Deep Learning

Or why you should just ask a computer to figure it out.

Tensors

Scalars – a single value

Vectors – A one-dimensional array of values

Matrices – A two-dimensional array of values

Tensors

Scalars – a single value

Vectors – A one-dimensional array of values

Matrices – A two-dimensional array of values

Tensors can have more than two dimensions (A hierarchical arrangement of matrices and vectors).
E.g. An Image.

Tensors

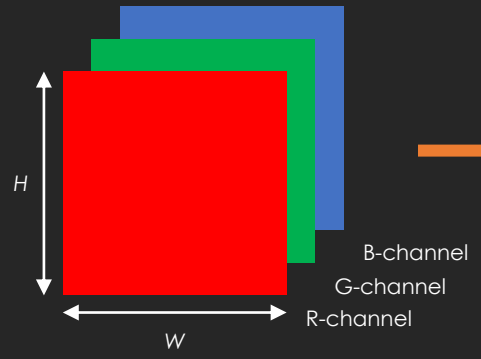
Scalars – a single value

Vectors – A one-dimensional array of values

Matrices – A two-dimensional array of values

Tensors can have more than three dimensions (A hierarchical arrangement of matrices and vectors). E.g. An Image.

Images



3 Dimensions
(H,W, {R/G/B})

Tensor shape
*3*H*W*
(or)
Torch.Size(3,H,W)
in PyTorch

Tensors

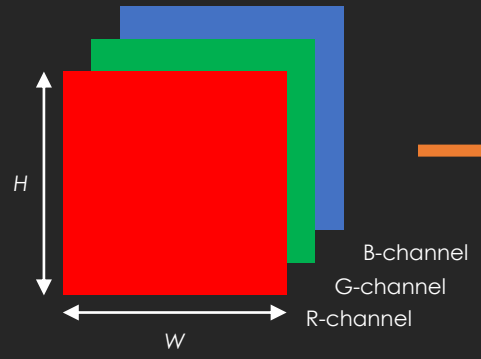
Scalars – a single value

Vectors – A one-dimensional array of values

Matrices – A two-dimensional array of values

Tensors can have more than two dimensions (A hierarchical arrangement of matrices and vectors).
E.g. An Image.

Images



3 Dimensions
(H,W, {R/G/B})

Tensor shape
*3*H*W*
(or)
Torch.Size(3,H,W)
in PyTorch

When is the image wrong?

Tensors Operations

Mathematical (including Boolean)

Join-Based

Indexing

Conversion

Task: Go look for useful tensor methods/functions and keywords and populate a shared google sheet with them. (Have one tab for pytorch and one for tensorflow)

Tensor Broadcasting

Broadcasting is an automatic tiling mechanism to ensure Tensor shapes match for operations to succeed.

Addition

$$\begin{bmatrix}[3],\\ [2],\\ [1]]\end{bmatrix} + [[0,1]]$$

3x1 *1x2*

https://pytorch.org/docs/stable/notes/broadcasting.html

Tensor Broadcasting

Broadcasting is an automatic tiling mechanism to ensure Tensor shapes match for operations to succeed.

Addition

```
[[3],
 [2],   +  [[0,1]]
 [1]]
```

3x1 *1x2*

```
=  [[3, 3],        [[0, 1],
   [2, 2],    +    [0, 1],
   [1, 1]]         [0, 1]]
```

(Broadcast tensors)

https://pytorch.org/docs/stable/notes/broadcasting.html

Tensor Broadcasting

Broadcasting is an automatic tiling mechanism to ensure Tensor shapes match for operations to succeed.

Addition

```
[[3],
 [2],   +  [[0,1]]
 [1]]
```

3x1 *1x2*

```
=  [[3, 3],        [[0, 1],
   [2, 2],    +    [0, 1],     (Broadcast tensors)
   [1, 1]]         [0, 1]]

=  [[3, 4],
   [2, 3],
   [1, 2]]
```

https://pytorch.org/docs/stable/notes/broadcasting.html

Tensor Broadcasting

Broadcasting is an automatic tiling mechanism to ensure Tensor shapes match for operations to succeed.

Addition

```
[[3],
 [2],   +  [[0,1]]
 [1]]
```

3x1 *1x2*

```
=  [[3, 3],        [[0, 1],
   [2, 2],    +    [0, 1],     (Broadcast tensors)
   [1, 1]]         [0, 1]]

=  [[3, 4],
   [2, 3],
   [1, 2]]
```

Metarule

1. For any given dimension, only one tensor can be tiled.
2. Size checks happen from right to left

https://pytorch.org/docs/stable/notes/broadcasting.html

Tensor Broadcasting

Broadcasting is an automatic tiling mechanism to ensure Tensor shapes match for operations to succeed.

Addition

```
[[3],
 [2],   +  [[0,1]]
 [1]]
```

3x1 *1x2*

```
=  [[3, 3],        [[0, 1],
   [2, 2],    +    [0, 1],     (Broadcast tensors)
   [1, 1]]         [0, 1]]

=  [[3, 4],
   [2, 3],
   [1, 2]]
```

Metarule

1. For any given dimension, only one tensor can be tiled.
2. Size checks happen from right to left

1) Can torch.Size(5,1,3) torch.Size(5,1,3) be broadcast?

https://pytorch.org/docs/stable/notes/broadcasting.html

Tensor Broadcasting

Broadcasting is an automatic tiling mechanism to ensure Tensor shapes match for operations to succeed.

Addition

```
[[3],
 [2],   +  [[0,1]]
 [1]]
```

3x1 *1x2*

```
=  [[3, 3],        [[0, 1],
   [2, 2],    +    [0, 1],     (Broadcast tensors)
   [1, 1]]         [0, 1]]

=  [[3, 4],
   [2, 3],
   [1, 2]]
```

Metarule

1. For any given dimension, only one tensor can be tiled.
2. Size checks happen from right to left

1) Can torch.Size(5,1,3)
   torch.Size(5,1,3) be broadcast?
2) Can torch.Size(5,2,3,1)
   torch.Size(5,1,3,1) be broadcast?

https://pytorch.org/docs/stable/notes/broadcasting.html

Tensor Broadcasting

Broadcasting is an automatic tiling mechanism to ensure Tensor shapes match for operations to succeed.

Addition

$$\begin{matrix}[[3], \\ [2], \\ [1]]\end{matrix} + [[0,1]]$$

3x1 *1x2*

$$= \begin{matrix}[[3, 3], \\ [2, 2], \\ [1, 1]]\end{matrix} + \begin{matrix}[[0, 1], \\ [0, 1], \\ [0, 1]]\end{matrix}$$

(Broadcast tensors)

$$= \begin{matrix}[[3, 4], \\ [2, 3], \\ [1, 2]]\end{matrix}$$

Metarule

1. For any given dimension, only one tensor can be tiled.
2. Size checks happen from right to left

1) Can torch.Size(5,1,3) torch.Size(5,1,3) be broadcast?
2) Can torch.Size(5,2,3,1) torch.Size(5,1,3,1) be broadcast?
3) Can torch.Size(5,2,3,1) torch.Size(5,1,3,5) be broadcast?

https://pytorch.org/docs/stable/notes/broadcasting.html

Tensor Broadcasting

Broadcasting is an automatic tiling mechanism to ensure Tensor shapes match for operations to succeed.

Addition

$$\begin{matrix}[[3], \\ [2], \\ [1]]\end{matrix} + [[0,1]]$$

3x1 *1x2*

$$= \begin{matrix}[[3, 3], \\ [2, 2], \\ [1, 1]]\end{matrix} + \begin{matrix}[[0, 1], \\ [0, 1], \\ [0, 1]]\end{matrix}$$ (Broadcast tensors)

$$= \begin{matrix}[[3, 4], \\ [2, 3], \\ [1, 2]]\end{matrix}$$

Metarule

1. For any given dimension, only one tensor can be tiled.
2. Size checks happen from right to left

1) Can torch.Size(5,1,3) torch.Size(5,1,3) be broadcast?
2) Can torch.Size(5,2,3,1) torch.Size(5,1,3,1) be broadcast?
3) Can torch.Size(5,2,3,1) torch.Size(5,1,3,5) be broadcast?
4) Can torch.Size(5,3,3) torch.Size(5,2,3) be broadcast?

https://pytorch.org/docs/stable/notes/broadcasting.html

Tensor Broadcasting

Broadcasting is an automatic tiling mechanism to ensure Tensor shapes match for operations to succeed.

Addition

$$\begin{matrix}[[3], \\ [2], \\ [1]]\end{matrix} + [[0,1]]$$

3x1 *1x2*

$$= \begin{matrix}[[3, 3], \\ [2, 2], \\ [1, 1]]\end{matrix} + \begin{matrix}[[0, 1], \\ [0, 1], \\ [0, 1]]\end{matrix}$$ (Broadcast tensors)

$$= \begin{matrix}[[3, 4], \\ [2, 3], \\ [1, 2]]\end{matrix}$$

Metarule

1. For any given dimension, only one tensor can be tiled.
2. Size checks happen from right to left

1) Can torch.Size(5,1,3) torch.Size(5,1,3) be broadcast?
2) Can torch.Size(5,2,3,1) torch.Size(5,1,3,1) be broadcast?
3) Can torch.Size(5,2,3,1) torch.Size(5,1,3,5) be broadcast?
4) Can torch.Size(5,3,3) torch.Size(5,2,3) be broadcast?
5) Can torch.Size(5,3,3) torch.Size(5,3) be broadcast?

https://pytorch.org/docs/stable/notes/broadcasting.html

Tensor Broadcasting

Broadcasting is an automatic tiling mechanism to ensure Tensor shapes match for operations to succeed.

Addition

$$\begin{bmatrix}[3],\\ [2],\\ [1]]\end{bmatrix} + [[0,1]]$$

3x1 *1x2*

$$= \begin{bmatrix}[3, 3],\\ [2, 2],\\ [1, 1]]\end{bmatrix} + \begin{bmatrix}[0, 1],\\ [0, 1],\\ [0, 1]]\end{bmatrix}$$ (Broadcast tensors)

$$= \begin{bmatrix}[3, 4],\\ [2, 3],\\ [1, 2]]\end{bmatrix}$$

Metarule

1. For any given dimension, only one tensor can be tiled.
2. Size checks happen from right to left

1) Can torch.Size(5,1,3) torch.Size(5,1,3) be broadcast?
2) Can torch.Size(5,2,3,1) torch.Size(5,1,3,1) be broadcast?
3) Can torch.Size(5,2,3,1) torch.Size(5,1,3,5) be broadcast?
4) Can torch.Size(5,3,3) torch.Size(5,2,3) be broadcast?
5) Can torch.Size(5,3,3) torch.Size(5,3) be broadcast?
6) Can torch.Size(5,3,3) torch.Size(3,3) be broadcast?

https://pytorch.org/docs/stable/notes/broadcasting.html

Differential Calculus

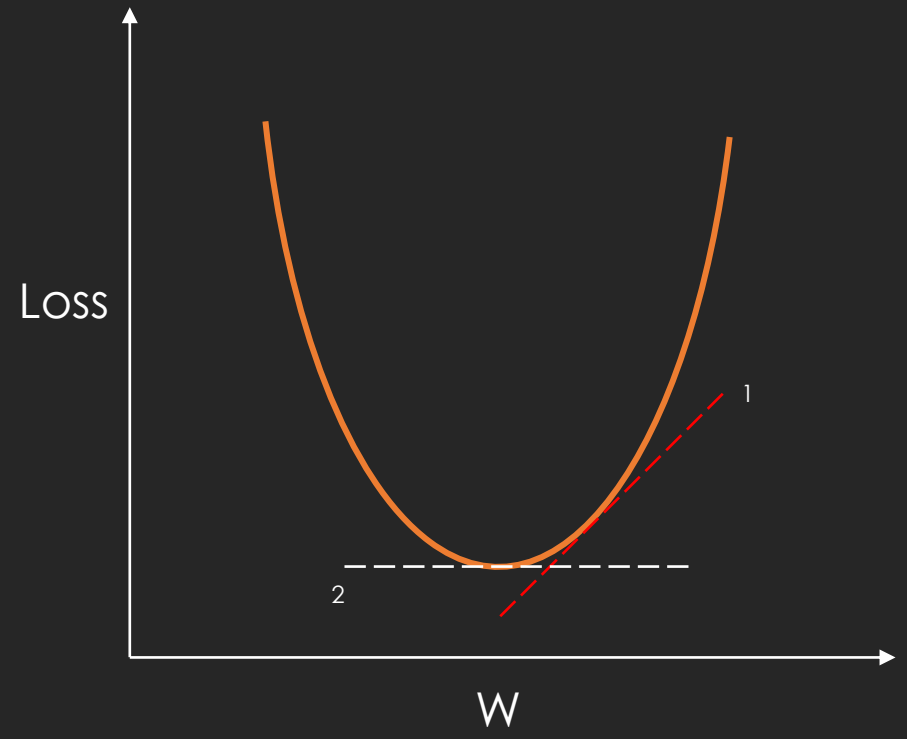


Image credit: https://losslandscape.com/

Automatic Differentiation

Calculating the gradients is difficult and cumbersome become of interactions in very large models. So, our modern frameworks use automatic differentiation.

With automatic differentiation,

1. the framework builds a graph that connects all the inputs and operations that produce a particular output
2. The nodes and edges in the graph capture input/output relationships automatically
3. Tracing through the graph allows us to calculate gradients automatically.

Reference: https://en.wikipedia.org/wiki/Automatic_differentiation

Probability

References:

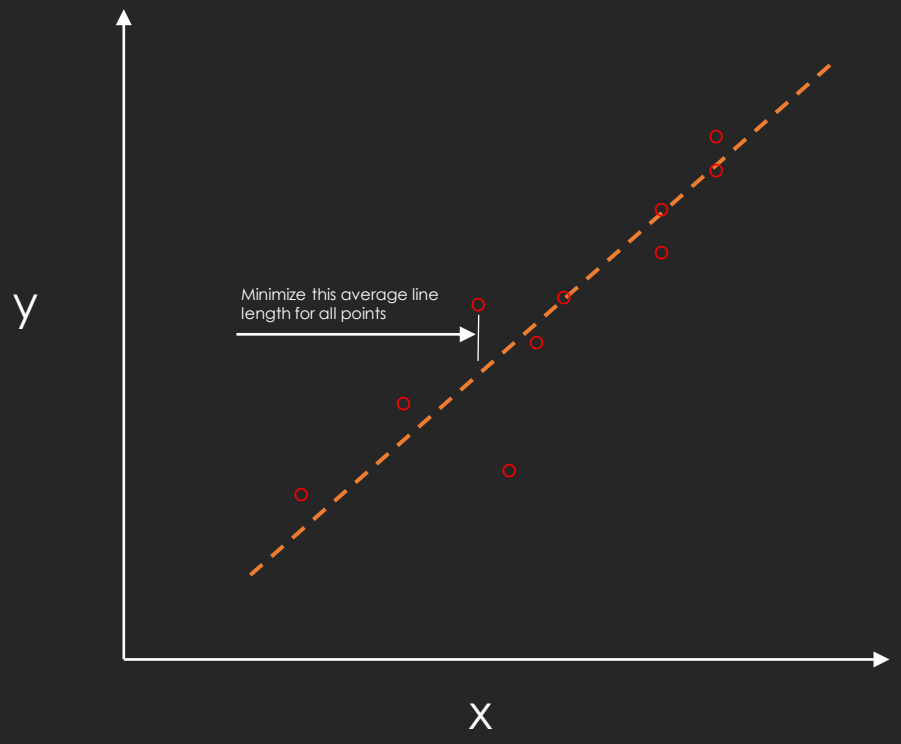
1. StatQuest: https://www.youtube.com/channel/UCtYLUTtgS3k1Fg4y5tAhLbw
2. MML book: https://mml-book.github.io/
3. The D2L.ai book (Section 2.6)
4. Google.

Break

CODING SESSION

Break

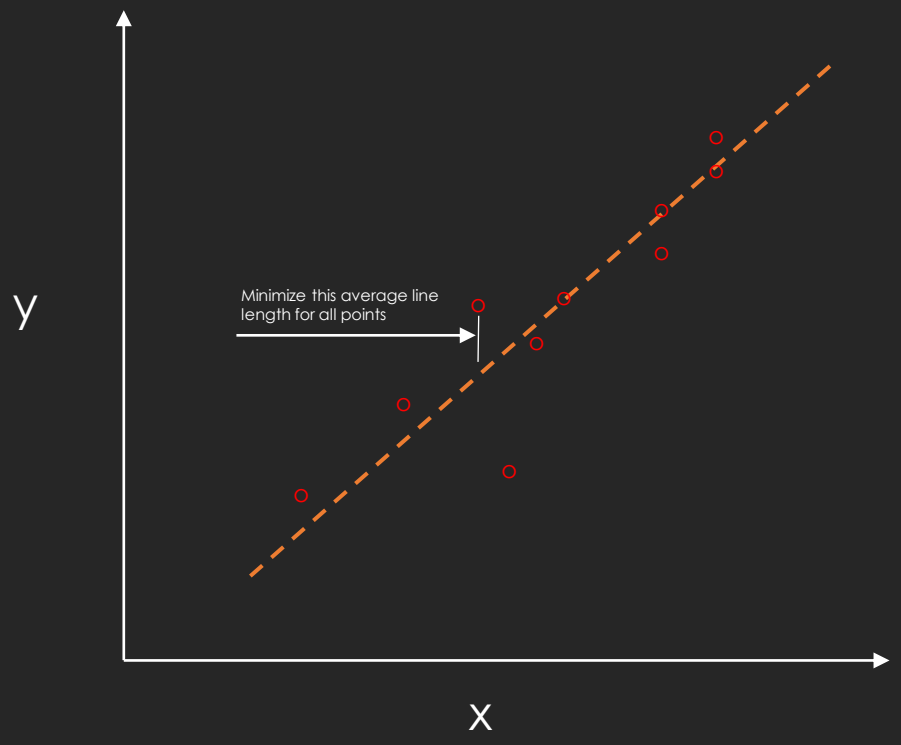
Linear Regression



$y = ax + b$

Objective function: Squared error

Linear Regression

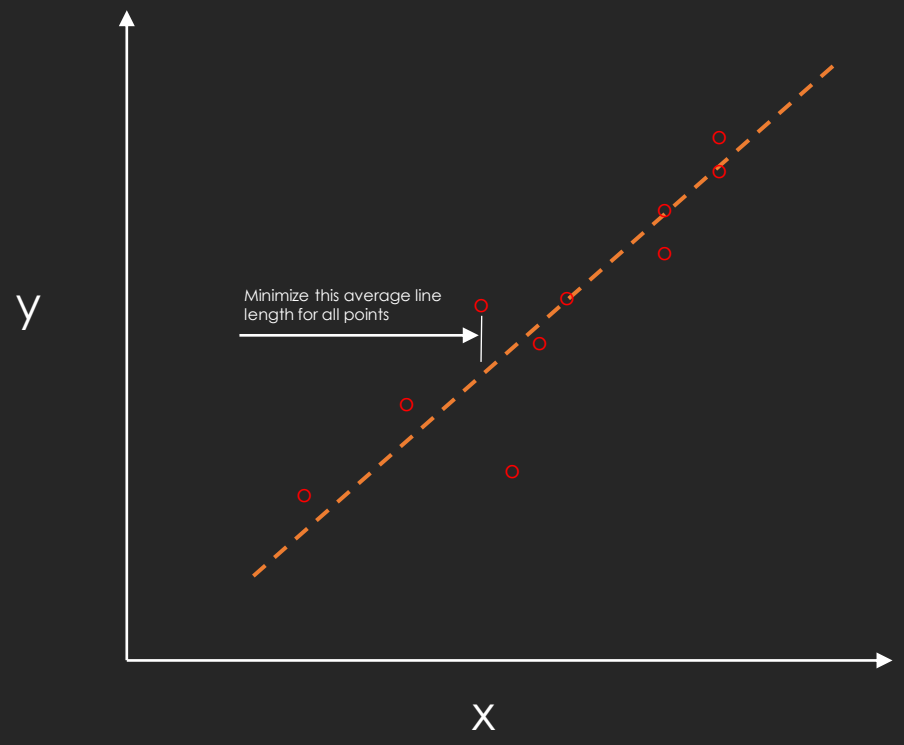


$y = ax + b$

Objective function: Squared error

Loss = 0.5*(predictions – actual)**2

Linear Regression



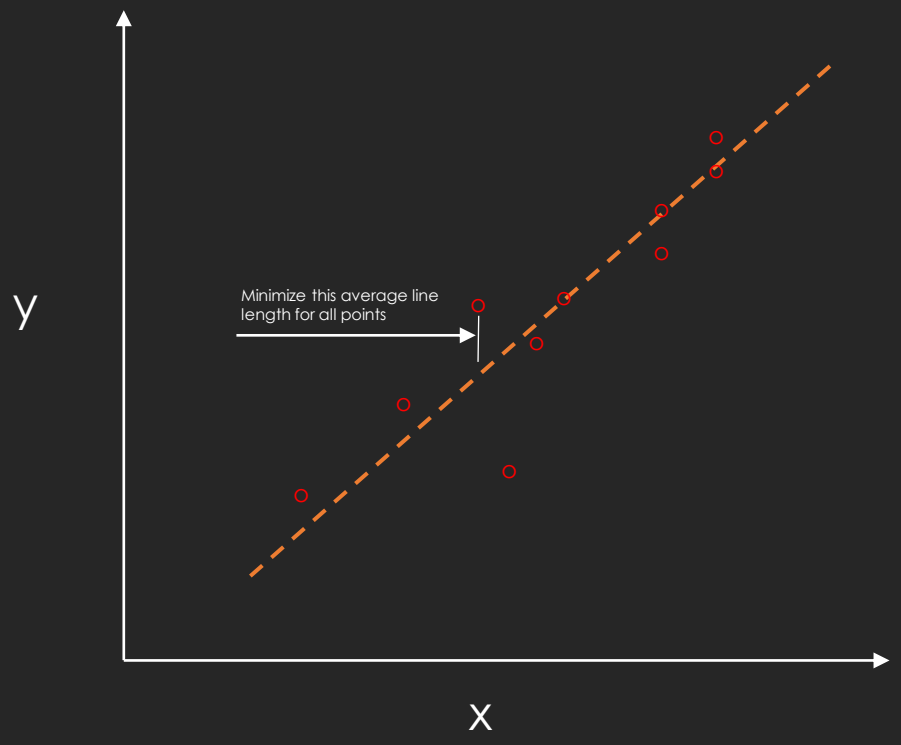
$y = ax + b$

Objective function: Squared error

Loss = 0.5*(predictions – actual)**2

Updates:
$(a,b)^N$ <- $(a,b)^{N-1}$ – lr * mean(D_i(loss(a,b;x_i)))

Linear Regression



$y = ax + b$

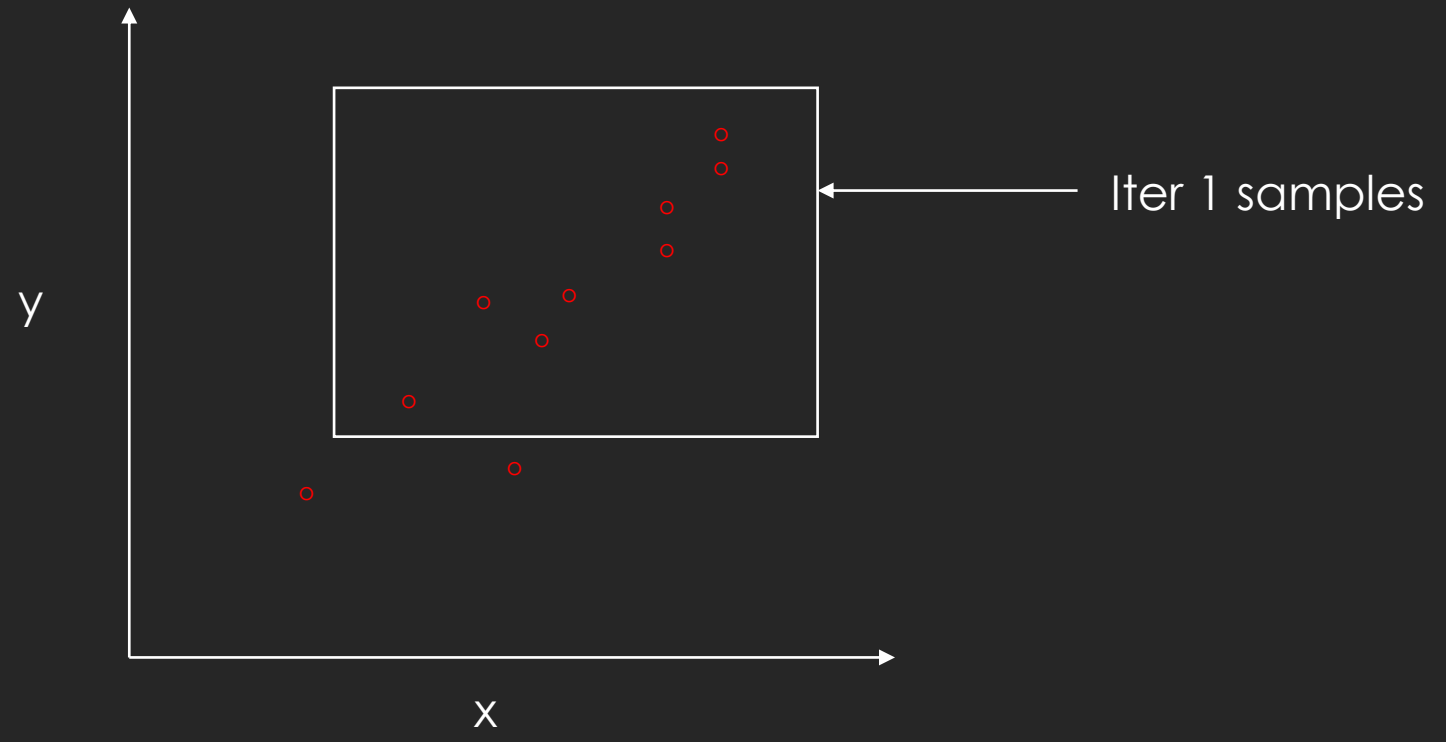
Objective function: Squared error

Loss = 0.5*(predictions – actual)**2

Updates:
$(a,b)^N$ <- $(a,b)^{N-1}$ – lr * mean(D_i(loss(a,b;x_i)))

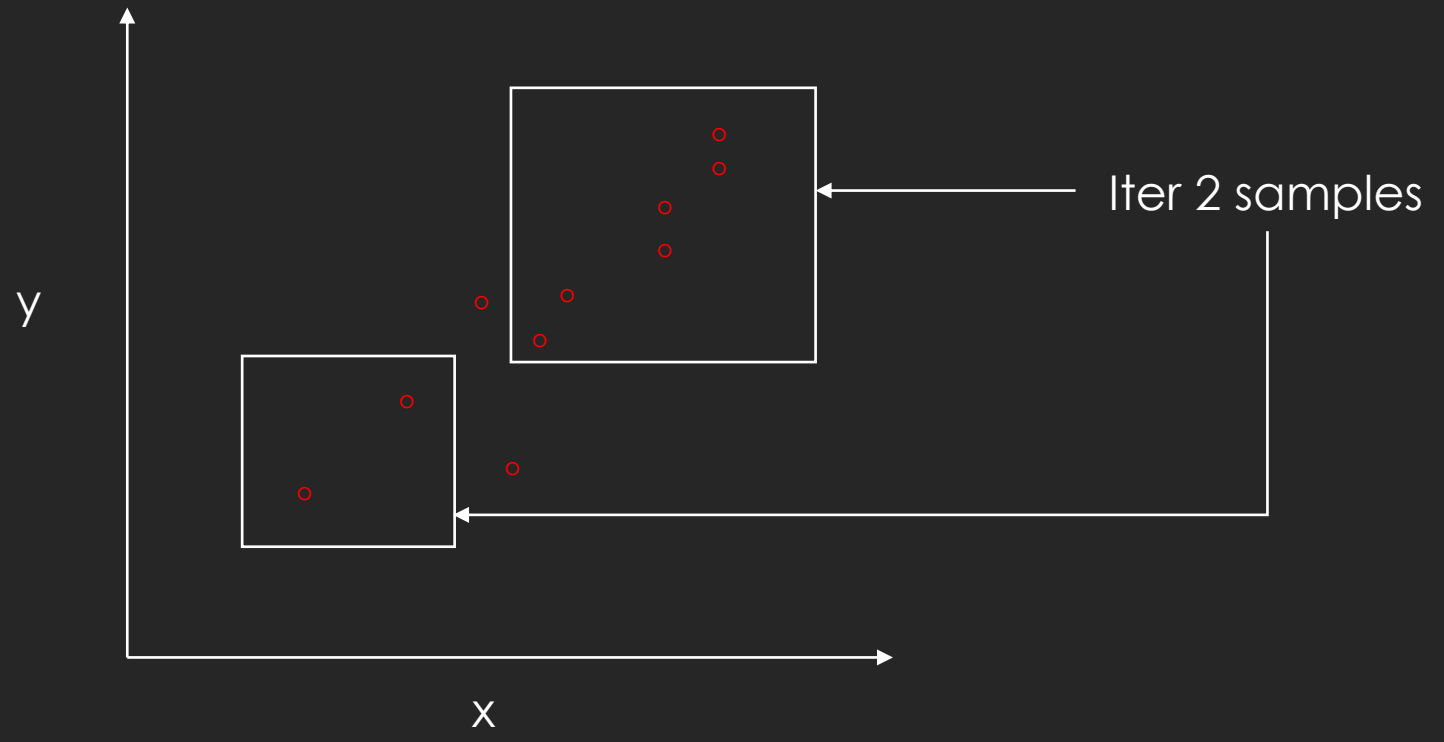
Algorithm: Stochastic Gradient Descent (SGD)

SGD



Dataset size = 10
Samples = 8

SGD



Dataset size = 10
Samples = 8

Break

CODING SESSION

Break

Biology

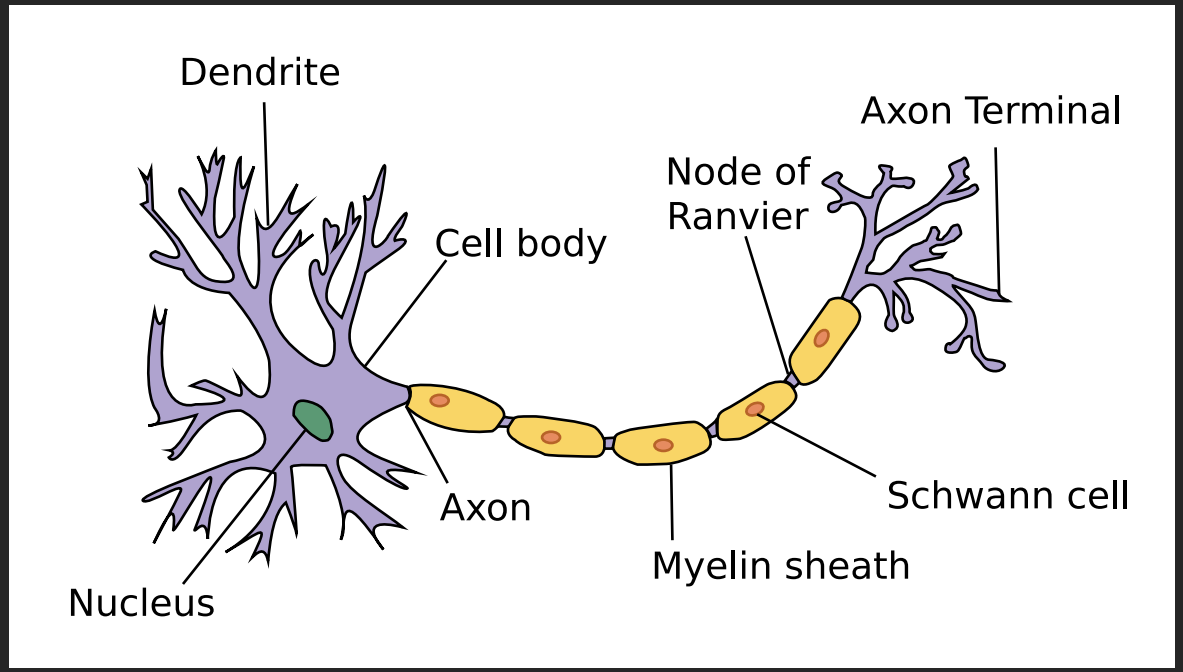


Image source: d2l.ai

Softmax Regression
(Multiclass Classification)

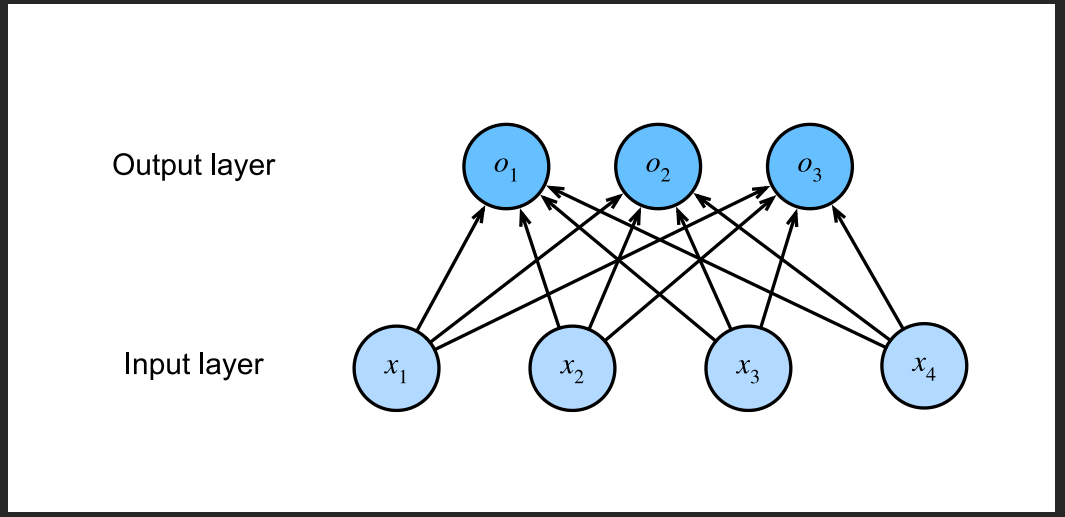


Image source: d2l.ai

Softmax Regression

(Multiclass Classification)

$$P(y|x) = \prod_i P(y^i|x^i)$$

Softmax Regression

(Multiclass Classification)

$$P(y|x) = \prod_i P(y^i|x^i)$$

For a Binary classifier,

$$P(y|x) = P(y)^y * P(\bar{y})^{\bar{y}}$$

Softmax Regression
(Multiclass Classification)

$$P(y|x) = \prod_i P(y^i|x^i)$$

For a Binary classifier,

$$P(y|x) = P(y)^y * P(\bar{y})^{\bar{y}}$$

$$P(y|x) = p(y)^y * \big(1 - p(y)\big)^{1-y}$$

← This is a Bernoulli distribution

Softmax Regression
(Multiclass Classification)

$$P(y|x) = \prod_i P(y^i|x^i)$$

For a Binary classifier,

$$P(y|x) = P(y)^y * P(\bar{y})^{\bar{y}}$$

$$P(y|x) = p(y)^y * \big(1 - p(y)\big)^{1-y}$$ ← This is a Bernoulli distribution

$$\log(P(y|x)) \Rightarrow y \log P(y) + (1 - y) \log\big(1 - P(y)\big)$$

Softmax Regression

(Multiclass Classification)

$$\log(P(y|x)) \Rightarrow y \log P(y) + (1 - y) \log\big(1 - P(y)\big)$$

Softmax Regression

(Multiclass Classification)

$$\log(P(y|x)) \Rightarrow y \log P(y) + (1 - y) \log\big(1 - P(y)\big)$$

Extending to multiple classes,

$$\Rightarrow y_1 \log P(y_1) + y_2 \log P(y_2) + \cdots + y_k \log P(y_k)$$

Binary Cross Entropy
(Multilabel Classification)

$$\Rightarrow y_1 \log P(y_1) + y_2 \log P(y_2) + \cdots + y_k \log P(y_k)$$

$$+$$

$$(1 - y_1) * \log\big(1 - P(y_1)\big) + (1 - y_2) * \log\big(1 - P(y_2)\big) + \cdots$$

Break

CODING SESSION

Break

References

https://stats.stackexchange.com/questions/245502/why-should-we-shuffle-data-while-training-a-neural-network

https://gregorygundersen.com/blog/2020/02/09/log-sum-exp/