

# Lecture 3 A Minimal Case Study

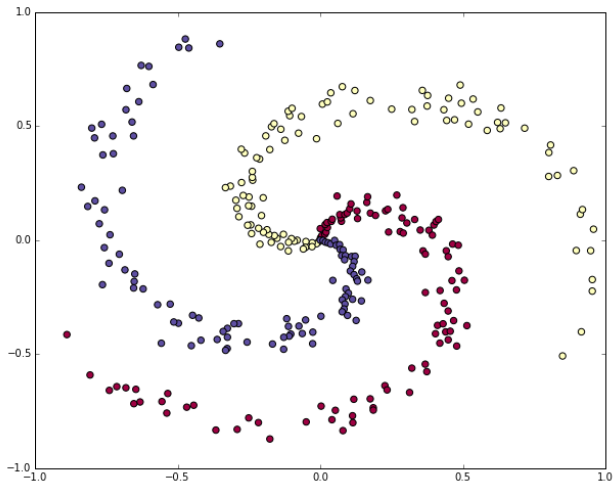
Fei Tan

Department of Economics  
Chaifetz School of Business  
Saint Louis University

E6930 Introduction to Neural Networks

June 28, 2024

# Nonlinear Classification



# The Road Ahead...

① Training Softmax Classifier

② Training Neural Network

# Throwback

## $L^2$ regularized loss

$$L = \frac{1}{N} \sum_i L_i + \frac{1}{2} \lambda \sum_{i,j} W_{ij}^2$$

- ▶ Softmax classifier

- ▶ linear score:  $f(x_i, W, b) = Wx_i + b$

- ▶ cross-entropy loss:  $L_i = -\log(p_{y_i})$ ,  $p_k = \frac{e^{f_k}}{\sum_j e^{f_j}}$

- ▶ Backpropagation

$$\frac{\partial L}{\partial W_k} = \frac{1}{N} \sum_i \underbrace{\frac{\partial L_i}{\partial f_k}}_{p_k - \mathbb{1}(y_i=k)} \underbrace{\frac{\partial f_k}{\partial W_k}}_{x'_i} + \lambda W_k.$$

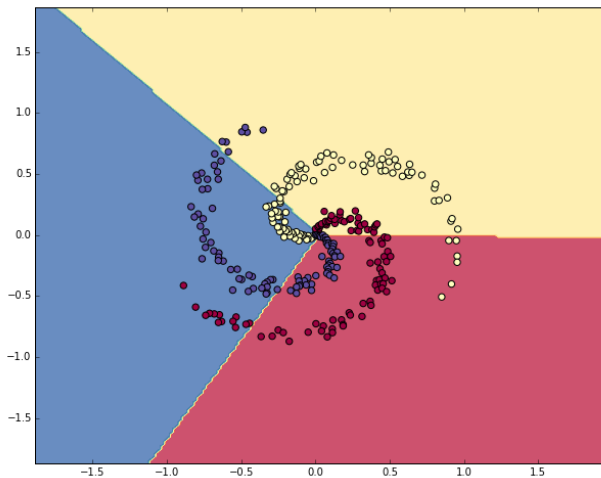
# Forward-Backward Pass

```
import numpy as np

# Forward pass
exp_scores = np.exp(np.dot(X, W) + b)
probs = exp_scores / np.sum(exp_scores, axis=1,
                             keepdims=True)
correct_logprobs = -np.log(probs[range(X.shape[0]),
                                     y])
loss = np.sum(correct_logprobs) / X.shape[0] + 0.5
       * reg * np.sum(W * W)

# Backward pass
probs[range(X.shape[0]), y] -= 1
dscores = probs / X.shape[0]
dW = np.dot(X.T, dscores)
db = np.sum(dscores, axis=0, keepdims=True)
dW += reg*W
```

# Softmax Classifier



# The Road Ahead...

① Training Softmax Classifier

② Training Neural Network

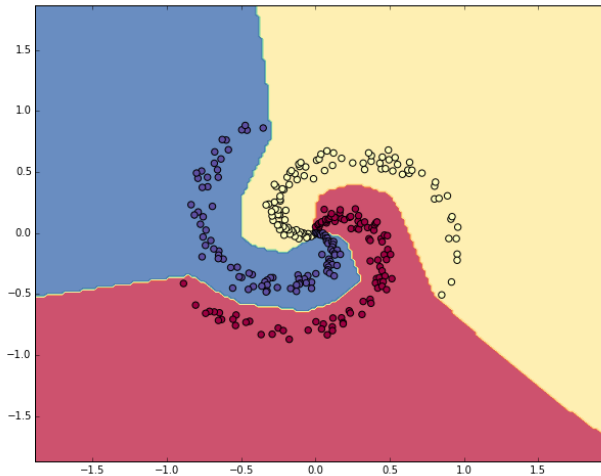
# Forward-Backward Pass

```
# Forward pass
hidden_layer = np.maximum(0, np.dot(X, W) + b)
scores = np.dot(hidden_layer, W2) + b2

# Backward pass
dW2 = np.dot(hidden_layer.T, dscores)
db2 = np.sum(dscores, axis=0, keepdims=True)
dhidden = np.dot(dscores, W2.T)
dhidden[hidden_layer <= 0] = 0 # ReLU
dW = np.dot(X.T, dhidden)
db = np.sum(dhidden, axis=0, keepdims=True)
```



# Neural Network



# PyTorch Implementation

```
import torch
import torch.nn as nn
import torch.optim as optim

X_tensor = torch.tensor(X, dtype=torch.float32)
y_tensor = torch.tensor(y, dtype=torch.long)

class SimpleNN(nn.Module):
    def __init__(self, D, h, K):
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(D, h)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(h, K)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x
```

# PyTorch Implementation (Cont'd)

```
# Initialization
model = SimpleNN(D, h, K)
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=1e-0,
                        weight_decay=1e-3)

# Training
for epoch in range(num_epochs):
    # Forward pass
    scores = model(X_tensor)
    loss = criterion(scores, y_tensor)
    optimizer.zero_grad() # zero grad

    # Backward pass
    loss.backward()
    optimizer.step()
```

# References

- ▶ [cs231n.stanford.edu](https://cs231n.stanford.edu) – CS231n: Deep Learning for Computer Vision
- ▶ [pytorch.org](https://pytorch.org) – PyTorch package for deep learning