COMP90051

Workshop Week 06

About the Workshops

- 7 sessions in total
  - Tue 12:00-13:00 AH211
  - Tue 12:00-13:00 AH108 *
  - Tue 13:00-14:00 AH210
  - Tue 16:15-17:15 AH109
  - Tue 17:15-18:15 AH236 *
  - Tue 18:15-19:15 AH236 *
  - Fri 14:15-15:15 AH211

About the Workshops

- Homepage
  - https://trevorcohn.github.io/comp90051-2017/workshops

- Solutions will be released on next Friday (a week later).

Syllabus

1	Introduction; Probability theory	Probabilistic models; Parameter fitting	
2	Linear regression; Intro to regularization	Logistic regression; Basis expansion	
3	Optimization; Regularization	Perceptron	
4	Backpropagation	CNNs; Auto-encoders	←
5	Hard-margin SVMs	Soft-margin SVMs	
6	Kernel methods	Ensemble Learning	
7	Unsupervised learning	Unsupervised learning	
8	Dimensionality reduction; Principal component analysis	Multidimensional scaling; Spectral clustering	
9	Bayesian fundamentals	Bayesian inference with conjugate priors	
10	PGMs, fundamentals	Conditional independence	
11	PGMs, inference	Belief propagation	
12	Statistical inference; Apps	Subject review	

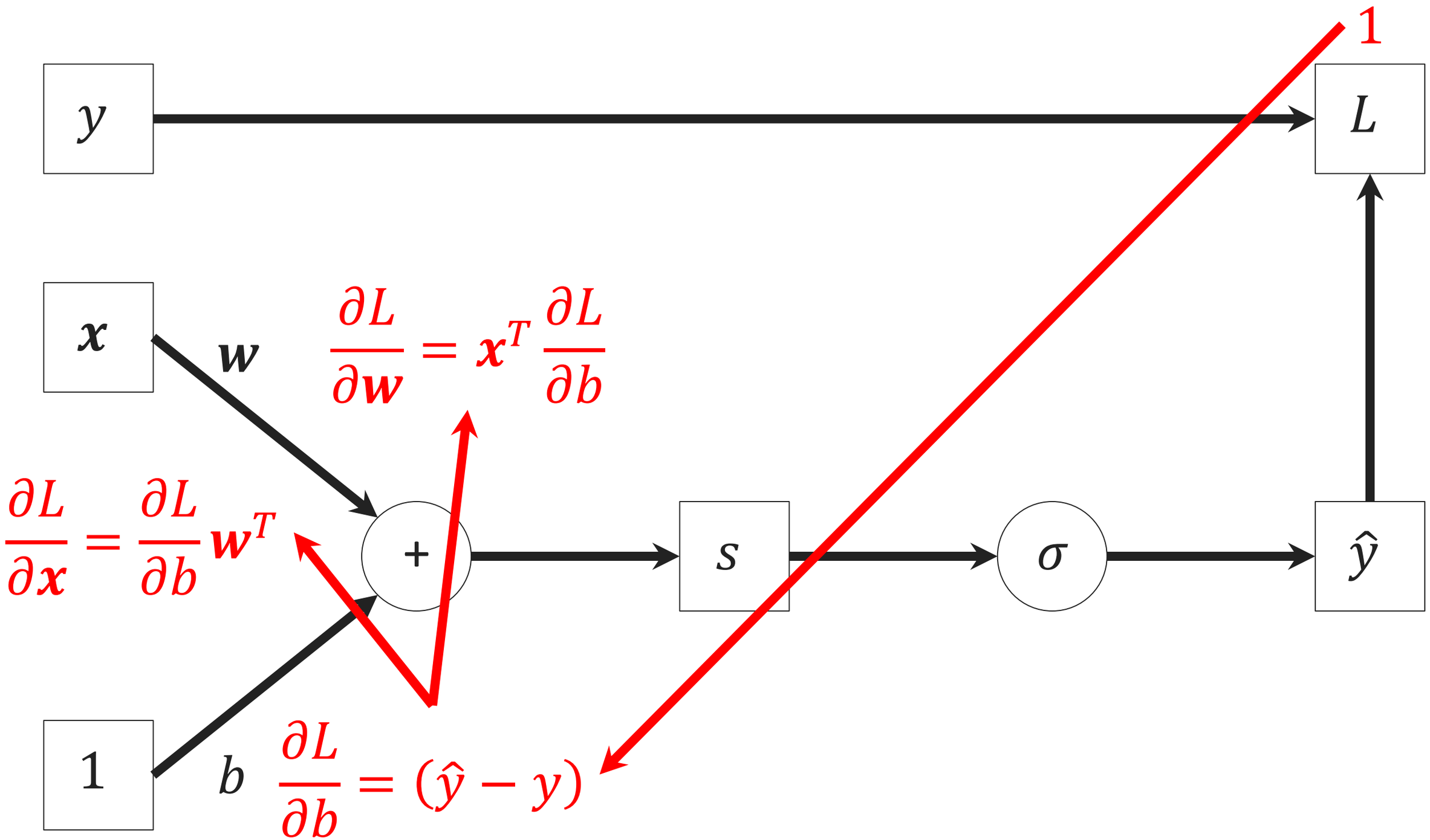
Outline

- Review the lecture, background knowledge, etc.
  - Backpropagation implementation
    - For neural networks with one hidden layer
- Work on the project, workshop materials, etc.

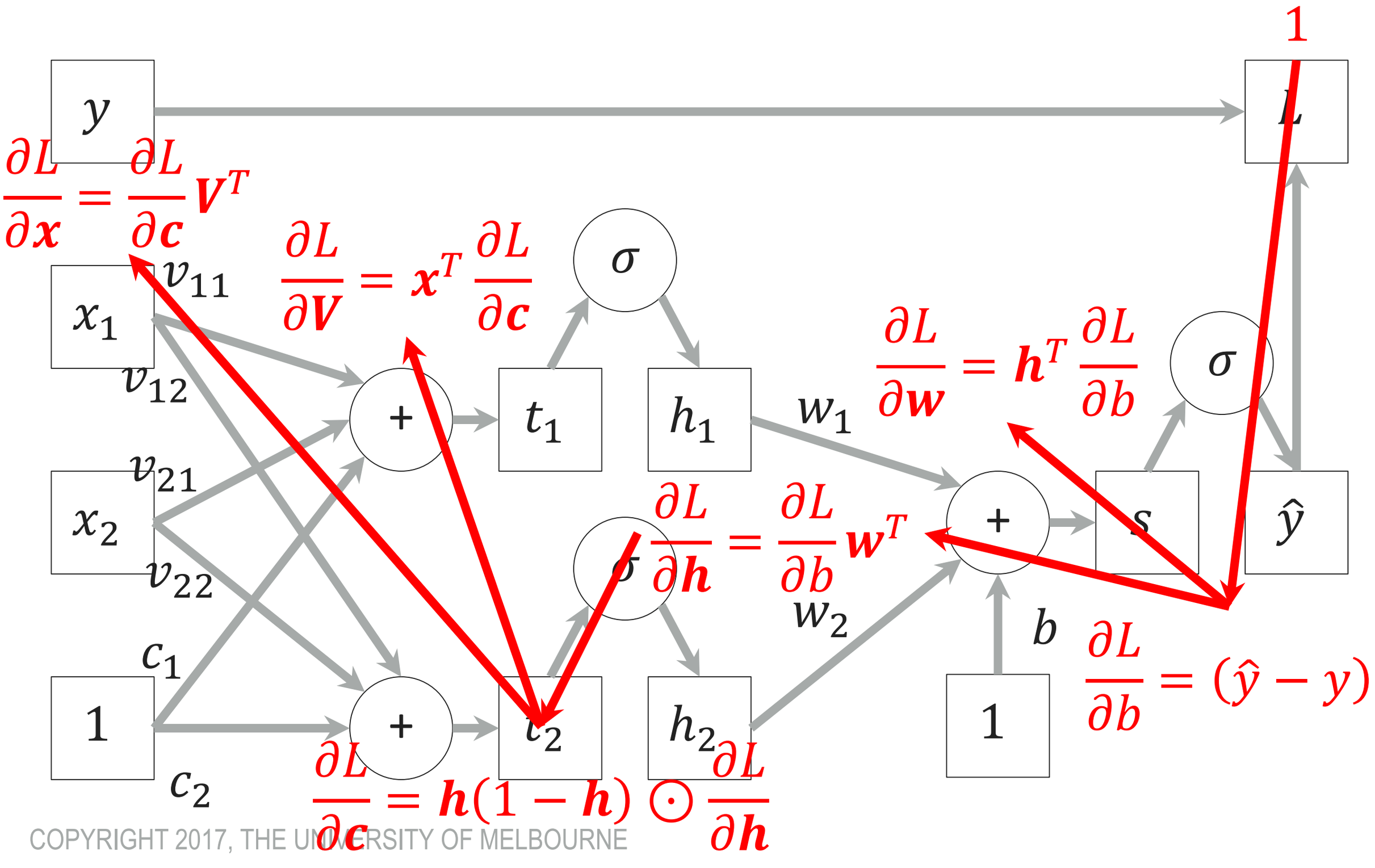
Outline

- Review the lecture, background knowledge, etc.
  - Backpropagation implementation
    - For neural networks with one hidden layer
- Work on the project, workshop materials, etc.

Backpropagation formulas



Backpropagation formulas



Neural networks with one hidden layer

- 3 layers
  - Input layer: num_feature units
  - Hidden layer: num_hidden units
  - Output layer: num_output units

- For 2-D points, binary classification
  - num_feature = 2
  - num_output = 1

Forward pass and backprop

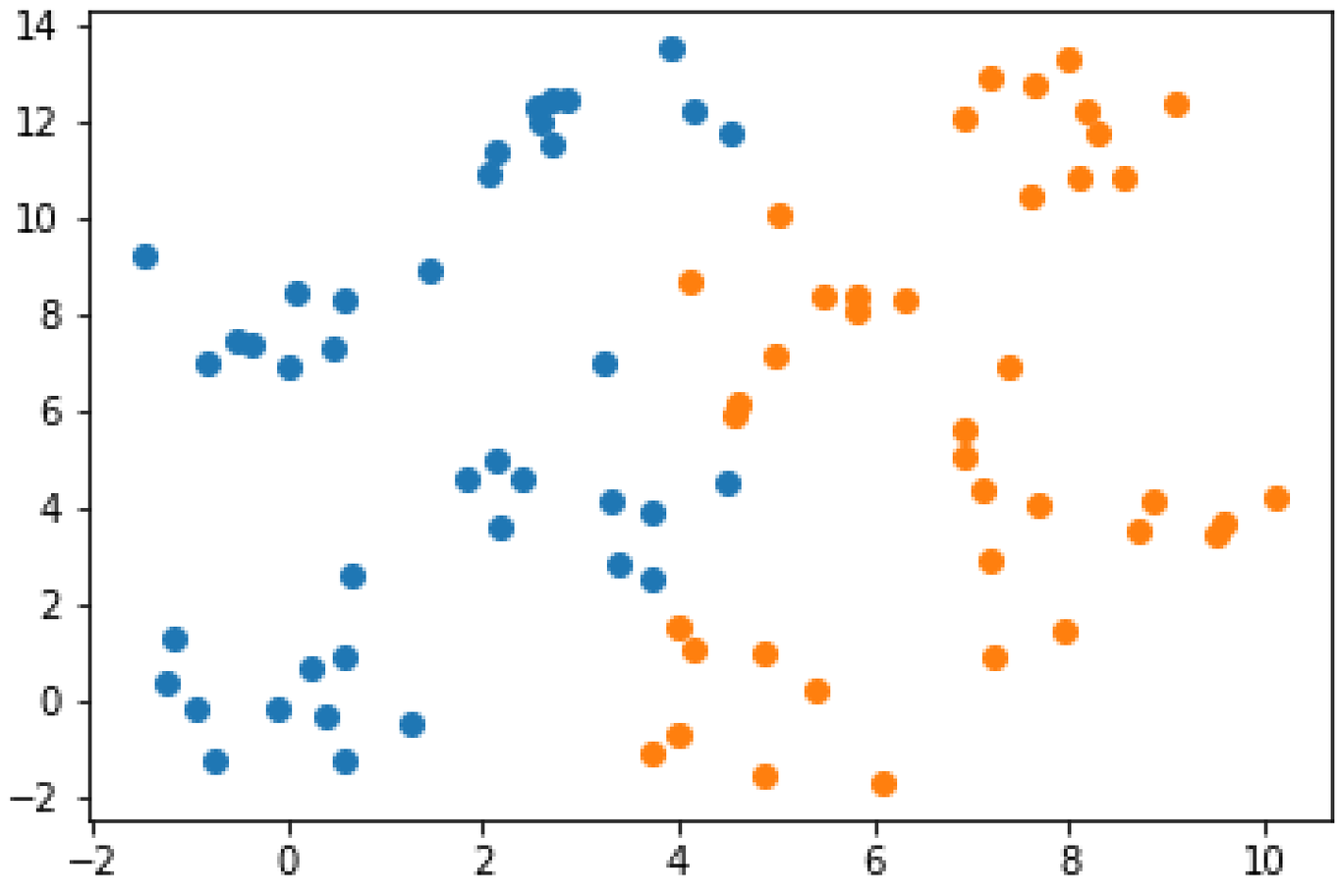
```
def forward(X, V, c, W, b):
    t = X.dot(V) + c
    h = scipy.special.expit(t)
    s = h.dot(W) + b
    y_hat = scipy.special.expit(s)

    return X, V, c, t, h, W, b, s, y_hat
```

```
def backprop(X, V, c, t, h, W, b, s, y_hat, y):
    grad_b = y_hat - y
    grad_W = h.T * grad_b
    grad_h = grad_b * W.T

    grad_c = h*(1-h) * grad_h
    grad_V = X.T.dot(grad_c)

    return grad_V, grad_c, grad_W, grad_b
```

Create a dataset

```
X, y = generate_s_shaped_data(5)
print('X:', X.shape)
print('y:', y.shape)

plt.plot(X[y==0, 0], X[y==0, 1], "o")
plt.plot(X[y==1, 0], X[y==1, 1], "o")
plt.show()
```

```
X: (80, 2)
y: (80,)
```



Initialization

```
num_points   = X.shape[0]
num_features = X.shape[1]
num_hidden = 5
num_output = 1

V = np.random.randn(num_features, num_hidden)
c = np.zeros(num_hidden)

W = np.random.randn(num_hidden, num_output)
b = np.zeros(num_output)

print('X:', X.shape)    →    X: (80, 2)
print('y:', y.shape)    →    y: (80,)
print('V:', V.shape)    →    V: (2, 5)
print('c:', c.shape)    →    c: (5,)
print('W:', W.shape)    →    W: (5, 1)
print('b:', b.shape)    →    b: (1,)
```

Training (SGD online)

```
eta = 0.1
num_epoch = 500
accuracy = 0
for k in range(num_epoch):
    for i in range(num_points):
        tmp = forward(X[i], V, c, W, b)
        grad_V, grad_c, grad_W, grad_b = backprop(*tmp, y[i])
        V -= eta * grad_V
        c -= eta * grad_c
        W -= eta * grad_W
        b -= eta * grad_b

    y_hat = forward(X, V, c, W, b)[-1].round()
    if accuracy_score(y, y_hat) > accuracy:
        accuracy = accuracy_score(y, y_hat)
        print('iter %d: accuracy %lf'%(k, accuracy))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-15-17b779b713c9> in <module>()
      5     for i in range(num_points):
      6         tmp = forward(X[i], V, c, W, b)
----> 7         grad_V, grad_c, grad_W, grad_b = backprop(*tmp, y[i])
      8
      9         V -= eta * grad_V

<ipython-input-14-15446808e179> in backprop(X, V, c, t, h, W, b, s, y_hat, y)
      5
      6     grad_c = h*(1-h) * grad_h
----> 7     grad_V = X.T.dot(grad_c)
      8
      9     return grad_V, grad_c, grad_W, grad_b

ValueError: shapes (2,) and (1,5) not aligned: 2 (dim 0) != 1 (dim 0)
```

- We expect X to be a row vector of shape (1, 2)
- So that X.T is a col vector of shape (2, 1)
- However, X.T now has shape (2,)

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-15-17b779b713c9> in <module>()
      5     for i in range(num_points):
      6         tmp = forward(X[i], V, c, W, b)
----> 7         grad_V, grad_c, grad_W, grad_b = backprop(*tmp, y[i])
      8
      9         V -= eta * grad_V

<ipython-input-14-15446808e179> in backprop(X, V, c, t, h, W, b, s, y_hat, y)
      5
      6     grad_c = h*(1-h) * grad_h
----> 7     grad_V = X.T.dot(grad_c)
      8
      9     return grad_V, grad_c, grad_W, grad_b

ValueError: shapes (2,) and (1,5) not aligned: 2 (dim 0) != 1 (dim 0)
```

The shape of X[i] is (2,) not (1, 2)

```
eta = 0.1
num_epoch = 500
accuracy = 0
for k in range(num_epoch):
    for i in range(num_points):
        tmp = forward(X[i], V, c, W, b)
        grad_V, grad_c, grad_W, grad_b = backprop(*tmp, y[i])
        V -= eta * grad_V
        c -= eta * grad_c
        W -= eta * grad_W
        b -= eta * grad_b

    y_hat = forward(X, V, c, W, b)[-1].round()
    if accuracy_score(y, y_hat) > accuracy:
        accuracy = accuracy_score(y, y_hat)
        print('iter %d: accuracy %lf'%(k, accuracy))
```

```
A = np.arange(6).reshape(2, 3)
print('A:')
print(A)
print('A[0]       \t', A[0].shape, A[0])
print('A[0, :]    \t', A[0, :].shape, A[0, :])
print('A[0, None]\t', A[0, None].shape, A[0, None])
print('A[[0]]     \t', A[[0]].shape, A[[0]])
```

```
----------------------------------------------------------------------

A:
[[0 1 2]
 [3 4 5]]
A[0]          (3,) [0 1 2]
A[0, :]       (3,) [0 1 2]
A[0, None]    (1, 3) [[0 1 2]]
A[[0]]        (1, 3) [[0 1 2]]
```

Training (SGD online)

```
eta = 0.1
num_epoch = 500
accuracy = 0
for k in range(num_epoch):
    for i in range(num_points):
        tmp = forward(X[i, None], V, c, W, b)
        grad_V, grad_c, grad_W, grad_b = backprop(*tmp, y[i])
        V -= eta * grad_V
        c -= eta * grad_c
        W -= eta * grad_W
        b -= eta * grad_b

    y_hat = forward(X, V, c, W, b)[-1].round()
    if accuracy_score(y, y_hat) > accuracy:
        accuracy = accuracy_score(y, y_hat)
        print('iter %d: accuracy %lf'%(k, accuracy))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-32-9f7c848f98dc> in <module>()
      8
      9         V -= eta * grad_V
---> 10         c -= eta * grad_c
     11         W -= eta * grad_W
     12         b -= eta * grad_b

ValueError: non-broadcastable output operand with shape (5,) doesn't match the broadcast shape (1,5)
```

Initialization

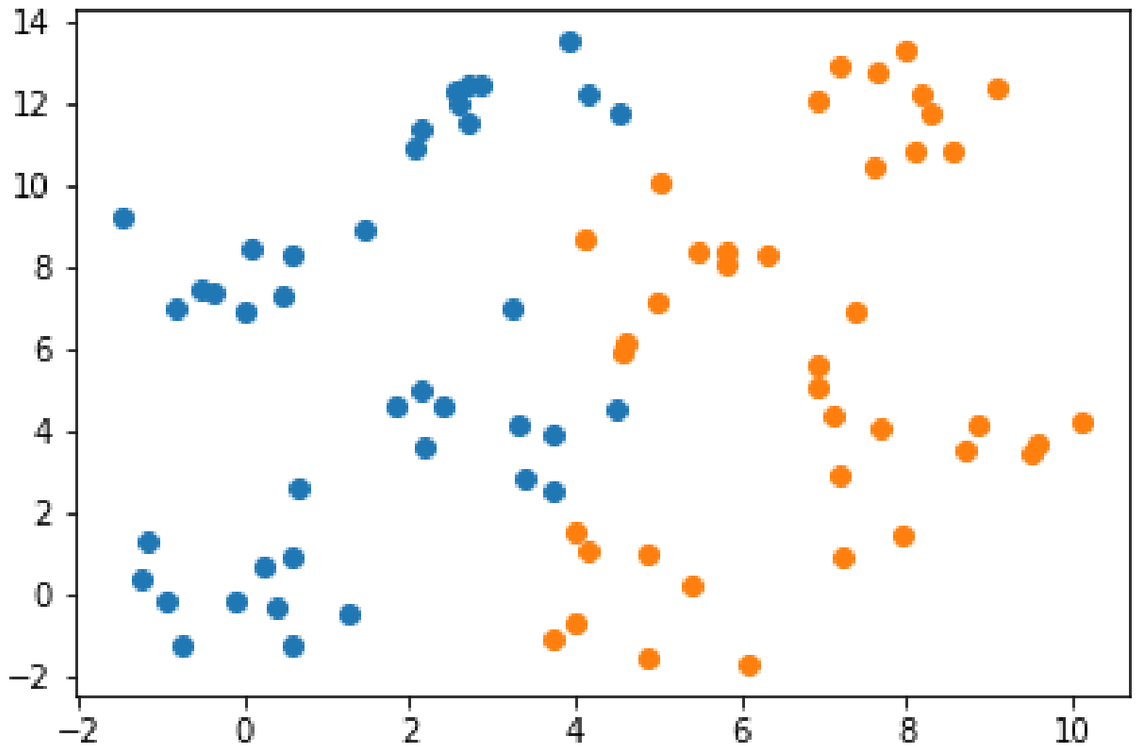
```
num_points   = X.shape[0]
num_features = X.shape[1]
num_hidden = 5
num_output = 1

V = np.random.randn(num_features, num_hidden)
c = np.zeros(num_hidden)

W = np.random.randn(num_hidden, num_output)
b = np.zeros(num_output)

print('X:', X.shape)    →    X: (80, 2)
print('y:', y.shape)    →    y: (80,)
print('V:', V.shape)    →    V: (2, 5)
print('c:', c.shape)    →    c: (5,)
print('W:', W.shape)    →    W: (5, 1)
print('b:', b.shape)    →    b: (1,)
```

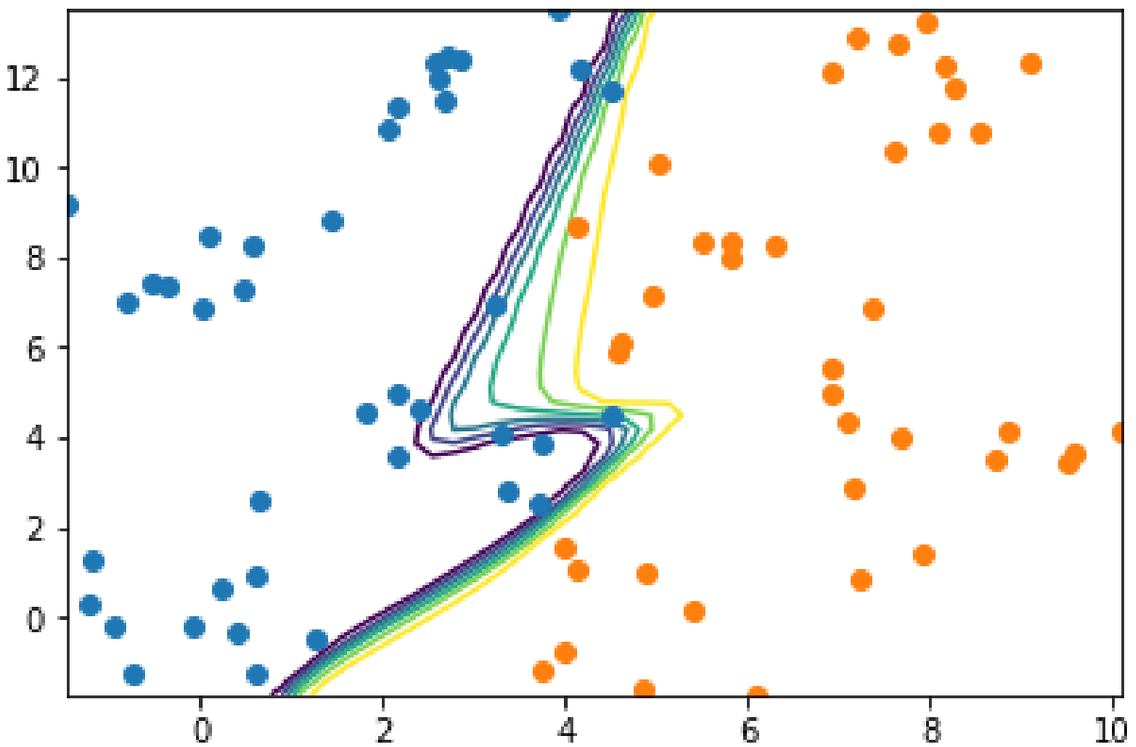
Explicitly define as row vectors

```
num_points   = X.shape[0]
num_features = X.shape[1]
num_hidden = 5
num_output = 1

V = np.random.randn(num_features, num_hidden)
c = np.zeros((1, num_hidden))

W = np.random.randn(num_hidden, num_output)
b = np.zeros((1, num_output))

print('X:', X.shape)    →    X: (80, 2)
print('y:', y.shape)    →    y: (80,)
print('V:', V.shape)    →    V: (2, 5)
print('c:', c.shape)    →    c: (5,)
print('W:', W.shape)    →    W: (5, 1)
print('b:', b.shape)    →    b: (1,)
```

Output looks like this…

```
iter 0: accuracy 0.500000
iter 1: accuracy 0.587500
iter 2: accuracy 0.662500
iter 3: accuracy 0.750000
iter 4: accuracy 0.812500
iter 5: accuracy 0.825000
iter 10: accuracy 0.837500
iter 14: accuracy 0.850000
iter 25: accuracy 0.862500
iter 34: accuracy 0.875000
iter 44: accuracy 0.887500
iter 99: accuracy 0.900000
iter 228: accuracy 0.912500
iter 229: accuracy 0.925000
iter 234: accuracy 0.937500
iter 300: accuracy 0.950000
iter 306: accuracy 0.962500
iter 367: accuracy 0.975000
iter 432: accuracy 0.987500
```





Other preprocessing steps

- Normalize data
  - `X = (X-X.mean(axis=0)) / X.std(axis=0)`
- Shuffle data
  - `from sklearn.utils import shuffle`
  - `X, y = shuffle(X, y)`

Outline

- Review the lecture, background knowledge, etc.
  - Backpropagation implementation
    - For neural networks with one hidden layer

- Work on the project, workshop materials, etc.