In-Class Exercises: Deep Learning

Work through these during the lecture. Ask if you get stuck.

Data Science with Python – BSc Course

Exercise 1: Forward Propagation by Hand

8 min | Pen & Paper

Consider a small 2-2-1 MLP (two inputs, two hidden neurons, one output) with parameters:

$$W^{(1)} = \begin{pmatrix} 0.5 & -0.3 \\ 0.2 & 0.8 \end{pmatrix}, \qquad b^{(1)} = \begin{pmatrix} 0.1 \\ -0.1 \end{pmatrix}, \qquad W^{(2)} = \begin{pmatrix} 0.7 & -0.5 \end{pmatrix}, \qquad b^{(2)} = 0.2.$$

The hidden layer uses $\mathrm{ReLU}(z) = \max(0, z)$ and the output uses $\sigma(z) = 1/(1 + e^{-z})$. The input is $x = (1,\ 2)$.

(a) Compute the pre-activation of the hidden layer $z^{(1)} = W^{(1)}x + b^{(1)}$. Show both components.

(b) Apply ReLU componentwise to get the hidden activations $h^{(1)} = \mathrm{ReLU}(z^{(1)})$.

(c) Compute the output pre-activation $z^{(2)} = W^{(2)}h^{(1)} + b^{(2)}$.

(d) Apply the sigmoid to get the final output $y = \sigma(z^{(2)})$. Round to three decimals.

(e) If this is a binary classifier and we threshold at 0.5, what class does the network predict for this input?

Exercise 2: Activation Function Match

5 min | Conceptual

Match each scenario to the *most appropriate* activation function. Write your reasoning in one sentence.

	Scenario	Best activation?
A	Deep network, you need fast training and want to avoid the vanishing-gradient issues of sigmoid/tanh.	
B	Final layer of a 10-class image classifier where the outputs should be probabilities summing to 1.	
C	ReLU network suffering from "dying neurons" — many activations stuck at zero and never recovering.	
D	Final layer of a binary (yes/no) classifier where you want a single output between 0 and 1.	

Activations to choose from: ReLU, Leaky ReLU / ELU, sigmoid, softmax, tanh.

Bonus: Why is softmax not used in *hidden* layers?

Exercise 3: Backprop Gradient for One Weight

7 min | Pen & Paper

Consider a tiny 1-1-1 network (one input, one hidden neuron with sigmoid activation, one linear output):

$$z_1 = w_1 x, \qquad h = \sigma(z_1) = \frac{1}{1+e^{-z_1}}, \qquad y = w_2 h, \qquad L = (y-t)^2.$$

Given values $x = 1$, $w_1 = 0.5$, $w_2 = 1.5$, target $t = 2$.

Forward pass

(a) Compute z_1, $h = \sigma(z_1)$, $y = w_2 h$, and the loss $L = (y-t)^2$. Round to three decimals where needed. (Hint: $\sigma(0.5) \approx 0.622$.)

Backward pass

(b) Compute the local derivatives needed for the chain rule:

- $\dfrac{\partial L}{\partial y} = 2(y-t)$
- $\dfrac{\partial y}{\partial h} = w_2$
- $\dfrac{\partial h}{\partial z_1} = h(1-h)$ (sigmoid derivative)
- $\dfrac{\partial z_1}{\partial w_1} = x$

(c) Apply the chain rule to compute $\dfrac{\partial L}{\partial w_1}$:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial h} \cdot \frac{\partial h}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1}.$$

(d) With learning rate $\eta = 0.1$, what is the updated value of w_1 after one gradient descent step ($w_1 \leftarrow w_1 - \eta\, \partial L/\partial w_1$)?

Exercise 4: Regularization Match

5 min | Discussion

Match each symptom to the *most appropriate* regularization technique. Explain in one sentence.

	Symptom	**Best technique?**
A	Validation loss starts rising while training loss keeps decreasing.	
B	Network reaches 99.9% training accuracy but only 70% on held-out data — classic memorization.	
C	Some weights blow up to very large magnitudes, making the network unstable and sensitive to input perturbations.	
D	Training is unstable: loss oscillates, activations in deep layers drift to very different scales over time.	

Techniques to choose from: early stopping, dropout, L2 (weight decay), batch normalization, L1 (lasso-style sparsity), data augmentation.

Bonus: Can you use several of these techniques at the same time? Why might you want to?

Exercise 5: Build a Simple MLP

10 min | Python (Optional)

Complete the blanks in this `sklearn` MLP classifier applied to a small tabular dataset. The goal is to practise choosing sensible hyperparameters and reading the training output.

```python
import numpy as np
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier

# 1) Synthetic 2-class tabular data
X, y = make_classification(n_samples=2000, n_features=20,
                           n_informative=10, n_redundant=5,
                           random_state=42)

# 2) Train/validation split and scaling
X_tr, X_va, y_tr, y_va = train_test_split(X, y,
                                          test_size=0.2,
                                          random_state=42)
scaler = StandardScaler().fit(X_tr)
X_tr = scaler.transform(X_tr)
X_va = scaler.transform(X_va)

# 3) Build the MLP --- fill in the blanks
mlp = MLPClassifier(
    hidden_layer_sizes = ________,   # (1) 2 hidden layers, 64 and 32
    activation         = ________,   # (2) sensible default for deep
       nets
    solver             = ________,   # (3) adaptive optimizer
    alpha              = ________,   # (4) small L2 regularization
    max_iter           = 100,
    random_state       = 42,
    early_stopping     = True,
    validation_fraction= 0.1,
)

# 4) Fit and score
mlp.fit(X_tr, y_tr)
print("Train accuracy:", mlp.score(X_tr, y_tr).round(3))
print("Val   accuracy:", mlp.score(X_va, y_va).round(3))
print("Iterations used:", mlp.n_iter_)
```

(a) Fill in blanks (1)–(4) with sensible choices.

(b) Why do we scale the features *before* fitting the MLP? What usually happens to gradient-based training if one feature is on a scale of thousands and another on a scale of 0-1?

(c) What does `early_stopping=True` do under the hood, and which task from the worksheet does it correspond to?

Exercise 6: Dropout Effect

10 min | Python (Bonus)

Investigate how dropout influences training on a noisy dataset. We use `keras`/`tensorflow` here; equivalent PyTorch code is fine.

```python
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# 1) Noisy synthetic data
X, y = make_classification(n_samples=3000, n_features=30,
                           n_informative=6, n_redundant=10,
                           flip_y=0.10, random_state=42)
X_tr, X_va, y_tr, y_va = train_test_split(X, y,
                                          test_size=0.3,
                                          random_state=42)
scaler = StandardScaler().fit(X_tr)
X_tr = scaler.transform(X_tr); X_va = scaler.transform(X_va)

def build_mlp(dropout_rate: float) -> keras.Model:
    m = keras.Sequential([
        layers.Input(shape=(30,)),
        layers.Dense(128, activation='relu'),
        layers.Dropout(dropout_rate),
        layers.Dense(64, activation='relu'),
        layers.Dropout(dropout_rate),
        layers.Dense(1,  activation='sigmoid'),
    ])
    m.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
    return m

tf.random.set_seed(42); np.random.seed(42)
no_drop   = build_mlp(0.0).fit(X_tr, y_tr,
                               validation_data=(X_va, y_va),
                               epochs=60, batch_size=64, verbose=0)
with_drop = build_mlp(0.3).fit(X_tr, y_tr,
                               validation_data=(X_va, y_va),
                               epochs=60, batch_size=64, verbose=0)

# Plot validation loss curves
plt.figure(figsize=(7, 4))
plt.plot(no_drop.history['val_loss'],   label='dropout = 0.0')
plt.plot(with_drop.history['val_loss'], label='dropout = 0.3')
plt.xlabel('Epoch'); plt.ylabel('Validation loss')
plt.legend(); plt.title('Dropout effect on validation loss')
plt.tight_layout(); plt.show()
```

(a) Run the code. Compare the *gap* between training and validation accuracy for both models. Which one overfits more?

(b) Describe what you see in the validation-loss plot. After how many epochs does the `dropout=0.0` model start to degrade on validation data?

(c) If you increased dropout to 0.7, what would you expect to happen? Try it if time permits. (Hint: there is a sweet spot — too little does nothing, too much starves the network.)