Lecture 5: Review of Neural Networks

CS109B Data Science 2
Pavlos Protopapas and Mark Glickman
Artificial Neural Networks

1. Machine Learning Algorithm
2. Very simplified parametric models of our brain
3. Networks of basic processing units: neurons
4. Neurons store information which has to be learned (the weights or the state for RNNs)
5. Many types of architectures to solve different tasks
6. They can scale to massive data
Outline

Anatomy of a NN
Design choices
Learning
Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- Activation function
- Loss function
- Output units
- Architecture

Learning
Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- Activation function
- Loss function
- Output units
- Architecture

Learning
Anatomy of artificial neural network (ANN)

input

neuron
node

output

X → node → Y
Anatomy of artificial neural network (ANN)

Input neuron node output

\[ Y = f(h) \]

We will talk later about the choice of activation function.
We will talk later about the choice of the output layer and the loss function.
Anatomy of artificial neural network (ANN)

Input layer  hidden layer 1  hidden layer 2  output layer

$X_1 \xrightarrow{} W_{11} \xrightarrow{} W_{21}$

$X_2 \xrightarrow{} W_{12} \xrightarrow{} W_{22}$

$Y$
Anatomy of artificial neural network (ANN)

We will talk later about the choice of the number of layers.
Anatomy of artificial neural network (ANN)

Input layer

hidden layer 1, 3 nodes

hidden layer $n$, 3 nodes

output layer

$X_1$

$X_2$

$W_{11}$

$W_{12}$

$W_{13}$

$W_{n1}$

$W_{n2}$

$W_{n3}$

$Y$
Anatomy of artificial neural network (ANN)

Input layer, hidden layer 1, hidden layer $n$, output layer

$m$ nodes

We will talk later about the choice of the number of nodes.
Anatomy of artificial neural network (ANN)

Input layer  hidden layer 1,  hidden layer \( n \)  output layer

\( m \) nodes

Number of inputs is specified by the data
Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- Activation function
- Loss function
- Output units
- Architecture

Learning
Activation function

\[ h = f(W^T X + b) \]

The activation function should:

- Ensures not linearity
- Ensure gradients remain large through hidden unit

Common choices are

- Sigmoid
- Relu, leaky ReLU, Generalized ReLU, MaxOut
- softplus
- tanh
- swish
Activation function

\[ h = f(W^T X + b) \]

The activation function should:

- \textbf{Ensures not linearity}
- Ensure gradients remain large through hidden unit

Common choices are

- Sigmoid
- Relu, leaky ReLU, Generalized ReLU, MaxOut
- softplus
- tanh
- swish
Beyond Linear Models

Linear models

– Can be fit efficiently (via convex optimization)
– Limited model capacity

Alternative:

\[ f(x) = w^T \phi(x) \]

Where \( \phi \) is a non-linear transform
Traditional ML

Manually engineer $\phi$

- Domain specific, enormous human effort

Generic transform

- Maps to a higher-dimensional space
- Kernel methods: e.g. RBF kernels
- **Over fitting:** does not generalize well to test set
- Cannot encode enough prior information
Deep Learning

- Directly learn $\phi$

$$f(x; \theta) = W^T \phi(x; \theta)$$

- where $\theta$ are parameters of the transform
- $\phi$ defines hidden layers
- Can encode prior beliefs, generalizes well
Activation function

\[ h = f(W^T X + b) \]

The activation function should:

- Ensures not linearity
- Ensure gradients remain large through hidden unit

Common choices are:

- Sigmoid
- Relu, leaky ReLU, Generalized ReLU, MaxOut
- softplus
- tanh
- swish
<table>
<thead>
<tr>
<th>Activation function</th>
<th>Equation</th>
<th>Example</th>
<th>1D Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit step (Heaviside)</td>
<td>( \phi(z) = \begin{cases} 0, &amp; z &lt; 0, \ 0.5, &amp; z = 0, \ 1, &amp; z &gt; 0, \end{cases} )</td>
<td>Perceptron variant</td>
<td></td>
</tr>
<tr>
<td>Sign (Signum)</td>
<td>( \phi(z) = \begin{cases} -1, &amp; z &lt; 0, \ 0, &amp; z = 0, \ 1, &amp; z &gt; 0, \end{cases} )</td>
<td>Perceptron variant</td>
<td></td>
</tr>
<tr>
<td>Linear</td>
<td>( \phi(z) = z )</td>
<td>Adaline, linear regression</td>
<td></td>
</tr>
<tr>
<td>Piece-wise linear</td>
<td>( \phi(z) = \begin{cases} 1, &amp; z \geq \frac{1}{2}, \ z + \frac{1}{2}, &amp; -\frac{1}{2} &lt; z &lt; \frac{1}{2}, \ 0, &amp; z \leq -\frac{1}{2}, \end{cases} )</td>
<td>Support vector machine</td>
<td></td>
</tr>
<tr>
<td>Logistic (sigmoid)</td>
<td>( \phi(z) = \frac{1}{1 + e^{-z}} )</td>
<td>Logistic regression, Multi-layer NN</td>
<td></td>
</tr>
<tr>
<td>Hyperbolic tangent</td>
<td>( \phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} )</td>
<td>Multi-layer Neural Networks</td>
<td></td>
</tr>
<tr>
<td>Rectifier, ReLU (Rectified Linear Unit)</td>
<td>( \phi(z) = \text{max}(0, z) )</td>
<td>Multi-layer Neural Networks</td>
<td></td>
</tr>
<tr>
<td>Rectifier, softplus</td>
<td>( \phi(z) = \ln(1 + e^z) )</td>
<td>Multi-layer Neural Networks</td>
<td></td>
</tr>
</tbody>
</table>
Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

⦁ Activation function
⦁ Loss function
⦁ Output units
⦁ Architecture

Learning
Cross-entropy between training data and model distribution (i.e. negative log-likelihood)

$$J(W) = -\mathbb{E}_{x,y \sim \hat{p}_{\text{data}}} \log p_{\text{model}}(y|x)$$

Do not need to design separate loss functions.

Gradient of cost function must be large enough
Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

• Activation function
• Loss function
• Output units
• Architecture

Learning
<table>
<thead>
<tr>
<th>Output Type</th>
<th>Output Distribution</th>
<th>Output layer</th>
<th>Cost Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Link function

\[ X \Rightarrow \phi(X) = W^T X \Rightarrow P(y = 0) = \frac{1}{1 + e^{\phi(x)}} \]

![Diagram of link function]

\[ X \xRightarrow{\phi(X)} \sigma(\phi) \xrightarrow{\text{OUTPUT UNIT}} Y \]
### Output Units

<table>
<thead>
<tr>
<th>Output Type</th>
<th>Output Distribution</th>
<th>Output layer</th>
<th>Cost Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>Bernoulli</td>
<td>Sigmoid</td>
<td>Binary Cross Entropy</td>
</tr>
</tbody>
</table>
## Output Units

<table>
<thead>
<tr>
<th>Output Type</th>
<th>Output Distribution</th>
<th>Output layer</th>
<th>Cost Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>Bernoulli</td>
<td>Sigmoid</td>
<td>Binary Cross Entropy</td>
</tr>
<tr>
<td>Discrete</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Link function multi-class problem

\[ Y = \frac{e^{\phi_k(X)}}{\sum_{k=1}^{K} e^{\phi_k(X)}} \]
## Output Units

<table>
<thead>
<tr>
<th>Output Type</th>
<th>Output Distribution</th>
<th>Output layer</th>
<th>Cost Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>Bernoulli</td>
<td>Sigmoid</td>
<td>Binary Cross Entropy</td>
</tr>
<tr>
<td>Discrete</td>
<td>Multinoulli</td>
<td>Softmax</td>
<td>Cross Entropy</td>
</tr>
</tbody>
</table>
## Output Units

<table>
<thead>
<tr>
<th>Output Type</th>
<th>Output Distribution</th>
<th>Output layer</th>
<th>Cost Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>Bernoulli</td>
<td>Sigmoid</td>
<td>Binary Cross Entropy</td>
</tr>
<tr>
<td>Discrete</td>
<td>Multinoulli</td>
<td>Softmax</td>
<td>Cross Entropy</td>
</tr>
<tr>
<td>Continuous</td>
<td>Gaussian</td>
<td>Linear</td>
<td>MSE</td>
</tr>
</tbody>
</table>
### Output Units

<table>
<thead>
<tr>
<th>Output Type</th>
<th>Output Distribution</th>
<th>Output layer</th>
<th>Cost Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>Bernoulli</td>
<td>Sigmoid</td>
<td>Binary Cross Entropy</td>
</tr>
<tr>
<td>Discrete</td>
<td>Multinoulli</td>
<td>Softmax</td>
<td>Cross Entropy</td>
</tr>
<tr>
<td>Continuous</td>
<td>Gaussian</td>
<td>Linear</td>
<td>MSE</td>
</tr>
<tr>
<td>Continuous</td>
<td>Arbitrary</td>
<td>-</td>
<td>GANS</td>
</tr>
</tbody>
</table>
Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

• Activation function
• Loss function
• Output units
• Architecture

Learning
Architecture

Input  Hidden  Output

![Architecture Diagram]

- real: $x \sin(x)$
- pred

![Graph]

$0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7$

$-4 \quad -2 \quad 0 \quad 2 \quad 4$
Architecture (cont)

Input  Hidden  Output

![Diagram showing a neural network with input, hidden, and output layers. The diagram includes a graph plotting the function $y = x \sin(x)$ and its prediction.](chart)
Architecture (cont)
Architecture (cont)

Diagram showing the architecture with input, hidden, and output layers.

Graph plotting the function $x \sin(x)$ and predicted values.

```
Input  →  Hidden  →  Output
```

```
0  1  2  3  4  5  6  7
0  2  0  -2  0  2  4  0
```

Legend: real: $x \sin(x)$, pred
Architecture (cont)

Input  Hidden  Hidden  Output

![Diagram of the architecture with input, hidden, and output layers.](image)

Graph showing real versus predicted values.

- **real**: $x \sin(x)$
- **pred**

CS109B, Procopapas, Glickman
Architecture (cont)

Input → Hidden → Hidden → Output

Graph showing a neural network architecture with input, hidden, and output layers. The graph includes connections between nodes and a plot with two curves labeled 'real: xsin(x)' and 'pred'.
Architecture (cont)

[Diagram showing a neural network with multiple hidden layers and an output layer.]
Universal Approximation Theorem

Think of Neural Network as function approximation.

\[ Y = f(x) + \epsilon \]
\[ Y = \hat{f}(x) + \epsilon \]

NN: \( \Rightarrow \hat{f}(x) \)

**One hidden layer is enough** to represent an approximation of any function to an arbitrary degree of accuracy

So why deeper?
- Shallow net may need (exponentially) more width
- Shallow net may overfit more
What does an astronomer blow with gum?

Hubbles
Auditors: Volunteers
To view

Download the add-in.
liveslides.com/download

Start the presentation.
Better Generalization with Depth

(Goodfellow 2017)
Large, Shallow Nets Overfit More

(Goodfellow 2017)
Why layers? Representation

Representation Matters

Cartesian coordinates vs. Polar coordinates
Learning Multiple Components

Rule-based systems

Input → Hand designed program → Output

Classic machine learning

Input → Hand designed features → Mapping from features → Output

Representation learning

Input → Features → Mapping from features → Output

Deep learning

Input → Simple features → Additional layers of more abstract features → Mapping from features → Output
Depth = Repeated Compositions

Visible layer (input pixels)
1st hidden layer (edges)
2nd hidden layer (corners and contours)
3rd hidden layer (object parts)
Output (object identity)
Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- Activation function
- Loss function
- Output units
- Architecture

Learning
Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- Activation function
- Loss function
- Output units
- Architecture

Learning (more next lecture)

Basics ideas of optimizer

Backprop
# Heart Data

<table>
<thead>
<tr>
<th>Age</th>
<th>Sex</th>
<th>ChestPain</th>
<th>RestBP</th>
<th>Chol</th>
<th>Fbs</th>
<th>RestECG</th>
<th>MaxHR</th>
<th>ExAng</th>
<th>Oldpeak</th>
<th>Slope</th>
<th>Ca</th>
<th>Thal</th>
<th>AHD</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>1</td>
<td>typical</td>
<td>145</td>
<td>233</td>
<td>1</td>
<td>2</td>
<td>150</td>
<td>0</td>
<td>2.3</td>
<td>3</td>
<td>0.0</td>
<td>fixed</td>
<td>No</td>
</tr>
<tr>
<td>67</td>
<td>1</td>
<td>asymptomatic</td>
<td>160</td>
<td>286</td>
<td>0</td>
<td>2</td>
<td>108</td>
<td>1</td>
<td>1.5</td>
<td>2</td>
<td>3.0</td>
<td>normal</td>
<td>Yes</td>
</tr>
<tr>
<td>67</td>
<td>1</td>
<td>asymptomatic</td>
<td>120</td>
<td>229</td>
<td>0</td>
<td>2</td>
<td>129</td>
<td>1</td>
<td>2.6</td>
<td>2</td>
<td>2.0</td>
<td>reversible</td>
<td>Yes</td>
</tr>
<tr>
<td>37</td>
<td>1</td>
<td>nonanginal</td>
<td>130</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>187</td>
<td>0</td>
<td>3.5</td>
<td>3</td>
<td>0.0</td>
<td>normal</td>
<td>No</td>
</tr>
<tr>
<td>41</td>
<td>0</td>
<td>nontypical</td>
<td>130</td>
<td>204</td>
<td>0</td>
<td>2</td>
<td>172</td>
<td>0</td>
<td>1.4</td>
<td>1</td>
<td>0.0</td>
<td>normal</td>
<td>No</td>
</tr>
</tbody>
</table>

**response variable Y is Yes/No**
Basic idea of learning

Start with Regression or Logistic Regression

**Classification**
\[ f(X) = \frac{1}{1 + e^{-W^T X}} \]

**Regression**
\[ f(X) = W^T X \]

\[ W^T = [W_0, W_1, ..., W_4] = [\beta_0, \beta_1, ..., \beta_4] \]

\[ Y = f(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4) \]

- Intercept or **Bias**
- Coefficients or **Weights**
But what is the idea?

Start with all randomly selected weights. Most likely it will perform horribly. For example, in our heart data, the model will be giving us the wrong answer.

\[
\hat{y} = 0.9 \rightarrow \text{Yes} \quad y = \text{No}
\]

\[
\begin{align*}
\text{MaxHR} &= 200 \\
\text{Age} &= 52 \\
\text{Sex} &= \text{Male} \\
\text{Chol} &= 152
\end{align*}
\]
But what is the idea?

Start with all randomly selected weights. Most likely it will perform horribly. For example, in our heart data, the model will be giving us the wrong answer.

\[ \hat{p} = 0.4 \rightarrow No \]

\[ y=Yes \]

\[ MaxHR = 170 \]
\[ Age = 42 \]
\[ Sex = Male \]
\[ Chol = 342 \]
But what is the idea?

- **Loss Function**: Takes all of these results and averages them and tells us how bad or good the computer or those weights are.

- Telling the computer how **bad** or **good** is, does not help.

- You want to tell it how to change those weights so it gets better.

Loss function: $\mathcal{L}(w_0, w_1, w_2, w_3, w_4)$

For now let’s only consider one weight, $\mathcal{L}(w_1)$
But what is the idea?

Trial and error:

Change the weights and see the effect.

This can take long long time especially in NN where we have millions of weights to adjust.
Minimizing the Loss function

Ideally we want to know the value of $w_1$ that gives the minimum $L(W)$.

To find the optimal point of a function $L(W)$

$$\frac{dL(W)}{dW} = 0$$

And find the $W$ that satisfies that equation. Sometimes there is no explicit solution for that.
Minimizing the Loss function

A more flexible method is

• Start from any point
  • Determine which direction to go to reduce the loss (left or right)
  • Specifically, we can calculate the slope of the function at this point
  • Shift to the right if slope is negative or shift to the left if slope is positive
• Repeat
Minimization of the Loss Function

If the step is proportional to the slope then you avoid overshooting the minimum.

**Question:** What is the mathematical function that describes the slope?

**Question:** How do we generalize this to more than one predictor?

**Question:** What do you think it is a good approach for telling the model how to change (what is the step size) to become better?
Minimization of the Loss Function

If the step is proportional to the slope then you avoid overshooting the minimum.

**Question:** What is the mathematical function that describes the slope?  
**Derivative**

**Question:** How do we generalize this to more than one predictor?  
**Take the derivative with respect to each coefficient and do the same sequentially**

**Question:** What do you think it is a good approach for telling the model how to change (what is the step size) to become better?  
**More on this later**
Let’s play the Pavlos game

We know that we want to go in the opposite direction of the derivative and we know we want to be making a step proportionally to the derivative.

Making a step means:

\[ w^{\text{new}} = w^{\text{old}} + \text{step} \]

Opposite direction of the derivative means:

\[ w^{\text{new}} = w^{\text{old}} - \lambda \frac{dL}{dw} \]

Change to more conventional notation:

\[ w^{(i+1)} = w^{(i)} - \lambda \frac{dL}{dw} \]
Gradient Descent

- Algorithm for optimization of first order to finding a minimum of a function.
- It is an iterative method.
- $L$ is decreasing in the direction of the negative derivative.
- The learning rate is controlled by the magnitude of $\lambda$.

$$w^{(i+1)} = w^{(i)} - \lambda \frac{dL}{dw}$$
Considerations

• We still need to derive the derivatives.

• We need to know what is the learning rate or how to set it.

• We need to avoid local minima.

• Finally, the full likelihood function includes summing up all individual ‘errors’. Unless you are a statistician, this can be hundreds of thousands of examples.
Local vs Global Minima

\[ \theta \]

\[ L \]
Local vs Global Minima

\[ L \] vs \[ \theta \]

- Local minima vs global minima in optimization problems.
- The diagram illustrates the concept with a loss function \( L \) and parameters \( \theta \).
- The path of optimization is shown with red arrows, converging towards the global minimum.
Local vs Global Minima

No guarantee that we get the global minimum.

**Question:** What would be a good strategy?
Large data
Batch and Stochastic Gradient Descent

\[ \mathcal{L} = - \sum_i [y_i \log p_i + (1 - y_i) \log(1 - p_i)] \]

Instead of using all the examples for every step, use a subset of them (batch).

For each iteration \( k \), use the following loss function to derive the derivatives:

\[ \mathcal{L}^k = - \sum_{i \in b^k} [y_i \log p_i + (1 - y_i) \log(1 - p_i)] \]

which is an approximation to the full Loss function.
Batch and Stochastic Gradient Descent

Full Likelihood:
Batch Likelihood:
Batch and Stochastic Gradient Descent

Full Likelihood:
Batch Likelihood:
Batch and Stochastic Gradient Descent

Full Likelihood: $L\theta$

Batch Likelihood: $L'\theta$
Batch and Stochastic Gradient Descent

Full Likelihood:
Batch Likelihood:
Batch and Stochastic Gradient Descent

Full Likelihood:  
Batch Likelihood:
Batch and Stochastic Gradient Descent

Full Likelihood:  
Batch Likelihood:
Batch and Stochastic Gradient Descent

Full Likelihood:  
Batch Likelihood:
Batch and Stochastic Gradient Descent

\[ L \]

\[ \theta \]

Full Likelihood:

Batch Likelihood:
Batch and Stochastic Gradient Descent

\[
\begin{align*}
L(	heta) &= \text{Full Likelihood:} \\
L_b &= \text{Batch Likelihood:}
\end{align*}
\]
Learning Rate
Learning Rate

NEXT LECTURE

There are many alternative methods which address how to set or adjust the learning rate, using the derivative or second derivatives and or the momentum. To be discussed in the next lectures on NN.

* TLDR: J. Bullinaria, “Learning with Momentum, Conjugate Gradient Learning”, 2015
Considerations

• We still need to derive the derivatives.

• We need to know what is the learning rate or how to set it.

• We need to avoid local minima.

• Finally, the full likelihood function includes summing up all individual ‘errors’. Unless you are a statistician, this can be hundreds of thousands of examples.
Considerations

• We still need to derive the derivatives.

• We need to know what is the learning rate or how to set it.

• We need to avoid local minima.

• Finally, the full likelihood function includes summing up all individual ‘errors’. Unless you are a statistician, this can be hundreds of thousands of examples.
Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- Activation function
- Loss function
- Output units
- Architecture

Learning (more next lecture)

Basics ideas of optimizer

Backprop
Derivatives: Linear Regression

\[
f = \sum_i (y_i - \beta_0 - \beta_1 x_i)^2
\]

\[
\frac{df}{d\beta_1} = 0 \Rightarrow 2 \sum_i (y_i - \beta_0 - \beta_1 x_i)(-x_i)
\]

\[
\frac{df}{d\beta_0} = 0 \Rightarrow 2 \sum_i (y_i - \beta_0 - \beta_1 x_i)
\]

\[
- \sum_i x_i y_i + \beta_0 \sum_i x_i + \beta_1 \sum_i x_i^2 = 0
\]

\[
- \sum_i x_i y_i + (\bar{y} - \beta_1 \bar{x}) \sum_i x_i + \beta_1 \sum_i x_i^2 = 0
\]

\[
\beta_1 \left( \sum_i x_i^2 - n \bar{x}^2 \right) = \sum_i x_i y_i - n \bar{x} \bar{y}
\]

\[
\Rightarrow \beta_1 = \frac{\sum_i x_i y_i - n \bar{x} \bar{y}}{\sum_i x_i^2 - n \bar{x}^2}
\]

\[
\Rightarrow \beta_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}
\]
Derivatives: Logistic Regression

Can we do it?

Wolfram Alpha can do it for us!

We need a general formalism to deal with these derivatives.
Backprop: Chain Rule

• Chain rule for computing gradients:

\[ y = g(x) \quad z = f(y) = f(g(x)) \]

\[ \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \]

• For longer chains

\[ \frac{\partial z}{\partial x_i} = \sum_{j_1} \ldots \sum_{j_m} \frac{\partial z}{\partial y_{j_1}} \ldots \frac{\partial y_{j_m}}{\partial x_i} \]
For logistic regression, the -ve log of the likelihood is:

\[
\mathcal{L} = \sum_i \mathcal{L}_i = \sum_i \log L_i = \sum_i \left[ y_i \log p_i + (1 - y_i) \log (1 - p_i) \right]
\]

\[
\mathcal{L}_i = -y_i \log \frac{1}{1 + e^{-w^T x}} - (1 - y_i) \log \left(1 - \frac{1}{1 + e^{-w^T x}}\right)
\]

To simplify the analysis let us split it into two parts,

\[
\mathcal{L}_i = \mathcal{L}_i^A + \mathcal{L}_i^B
\]

So the derivative with respect to \( W \) is:

\[
\frac{\partial \mathcal{L}}{\partial W} = \sum_i \frac{\partial \mathcal{L}_i}{\partial W} = \sum_i \left( \frac{\partial \mathcal{L}_i^A}{\partial W} + \frac{\partial \mathcal{L}_i^B}{\partial W} \right)
\]
\[ L_i^A = -y_i \log \frac{1}{1 + e^{-WX}} \]

<table>
<thead>
<tr>
<th>Variables</th>
<th>( \frac{\partial \xi}{\partial W} )</th>
<th>( \frac{\partial \xi}{\partial W} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \xi_1 = -WX )</td>
<td>( -X )</td>
<td>( -X )</td>
</tr>
<tr>
<td>( \xi_2 = e^{\xi_1} = e^{-WX} )</td>
<td>( e^{\xi_1} )</td>
<td>( e^{-WX} )</td>
</tr>
<tr>
<td>( \xi_3 = 1 + \xi_2 = 1 + e^{-WX} )</td>
<td>( 1 )</td>
<td>( 1 )</td>
</tr>
<tr>
<td>( \xi_4 = \frac{1}{\xi_3} = \frac{1}{1 + e^{-WX}} = p )</td>
<td>( \frac{1}{\xi_3} )</td>
<td>( \frac{1}{\xi_3} )</td>
</tr>
<tr>
<td>( \xi_5 = \log \xi_4 = \log p = \log \frac{1}{1 + e^{-WX}} )</td>
<td>( \frac{1}{\xi_4} )</td>
<td>( \frac{1}{\xi_4} )</td>
</tr>
</tbody>
</table>

\[ \frac{\partial L_i^A}{\partial W} = \frac{\partial L_i}{\partial \xi_5} \frac{\partial \xi_5}{\partial \xi_4} \frac{\partial \xi_4}{\partial \xi_3} \frac{\partial \xi_3}{\partial \xi_2} \frac{\partial \xi_2}{\partial \xi_1} \frac{\partial \xi_1}{\partial W} \]

\[ \frac{\partial L_i^A}{\partial W} = -yXe^{-WX} \frac{1}{1 + e^{-WX}} \]

\[ \frac{\partial L_i^A}{\partial W} = -yXe^{-WX} \frac{1}{(1 + e^{-WX})^2} \]
\[ L_i^B = -(1 - y_i) \log \left[ 1 - \frac{1}{1 + e^{-W^TX}} \right] \]

<table>
<thead>
<tr>
<th>Variables</th>
<th>derivatives</th>
<th>Partial derivatives wrt to X,W</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \xi_1 = -W^TX )</td>
<td>( \frac{\partial \xi_1}{\partial W} = -X )</td>
<td>( \frac{\partial \xi_1}{\partial W} = -X )</td>
</tr>
<tr>
<td>( \xi_2 = e^{\xi_1} = e^{-W^TX} )</td>
<td>( \frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1} )</td>
<td>( \frac{\partial \xi_2}{\partial \xi_1} = e^{-W^TX} )</td>
</tr>
<tr>
<td>( \xi_3 = 1 + \xi_2 = 1 + e^{-W^TX} )</td>
<td>( \frac{\partial \xi_3}{\partial \xi_2} = 1 )</td>
<td>( \frac{\partial \xi_3}{\partial \xi_2} = 1 )</td>
</tr>
<tr>
<td>( \xi_4 = \frac{1}{\xi_3} = \frac{1}{1 + e^{-W^TX}} = p )</td>
<td>( \frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2} )</td>
<td>( \frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{(1 + e^{-W^TX})^2} )</td>
</tr>
<tr>
<td>( \xi_5 = 1 - \xi_4 = 1 - \frac{1}{1 + e^{-W^TX}} )</td>
<td>( \frac{\partial \xi_5}{\partial \xi_4} = -1 )</td>
<td>( \frac{\partial \xi_5}{\partial \xi_4} = -1 )</td>
</tr>
<tr>
<td>( \xi_6 = \log \xi_5 = \log(1 - p) = \log \frac{1}{1 + e^{-W^TX}} )</td>
<td>( \frac{\partial \xi_6}{\partial \xi_5} = \frac{1}{\xi_5} )</td>
<td>( \frac{\partial \xi_6}{\partial \xi_5} = \frac{1 + e^{-W^TX}}{e^{-W^TX}} )</td>
</tr>
<tr>
<td>( L_i^B = (1 - y)\xi_6 )</td>
<td>( \frac{\partial L}{\partial \xi_6} = 1 - y )</td>
<td>( \frac{\partial L}{\partial \xi_6} = 1 - y )</td>
</tr>
<tr>
<td>( \frac{\partial L_i^B}{\partial W} = \frac{\partial L_i^B}{\partial \xi_6} \frac{\partial \xi_6}{\partial \xi_5} \frac{\partial \xi_5}{\partial \xi_4} \frac{\partial \xi_4}{\partial \xi_3} \frac{\partial \xi_3}{\partial \xi_2} \frac{\partial \xi_2}{\partial \xi_1} \frac{\partial \xi_1}{\partial W} )</td>
<td>( \frac{\partial L_i^B}{\partial W} = (1 - y)X \frac{1}{(1 + e^{-W^TX})} )</td>
<td></td>
</tr>
</tbody>
</table>
Backpropagation: Logistic Regression Revisited

\[ X \rightarrow \text{Affine} \rightarrow h = \beta_0 + \beta_1 X \rightarrow \text{Activation} \rightarrow p = \frac{1}{1 + e^{-h}} \rightarrow \text{Loss Fun} \rightarrow \mathcal{L}(\beta) = \sum_i^n \mathcal{L}_i(\beta) \]

\[ \frac{\partial \mathcal{L}}{\partial p} \frac{\partial p}{\partial h} \frac{\partial h}{\partial \beta} \]

\[ \frac{\partial h}{\partial \beta_1} = X, \quad \frac{d \mathcal{L}}{d \beta_0} = 1 \]

\[ \frac{\partial p}{\partial h} = \sigma(h)(1 - \sigma(h)) \]

\[ \frac{\partial \mathcal{L}}{\partial p} = -y \frac{1}{p} - (1 - y) \frac{1}{1-p} \]

\[ \frac{\partial \mathcal{L}}{\partial \beta_1} = -X\sigma(h)(1 - \sigma(h))[y \frac{1}{p} + (1 - y) \frac{1}{1-p}] \]

\[ \frac{\partial \mathcal{L}}{\partial \beta_0} = -\sigma(h)(1 - \sigma(h))[y \frac{1}{p} + (1 - y) \frac{1}{1-p}] \]
Backpropagation

1. Derivatives need to be evaluated at some values of $X$, $y$ and $W$.
2. But since we have an expression, we can build a function that takes as input $X$, $y$, $W$ and returns the derivatives and then we can use gradient descent to update.
3. This approach works well but it does not generalize. For example if the network is changed, we need to write a new function to evaluate the derivatives.

For example this network will need a different function for the derivatives,
Backpropagation

1. Derivatives need to be evaluated at some values of $X$, $y$ and $W$.
2. But since we have an expression, we can build a function that takes as input $X$, $y$, $W$ and returns the derivatives and then we can use gradient descent to update.
3. This approach works well but it does not generalize. For example if the network is changed, we need to write a new function to evaluate the derivatives.

Than this one:
Need to find a formalism to calculate the derivatives of the loss function wrt to weights that is:

1. Flexible enough that adding a node or a layer or changing something in the network won’t require to re-derive the functional form from scratch.
2. It is exact.
3. It is computationally efficient.

Hints:

1. Remember we only need to evaluate the derivatives at $X_i, y_i$ and $W^{(k)}$.
2. We should take advantage of the chain rule we learned before
Idea 1: Evaluate the derivative at: $X=\{3\}, y=1, W=3$

<table>
<thead>
<tr>
<th>Variables</th>
<th>derivatives</th>
<th>Value of the variable</th>
<th>Value of the partial derivative</th>
<th>$d\xi_n\over dW$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\xi_1 = -W^T X$</td>
<td>$\frac{\partial \xi_1}{\partial W} = -X$</td>
<td>$-9$</td>
<td>$-3$</td>
<td>$-3$</td>
</tr>
<tr>
<td>$\xi_2 = e^{\xi_1} = e^{-W^T x}$</td>
<td>$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$</td>
<td>$e^{-9}$</td>
<td>$e^{-9}$</td>
<td>$-3e^{-9}$</td>
</tr>
<tr>
<td>$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T x}$</td>
<td>$\frac{\partial \xi_3}{\partial \xi_2} = 1$</td>
<td>$1 + e^{-9}$</td>
<td>$1$</td>
<td>$-3e^{-9}$</td>
</tr>
<tr>
<td>$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1 + e^{-W^T x}} = p$</td>
<td>$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$</td>
<td>$\frac{1}{1 + e^{-9}}$</td>
<td>$\left(\frac{1}{1 + e^{-9}}\right)^2$</td>
<td>$-3e^{-9}\left(\frac{1}{1 + e^{-9}}\right)^2$</td>
</tr>
<tr>
<td>$\xi_5 = \log \xi_4 = \log p = \log \frac{1}{1 + e^{-W^T x}}$</td>
<td>$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$</td>
<td>$\log \frac{1}{1 + e^{-9}}$</td>
<td>$1 + e^{-9}$</td>
<td>$-3e^{-9}\left(\frac{1}{1 + e^{-9}}\right)$</td>
</tr>
<tr>
<td>$L_i^A = -y \xi_5$</td>
<td>$\frac{\partial L_i^A}{\partial \xi_5} = -y$</td>
<td>$-\log \frac{1}{1 + e^{-9}}$</td>
<td>$-1$</td>
<td>$3e^{-9}\left(\frac{1}{1 + e^{-9}}\right)$</td>
</tr>
<tr>
<td>$\frac{\partial L_i^A}{\partial W} = \frac{\partial L_i}{\partial \xi_5} \frac{\partial \xi_5}{\partial \xi_4} \frac{\partial \xi_4}{\partial \xi_3} \frac{\partial \xi_3}{\partial \xi_2} \frac{\partial \xi_2}{\partial \xi_1} \frac{\partial \xi_1}{\partial W}$</td>
<td></td>
<td></td>
<td>$-3$</td>
<td>0.00037018372</td>
</tr>
</tbody>
</table>
We still need to derive derivatives 😞

<table>
<thead>
<tr>
<th>Variables</th>
<th>derivatives</th>
<th>Value of the variable</th>
<th>Value of the partial derivative</th>
<th>( \frac{d\xi_n}{dW} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \xi_1 = -W^TX )</td>
<td>( \frac{\partial \xi_1}{\partial W} = -X )</td>
<td>-9</td>
<td>-3</td>
<td>-3</td>
</tr>
<tr>
<td>( \xi_2 = e^{\xi_1} = e^{-W^TX} )</td>
<td>( \frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1} )</td>
<td>( e^{-9} )</td>
<td>( e^{-9} )</td>
<td>-3e^{-9}</td>
</tr>
<tr>
<td>( \xi_3 = 1 + \xi_2 = 1 + e^{-W^TX} )</td>
<td>( \frac{\partial \xi_3}{\partial \xi_2} = 1 )</td>
<td>1 + ( e^{-9} )</td>
<td>1</td>
<td>-3e^{-9}</td>
</tr>
<tr>
<td>( \xi_4 = \frac{1}{\xi_3} = \frac{1}{1 + e^{-W^TX}} = p )</td>
<td>( \frac{\partial \xi_4}{\partial \xi_3} = -\frac{\xi_2}{\xi_3^2} )</td>
<td>( \frac{1}{1 + e^{-9}} )</td>
<td>( \left( \frac{1}{1 + e^{-9}} \right)^2 )</td>
<td>-3e^{-9} ( \left( \frac{1}{1 + e^{-9}} \right)^2 )</td>
</tr>
<tr>
<td>( \xi_5 = \log \xi_4 = \log p = \log \frac{1}{1 + e^{-W^TX}} )</td>
<td>( \frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4} )</td>
<td>( \log \frac{1}{1 + e^{-9}} )</td>
<td>1 + ( e^{-9} )</td>
<td>-3e^{-9} ( \left( \frac{1}{1 + e^{-9}} \right) )</td>
</tr>
<tr>
<td>( \mathcal{L}_l^A = -y \xi_5 )</td>
<td>( \frac{\partial \mathcal{L}_l^A}{\partial \xi_5} = -y )</td>
<td>-( \log \frac{1}{1 + e^{-9}} )</td>
<td>-1</td>
<td>3e^{-9} ( \left( \frac{1}{1 + e^{-9}} \right) )</td>
</tr>
<tr>
<td>[ \frac{\partial \mathcal{L}_l^A}{\partial W} ] &amp; ( \frac{\partial \mathcal{L}_l^A}{\partial \xi_5} \frac{\partial \xi_5}{\partial \xi_4} \frac{\partial \xi_4}{\partial \xi_3} \frac{\partial \xi_3}{\partial \xi_2} \frac{\partial \xi_2}{\partial \xi_1} \frac{d\xi_1}{dW} ) &amp; -3</td>
<td>0.00037018372</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Notice though those are basic functions that my grandparent can do

| $\xi_0 = X$ | $\frac{\partial \xi_0}{\partial X} = 1$ | def x0(x):
| | | return X |
| | | def derx0():
| | | return 1 |
| $\xi_1 = -W^T \xi_0$ | $\frac{\partial \xi_1}{\partial W} = -X$ | def x1(a,x):
| | | return -a*X |
| | | def derx1(a,x):
| | | return -a |
| $\xi_2 = e^{\xi_1}$ | $\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$ | def x2(x):
| | | return np.exp(x) |
| | | def derx2(x):
| | | return np.exp(x) |
| $\xi_3 = 1 + \xi_2$ | $\frac{\partial \xi_3}{\partial \xi_2} = 1$ | def x3(x):
| | | return 1+x |
| | | def derx3(x):
| | | return 1 |
| $\xi_4 = \frac{1}{\xi_3}$ | $\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$ | def der1(x):
| | | return 1/(x) |
| | | def derx4(x):
| | | return -(1/x)**(2) |
| $\xi_5 = \log \xi_4$ | $\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$ | def der1(x):
| | | return np.log(x) |
| | | def derx5(x)
| | | return 1/x |
| $L_i^A = -y\xi_5$ | $\frac{\partial L}{\partial \xi_5} = -y$ | def derL(y,x):
| | | return -y*x |
| | | def derL(y):
| | | return -y |
Autograd: Auto-differentiation

1. We specify the network structure

\[ X \xrightarrow{W_1} W_3 \xrightarrow{W_4} W_5 \xrightarrow{} Y \]

2. We create the computational graph ...

What is computational graph?
\[
\begin{align*}
\xi_3 &= 1 + e^{-WTX} \\
\xi_2 &= e^{-\xi_1} \\
\xi_2' &= -e^{-\xi_1} \\
\xi_1 &= WTX \\
\xi_1' &= X \\
\xi_0 &= W \\
\xi_4 &= \frac{1}{1 + e^{-WTX}} \\
\xi_5 &= \log \left( \frac{1}{1 + e^{-WTX}} \right) \\
\xi_6 &= 1 - \frac{1}{1 + e^{-WTX}} \\
\xi_7 &= \log(1 - \frac{1}{1 + e^{-WTX}}) \\
\xi_8 &= (1 - y)\log(1 - \frac{1}{1 + e^{-WTX}}) \\
\xi_9 &= y\log(\frac{1}{1 + e^{-WTX}}) \\
-L &= \xi_9 = y\log(\frac{1}{1 + e^{-WTX}}) + (1 - y)\log(1 - \frac{1}{1 + e^{-WTX}})
\end{align*}
\]
1. We specify the network structure

\[ X \xrightarrow{W_1} W_3 \xrightarrow{W_5} Y \]

\[ W_2 \xrightarrow{W_4} \]

- We create the computational graph.
- At each node of the graph we build two functions: the evaluation of the variable and its partial derivative with respect to the previous variable (as shown in the table few slides back)
- Now we can either go forward or backward depending on the situation. In general, forward is easier to implement and to understand. The difference is clearer when there are multiple nodes per layer.
**Forward mode:** Evaluate the derivative at: $X=\{3\}, \ y=1, \ W=3$

<table>
<thead>
<tr>
<th>Variables</th>
<th>derivatives</th>
<th>Value of the variable</th>
<th>Value of the partial derivative</th>
<th>$\frac{dL}{d\xi_n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\xi_1 = -W^T X$</td>
<td>$\frac{\partial \xi_1}{\partial W} = -X$</td>
<td>$-9$</td>
<td>$-3$</td>
<td>$-3$</td>
</tr>
<tr>
<td>$\xi_2 = e^{\xi_1} = e^{-W^T X}$</td>
<td>$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$</td>
<td>$e^{-9}$</td>
<td>$e^{-9}$</td>
<td>$-3e^{-9}$</td>
</tr>
<tr>
<td>$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T X}$</td>
<td>$\frac{\partial \xi_3}{\partial \xi_2} = 1$</td>
<td>$1 + e^{-9}$</td>
<td>$1$</td>
<td>$-3e^{-9}$</td>
</tr>
<tr>
<td>$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1 + e^{-W^T X}} = p$</td>
<td>$\frac{\partial \xi_4}{\partial \xi_3} = \frac{-1}{(1 + e^{-9})^2}$</td>
<td>$\frac{1}{1 + e^{-9}}$</td>
<td>$\left(\frac{1}{1 + e^{-9}}\right)^2$</td>
<td>$-3e^{-9}\left(\frac{1}{1 + e^{-9}}\right)^2$</td>
</tr>
<tr>
<td>$\xi_5 = \log \xi_4 = \log p = \log \frac{1}{1 + e^{-W^T X}}$</td>
<td>$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$</td>
<td>$\log \frac{1}{1 + e^{-9}}$</td>
<td>$1 + e^{-9}$</td>
<td>$-3e^{-9}\left(\frac{1}{1 + e^{-9}}\right)$</td>
</tr>
<tr>
<td>$L_i^A = -y \xi_5$</td>
<td>$\frac{\partial L}{\partial \xi_5} = -y$</td>
<td>$-\log \frac{1}{1 + e^{-9}}$</td>
<td>$-1$</td>
<td>$3e^{-9}\left(\frac{1}{1 + e^{-9}}\right)$</td>
</tr>
<tr>
<td>$\frac{\partial L_i^A}{\partial W} = \frac{\partial L_i}{\partial \xi_5} \frac{\partial \xi_5}{\partial \xi_4} \frac{\partial \xi_4}{\partial \xi_3} \frac{\partial \xi_3}{\partial \xi_2} \frac{\partial \xi_2}{\partial \xi_1} \frac{\partial \xi_1}{\partial W}$</td>
<td></td>
<td></td>
<td>$-3$</td>
<td>$0.00037018372$</td>
</tr>
</tbody>
</table>
**Backward mode:** Evaluate the derivative at: $X=\{3\}, y=1, W=3$

<table>
<thead>
<tr>
<th>Variables</th>
<th>derivatives</th>
<th>Value of the variable</th>
<th>Value of the partial derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\xi_1 = -W^T X$</td>
<td>$\frac{\partial \xi_1}{\partial W} = -X$</td>
<td>$-9$</td>
<td>$-3$</td>
</tr>
<tr>
<td>$\xi_2 = e^{\xi_1} = e^{-W^T X}$</td>
<td>$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$</td>
<td>$e^{-9}$</td>
<td>$e^{-9}$</td>
</tr>
<tr>
<td>$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T X}$</td>
<td>$\frac{\partial \xi_3}{\partial \xi_2} = 1$</td>
<td>$1+e^{-9}$</td>
<td>$1$</td>
</tr>
<tr>
<td>$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1 + e^{-W^T X}} = p$</td>
<td>$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$</td>
<td>$\frac{1}{1 + e^{-9}}$</td>
<td>$\left(\frac{1}{1 + e^{-9}}\right)^2$</td>
</tr>
<tr>
<td>$\xi_5 = \log \xi_4 = \log p = \log \frac{1}{1 + e^{-W^T X}}$</td>
<td>$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$</td>
<td>$\log \frac{1}{1 + e^{-9}}$</td>
<td>$1 + e^{-9}$</td>
</tr>
<tr>
<td>$L^A = -y\xi_5$</td>
<td>$\frac{\partial L^A}{\partial \xi_5} = -y$</td>
<td>$-\log \frac{1}{1 + e^{-9}}$</td>
<td>$-1$</td>
</tr>
</tbody>
</table>
| $\frac{\partial L^A}{\partial W} = \frac{\partial L^A}{\partial \xi_5} \frac{\partial \xi_5}{\partial \xi_4} \frac{\partial \xi_4}{

Store all these values