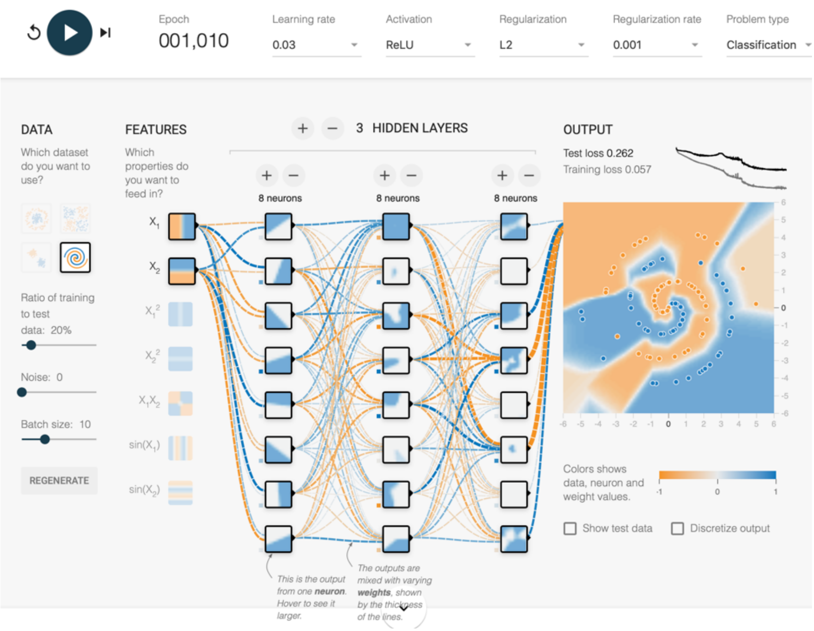
DATA
Which dataset do you want to use?
REGENERATE
FEATURES
Which properties do you want to feed in?
X_1
X_2
X_1^2
X_2^2
X_1X_2
$\sin(X_1)$
$\sin(X_2)$
+ − 3 HIDDEN LAYERS
8 neurons
8 neurons
8 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Colors shows data, neuron and weight values.
-1
0
1
□ Show test data
□ Discretize output

4. **Now we are going to run the program with different regularization parameters, and using L1 and L2 regularization. Write the mathematical expression of the loss function for the two configurations L1 and L2.**
The mathematical expression is as given during the presentation this morning:

$$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right) \text{ if L2 norm is used}$$

$$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}|\theta_i|\right) \text{ if L1 norm is used}$$

5. **Recalculate the network for L2 optimization for values of the regularization parameter equal to 0.001, 0.003, 0.01 and 0.03. What do you observe in each case?**
We observe that the rather chaotic behaviour obtained when there is no regularization is initially improved for values of the regularization parameter equal to 0.001 and 0.003. Surprisingly, the training error decreases for these two values as compared to the no regularization case, but as expected the test error decreases for values equal to 0.001 and 0.003. We observe also that the weight decay term becomes too strong as soon as the regularization parameter reaches values of 0.01 and 0.03, as it is clear from the hidden layers activations that the parameters do not vary enough to accommodate the data. We are in a situation of Bias.



Epoch 001,010

Learning rate 0.03

Activation ReLU

Regularization L2

Regularization rate 0.001

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 20%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

3 HIDDEN LAYERS

OUTPUT

Test loss 0.262

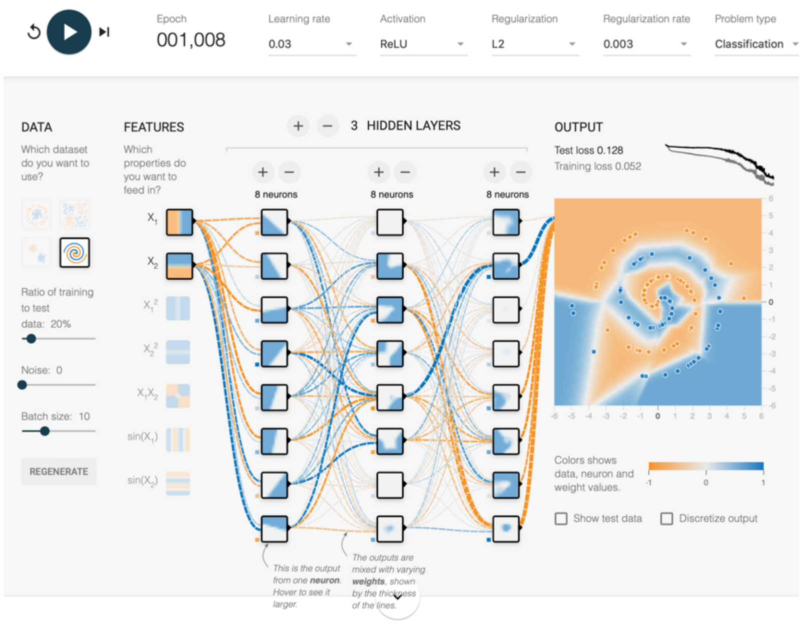
Training loss 0.057

Colors shows data, neuron and weight values.

□ Show test data □ Discretize output

*This is the output from one **neuron**. Hover to see it larger.*

*The outputs are mixed with varying **weights**, shown by the thickness of the lines.*



Epoch 001,008

Learning rate 0.03

Activation ReLU

Regularization L2

Regularization rate 0.003

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 20%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

3 HIDDEN LAYERS

OUTPUT

Test loss 0.128

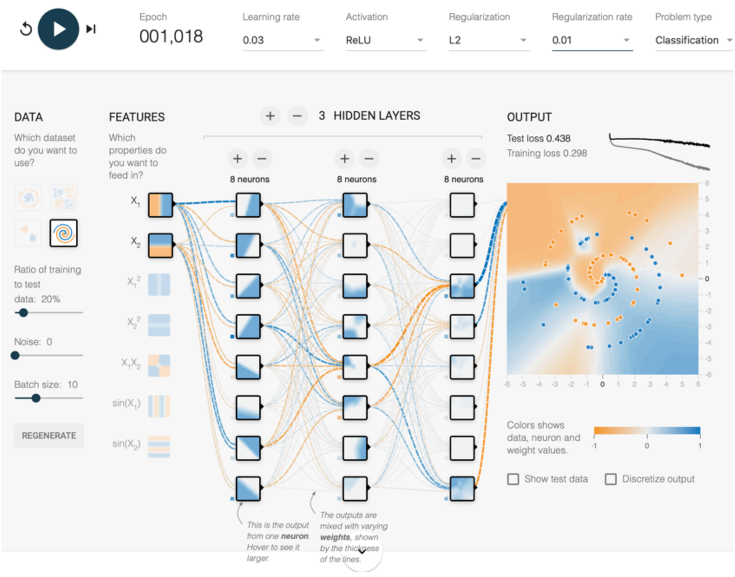
Training loss 0.052

Colors shows data, neuron and weight values.

□ Show test data □ Discretize output

*This is the output from one **neuron**. Hover to see it larger.*

*The outputs are mixed with varying **weights**, shown by the thickness of the lines.*



Epoch 001,018

Learning rate 0.03

Activation ReLU

Regularization L2

Regularization rate 0.01

Problem type Classification

Test loss 0.438

Training loss 0.298

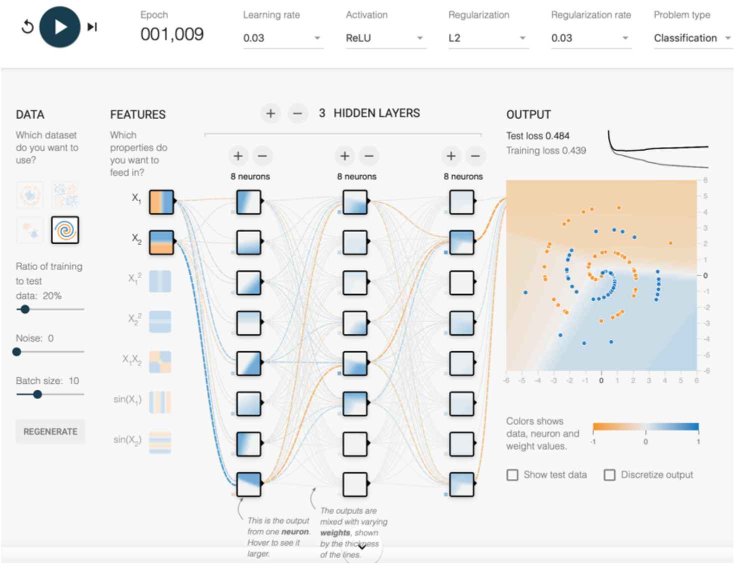
Ratio of training to test data: 20%

Noise: 0

Batch size: 10

□ Show test data □ Discretize output

DATA
Which dataset do you want to use?
REGENERATE
FEATURES
Which properties do you want to feed in?
X_1
X_2
X_1^2
X_2^2
X_1X_2
$\sin(X_1)$
$\sin(X_2)$
+ − 3 HIDDEN LAYERS
8 neurons
8 neurons
8 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Colors shows data, neuron and weight values.
-1
0
1



Epoch 001,009

Learning rate 0.03

Activation ReLU

Regularization L2

Regularization rate 0.03

Problem type Classification

Test loss 0.484

Training loss 0.439

Ratio of training to test data: 20%

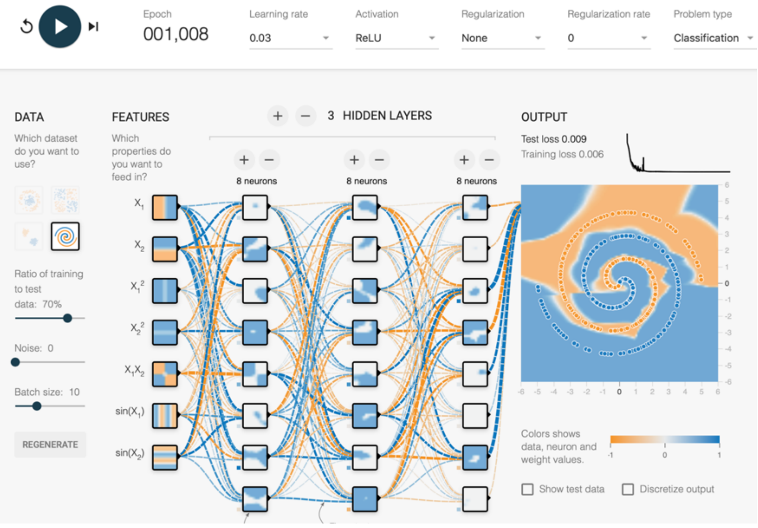
Noise: 0

Batch size: 10

□ Show test data □ Discretize output

DATA
Which dataset do you want to use?
REGENERATE
FEATURES
Which properties do you want to feed in?
X_1
X_2
X_1^2
X_2^2
X_1X_2
$\sin(X_1)$
$\sin(X_2)$
+ − 3 HIDDEN LAYERS
8 neurons
8 neurons
8 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Colors shows data, neuron and weight values.
-1
0
1

6. **Now repeat the operation with 7 instead of two input data (or features) : $X_1, X_2, X_1^2, X_2^2, X_1X_2, \sin X_1, \sin X_2$, keeping all the hyperparameters unchanged. First calculate the network without regularization, then calculate it with a 0.03 regularization coefficient. What do you observe say after 1000 epochs?**
The two images below show what happens. The first one is obtained without regularization. The image nicely matches the data but is quite rough. The second one is obtained with a L2 regularization parameter equal to 0.03. The image is much smoother, as expected. The difference is more conclusive than with just two input features, because (as seen on Wednesday) with seven input features we have more overfitting than with just two input features.



Epoch 001,008

Learning rate 0.03

Activation ReLU

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 70%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

3 HIDDEN LAYERS

8 neurons

8 neurons

8 neurons

OUTPUT

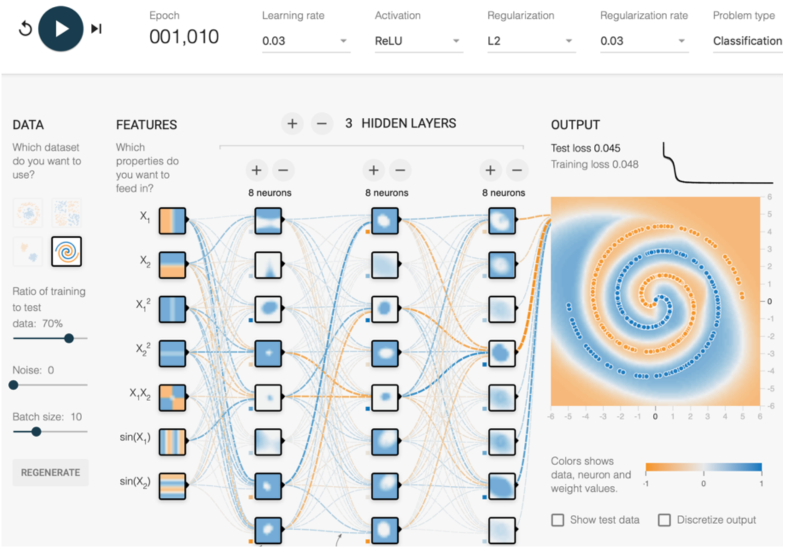
Test loss 0.009

Training loss 0.006

Colors shows data, neuron and weight values.

□ Show test data

□ Discretize output



Epoch 001,010

Learning rate 0.03

Activation ReLU

Regularization L2

Regularization rate 0.03

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 70%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

3 HIDDEN LAYERS

8 neurons

8 neurons

8 neurons

OUTPUT

Test loss 0.045

Training loss 0.048

Colors shows data, neuron and weight values.

□ Show test data

□ Discretize output

Exercise 2

The article by (Rashka et al., 2018) discusses a number of approaches for optimizing the choice of hyperparameters. It is recommended that you read the whole paper but skip Paragraphs 1.7 (Confidence Intervals via Normal Approximations) and 2.4 (The Bootstrap Method and Empirical Confidence Intervals).

Questions about the text:

1. **Please explain what is stratified sampling - or stratification - and why it is important.**
   When a dataset is partitioned into a training set and a test (or validation) set, and when this is performed randomly, it may happen that the proportion of each class is quite different in the test (or validation) set as compared to the training set. Sometimes the training set and test set can be unbalanced in opposite directions. Stratification is an approach which produces a test set with the same proportion of data in each class as in the training set. Unbalance between datasets is less of a concern in the case of large and well balanced initial datasets.

2. **Explain what is pessimistic and optimistic bias and how it relates to the size of the training and test set.**
   Pessimistic bias: the algorithm could learn a better model if it was given more data – by splitting off a portion of the dataset for testing, we withhold valuable data for estimating the generalization performance. To address this issue we may recalculate the network using both training and test data once we have checked that the model trained only on the training set fits the test (or validation) set well.
   We absolutely do not want to train and evaluate a model on the same training dataset, since it would typically introduce a very optimistic bias due to overfitting. In other words, we cannot tell whether the model simply memorized the training data, or whether it generalizes well to new, unseen test data (this optimism bias can be characterized by as the difference between the training and test accuracy).

3. **What are the three goals of performance estimation?**

a. Estimate the generalization accuracy, that is the predictive performance of a model on future (unseen) data. This is for a fixed value of the hyper-parameters.

b. Increase the predictive performance by tweaking the hyper-parameters of the learning algorithm and selecting the best-performing hyper-parameters combination.

c. Identify not only the best hyper-parameters but more generally the deep learning algorithm - or model - that is best-suited for the problem at hand: in other words, we want to compare different algorithms and select the best performing one.

4. **Why do we need a validation set?**
   The training set cannot be used to optimize the hyper-parameters, as we need to use data that have not yet been seen. A validation set must be used. The hyper-parameters which lead to the best neural network for predicting the validation set are chosen. The results on the test set should not be used to optimize the hyper-parameters.

5. **What is the difference between holdout method, 2-fold cross-validation, and repeated holdout methods?**
   With the holdout method there is half of the data used as training set and half as test (or validation) set. With 2-fold cross-validation, we do as with the holdout method but then the two sets are swapped, training set becomes test set and test set becomes training set. With the repeated holdout method (also called Monte Carlo cross-validation), we create the test set by randomly drawing half of the data. Then we repeat again by randomly drawing half of the data. We do this a number of times and calculate the average performance of the algorithm.