Classification Part 1: Logistic Regression & NN

Rutgers Cognitive Science Club
Rutgers Statistics Club

A Review Of Linear Regression

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$$

- ŷ is a prediction of the linear relationship between k independent variables (also called features) and a dependent variable (target)

- This prediction is **quantitative**- It predicts a continuous numerical quantity e.g. a person’s height, the cost of a house, city population

From Quantitative to Qualitative

Instead of predicting quantitative data, how do we predict data that fits into categories (ie categorical)?

Qualitative (Categorical) Data

- Let's say that our dependent variable now falls into two levels - (e.g. success or failure).

- Examples: Given an image of an animal, can we predict whether it is a cat or a dog? Given a person's shopping preferences, can we predict whether they are male or female?

Qualitative Predictions

- Note that there can be more than 2 levels.

- However, we will initially constrain ourselves to choosing between 2 levels (in this case cat/dog).

- How do we use features (both qualitative and quantitative) to classify things into categories?

A Classification Problem

- Let’s say we have some data on someone’s annual income, their age, and whether they studied political science in college.
- Can we predict whether they are a politician?
- Is it possible to use linear regression for this?

Identifying the Problem Structure

- We can identify 3 independent variables (features) and 1 dependent variable (target):
- x_1 = annual income, x_2 = age, x_3 = political science major or not
- y_1 = politician or not

A Quick Aside on Data Manipulation

- How would we use political science major or not to the regression model? It is not a quantitative variable!

- We can convert this variable into what is called a **dummy variable** or **binary variable** by setting the variable to 1 if political science major and 0 if not

Let's Formalize the Problem

- For our features, we now have:
  - x_1 = age: ranges from 0 to 120(ish)
  - x_2 = annual income: ranges from 0 to millions (rarely 100s of millions & billions)
  - x_3 = was political science major: either 0 or 1
  - y = is a politician: either 0 or 1
- **Note**: for all classification problems, y will always be categorical, i.e. taking on integers, each representing a class (category)

Based On This Formalization …

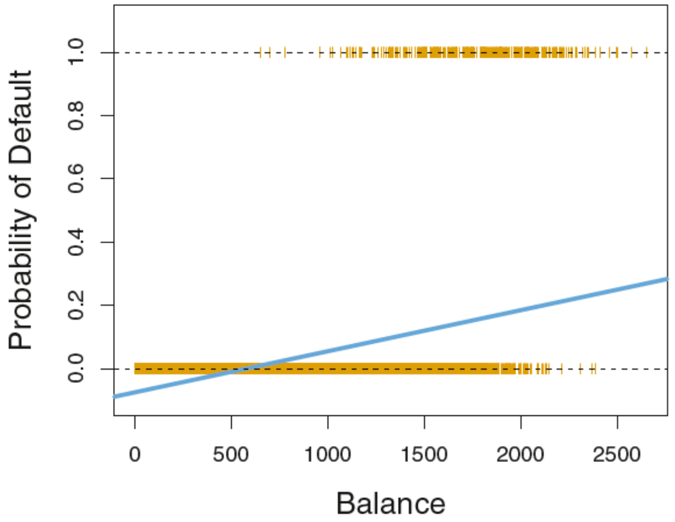
Can we do linear regression?

NO! (For Most Cases)

- Since linear regression will fit a continuous line:
- $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$
- Let’s say we have y take on 3 values: 1 for politician, 2 for lawyer, and 0 for neither.
- Then if linear regression were run, we might get $\hat{y}$=1.5.
- Would it be sensible to say that our person is some hybrid of lawyer and politician?

NO! cont.

- In Chapter 4 of *Introduction to Statistical Learning*, the authors introduce a special case of where linear regression may work for classification.
- But keep in mind, this is the exception and not the rule.
- And even in this special case there are problems.



Logistic Regression

A transition from quantitative to qualitative prediction

Logistic Regression: A Transformation of Linear Regression

- Biggest problem we need to address is forcing our prediction, ŷ, to be either a 0 or 1
- **Let's try**:
  1. $\hat{z} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$
  2. Convert ẑ to some value between 0 and 1 & call it p
  3. Pick a threshold, t, such as 0.5
  4. If p >= t, then ŷ = 1

     If If p < t, then ŷ = 0

1. Linear Model

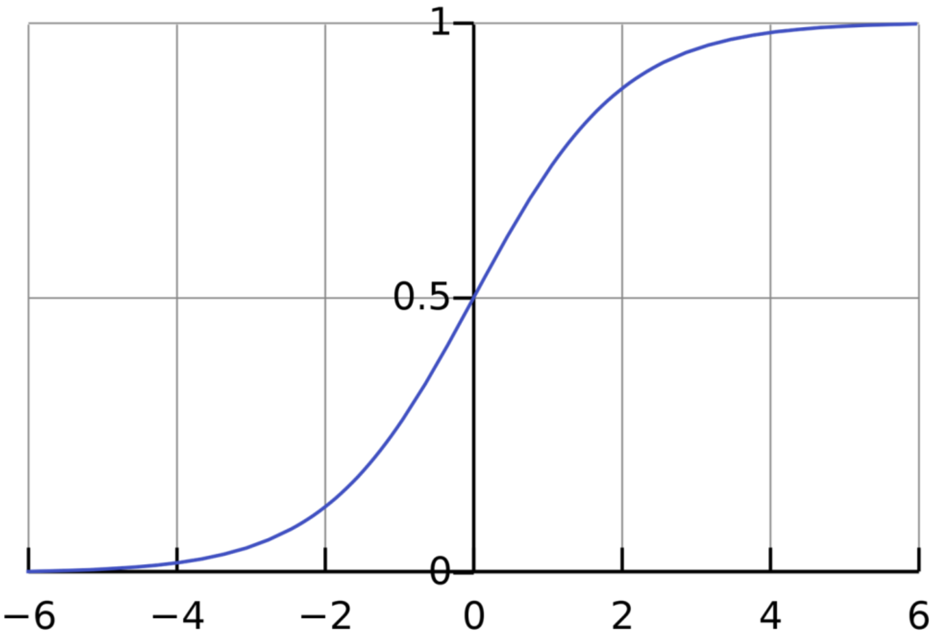
- Hopefully, by now
- $\hat{z} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$ is familiar
- We only use ẑ as opposed to ŷ to denote that this is not our final prediction

2. Introducing Logistic Function

- The **logistic** function (often referred to as **sigmoid** in machine learning) has the form:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

2 (cont). Logistic Function Plot



2. Applying The Logistic Function

- So now, we have

$$p = \sigma(\hat{z}) = \frac{1}{1 + e^{(-\beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k)}}$$

- So p will now be between 0 and 1. We can think of this as the **probability** that y is 1.

Aside: What Are the Odds?

- Let’s say I ask the question, “What are the odds that you get an A on your next exam?”
- Let’s say you tell me 1 to 3 (although I hope they’re better).
- What you’re really saying is the probability, p, that I get an A on my exam is 3 times lower than the probability, (1-p), that I don’t get an A.
- In an equation: **p / (1-p)** = 1/3

Aside cont.: What are the Log-Odds

In our case, we have the **odds**:

$$\frac{p}{1-p} = e^{\beta_0+\beta_1 x_1+\ldots+\beta_k x_k}$$

So the **log-odds** are:

$$log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k$$

So What?

- What this tells us is that:
  - Whereas in linear regression, a unit change in x_i gave us a β_i increase in y.
  - Here, a unit change in x_i yields an **$e^{\beta i}$ increase in the odds that y = 1**.
  - In other words, we can also say that a unit change in x_i gives us a **β_i increase in the log-odds that y = 1.**

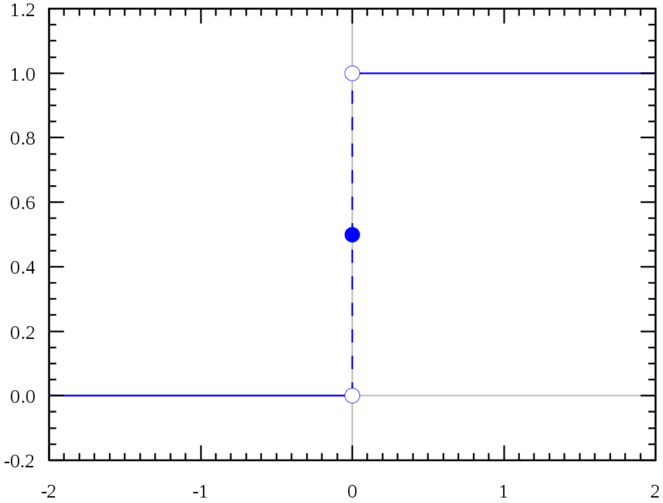
3. Choosing The Threshold

- Normally, it makes sense to simply choose t=0.5.
- However, if we are doctors, for example, and we want to avoid missing a case of cancer if someone has it (**false negative** or **type II error**), but we are okay with accidentally classifying some people without cancer as having it, we can **lower t** to (e.g.) 0.4
- On the other hand, if we are deciding whether to give someone the death penalty or not based on the evidence of the crime, we want to avoid the possibility of classifying someone innocent as guilty (**false positive** or known as **type I** error), which allows for the possibility of mistakenly classifying someone guilty as innocent so we could **raise t** to 0.6 for example.

4. Applying the Threshold

- Then, all that’s left is to decide our prediction ŷ, by doing
  - If p >= t, ŷ = 1
  - If p < t, ŷ = 0

Step (Threshold) Function with t = 0



How Do We Measure Our Cost?

- Intuitively, our cost function should ensure we are maximizing the number of correct predictions given our features.
- So, per datapoint, we want that our predicted value, $\hat{y}$, match y.
- So we need the chances that $\hat{y} = y$ given all our x_i's.
- Also, we normally take the log of this, for similar reasons as to why we talk about log-odds

Maximum Likelihood & Cross Entropy

- We want to maximize the **log-likelihood** over all data points and so here we are actually maximizing the opposite of cost (which we will call **gain**)

$$LLE = \sum_{i=1}^{n} y_i log(p_i) + (1 - y_i) log(1 - p_i)$$

- We can also **minimize** the **negative** of log-likelihood to turn it into a cost function. This cost function is called **binary cross entropy**.

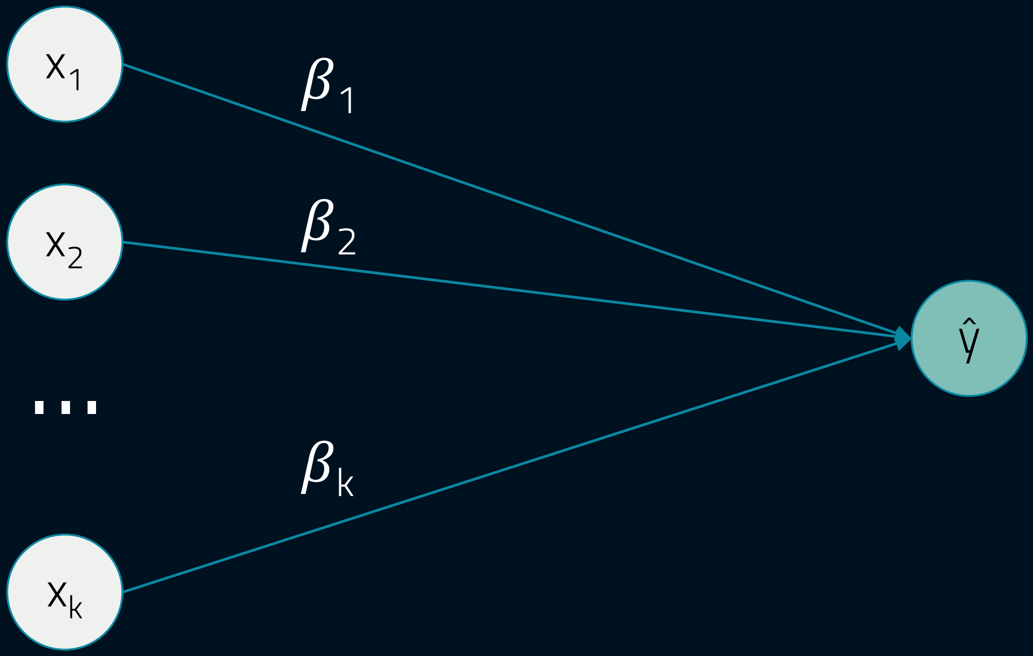
$$BCE = -\sum_{i=1}^{n} y_i log(p_i) + (1 - y_i) log(1 - p_i)$$

In Practice

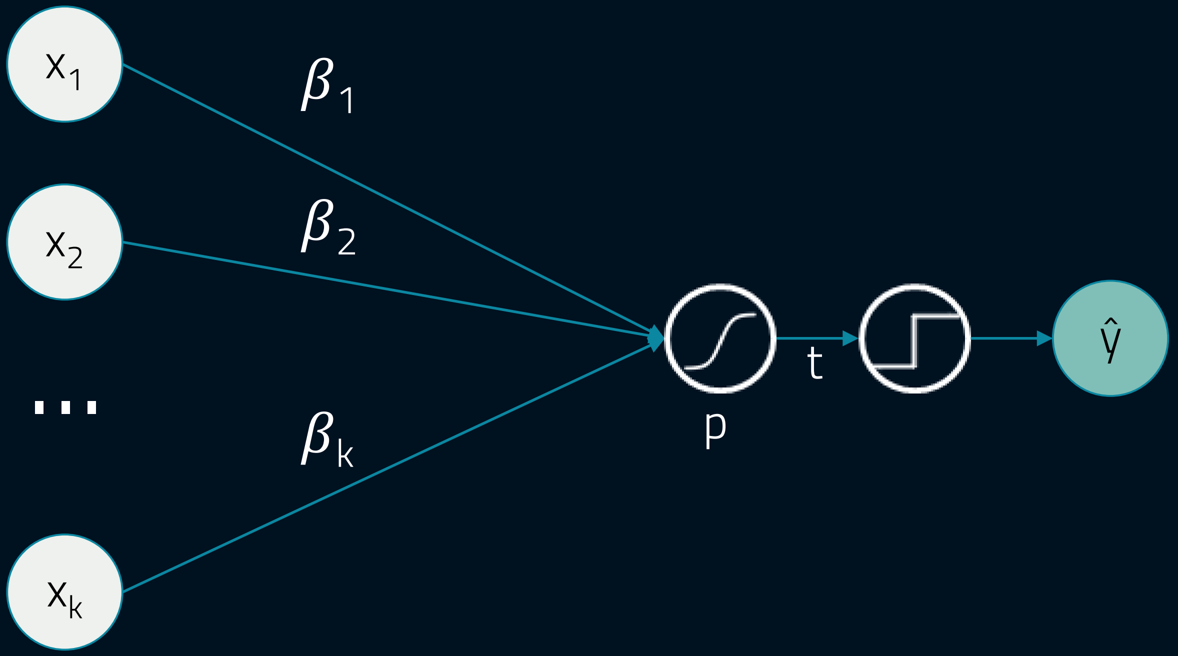
- In practice, the way we do this is train our model, by feeding it data points one by one
- Then, for each datapoint, we compute (if we are using BCE): $-y_i log(p_i) - (1 - y_i) log(1 - p_i)$
- This tells us the error per datapoint. Then, we use something called **gradient descent** to modify our β_i's so that next time around we may correctly classify the point if we didn't this time

A Visual Representation

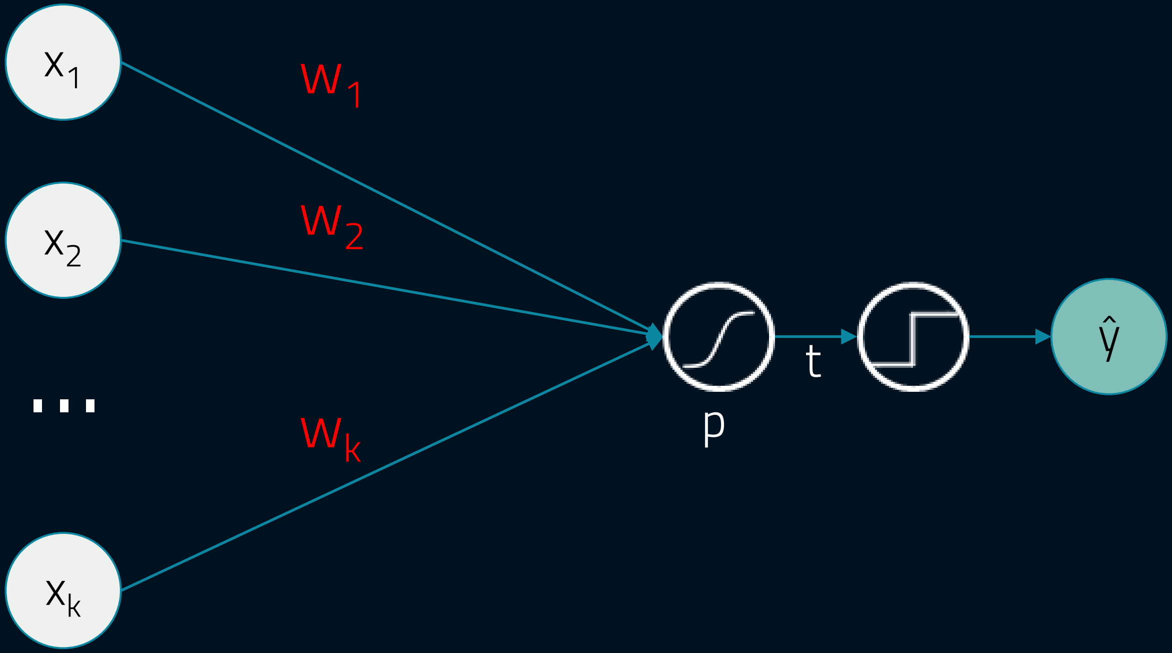
Multiple Linear Regression



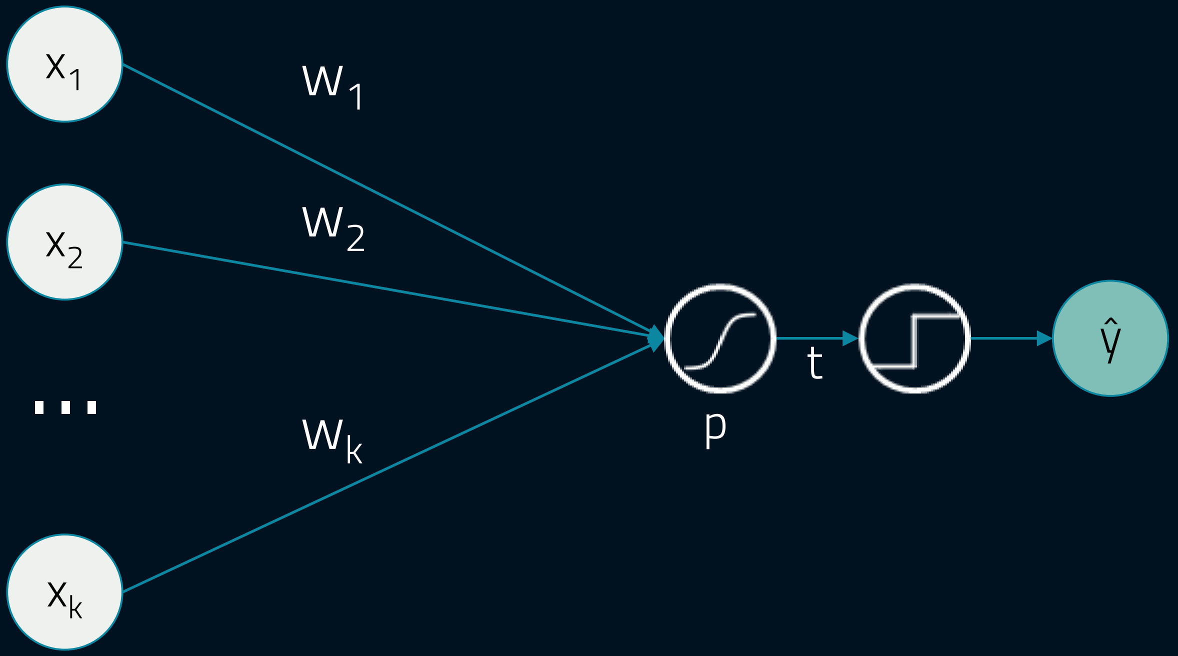
Logistic Regression



Logistic Regression (Relabeled)



Single Layer Neural Network



The Terminology of Neural Networks

- What terminology stays the same?
  - Features (independent variable), target (dependent variable)

The Terminology of Neural Networks

- What changes: **Neurons** and **Layers**:
  - Every location where we can store separate information i.e. the feature values, intermediate values, and prediction values are each referred to as neurons.
  - We now group things into layers: the feature neurons collectively make up the **input layer**. The prediction neurons collectively make up the **output layer**. We can also add neurons in between these two layers for intermediate processing steps. These are called **hidden layers**.

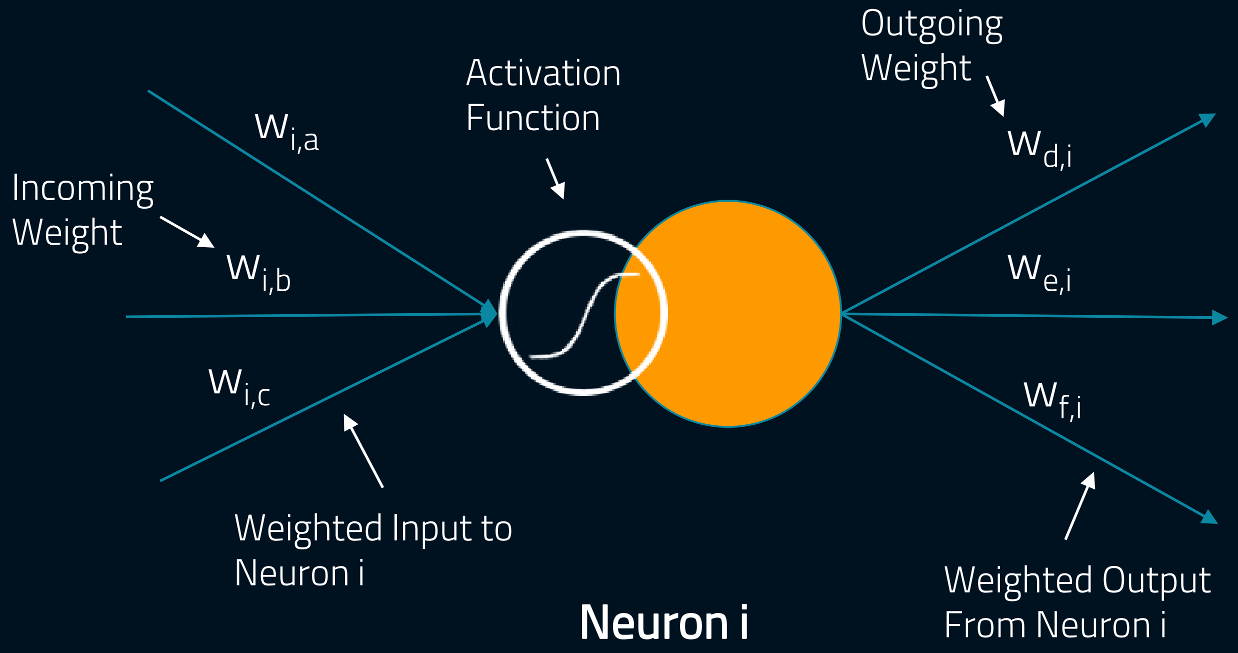
The Terminology of Neural Networks

- What changes: **Weights** and **Activation Functions**:
  - The coefficients that relate the value of each neuron in one layer to the neurons of the subsequent layers are called **weights** and they are often denoted $\mathbf{w}_{j,i}$ indicating the connection between source neuron **i** and destination neuron **j**
  - The logistic or sigmoid function from before is one type of **activation function**.
    - Activation functions let us put all the outputs of neurons on the same scale and also introduce **nonlinearities**

Aside: Nonlinearity

- If we look at one of the benefits of applying the logistic function to our inputs, it is that we can move beyond predicting linear relationships in our data.
- The logistic function provides a way to model some sort of nonlinear relationship (converting our feature input to a prediction of its category)
- Chaining many of these nonlinear relationships one on top of the other allows us to model arbitrarily complicated phenomena

The “Anatomy” of a Neuron



Incoming Weight

Activation Function

Outgoing Weight

$w_{i,a}$

$w_{i,b}$

$w_{i,c}$

$w_{d,i}$

$w_{e,i}$

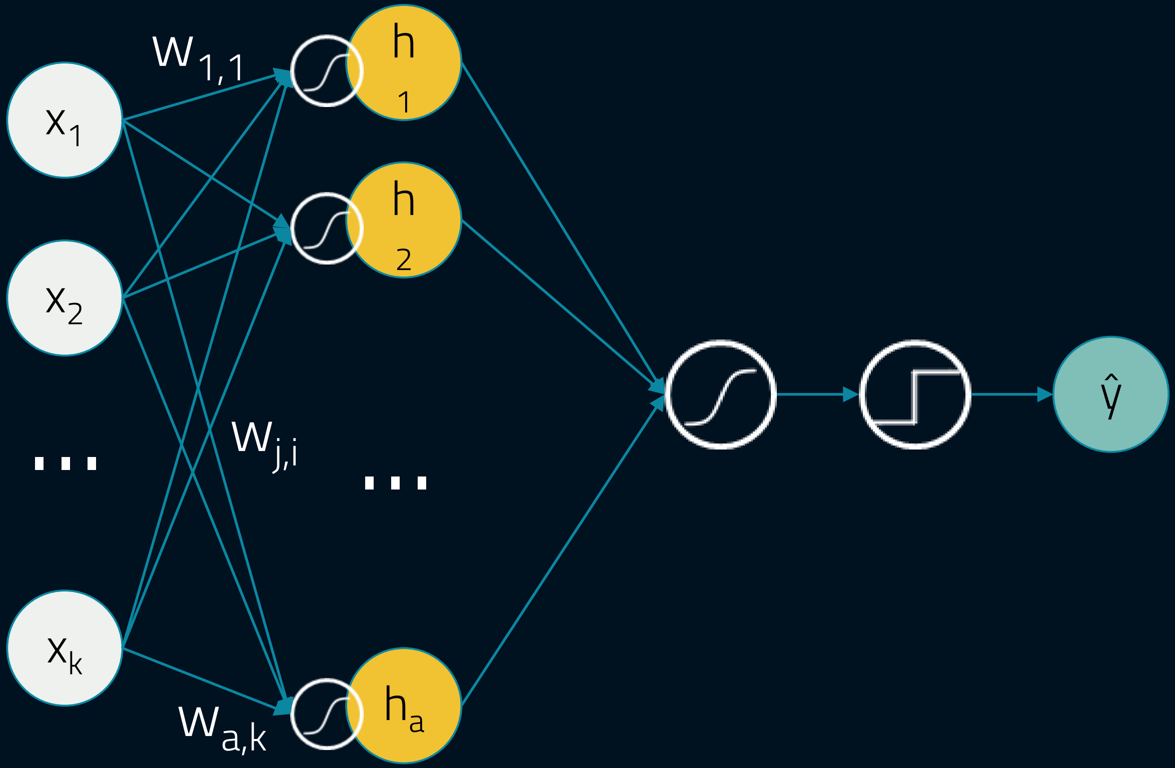
$w_{f,i}$

Weighted Input to Neuron i

Neuron i

Weighted Output From Neuron i

Neural Network With 1 Hidden Layer

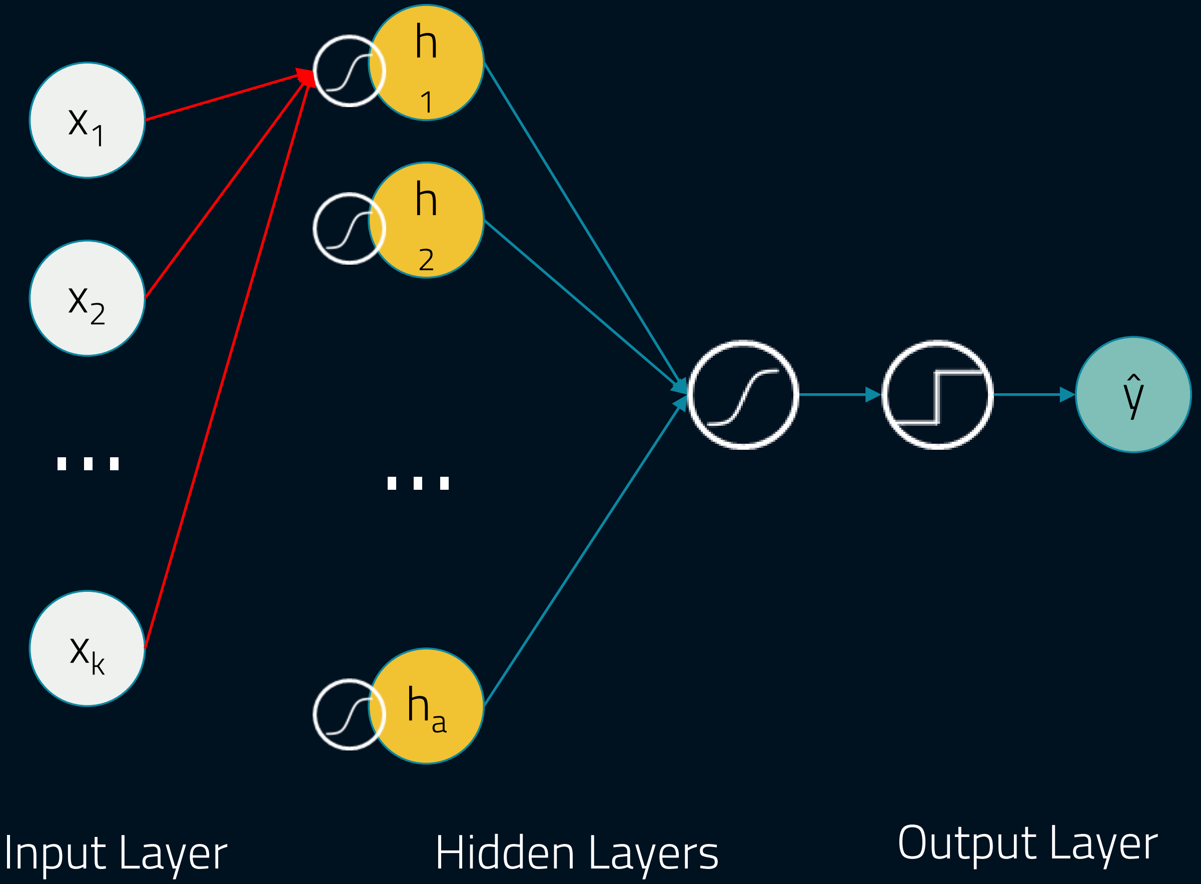


Input Layer

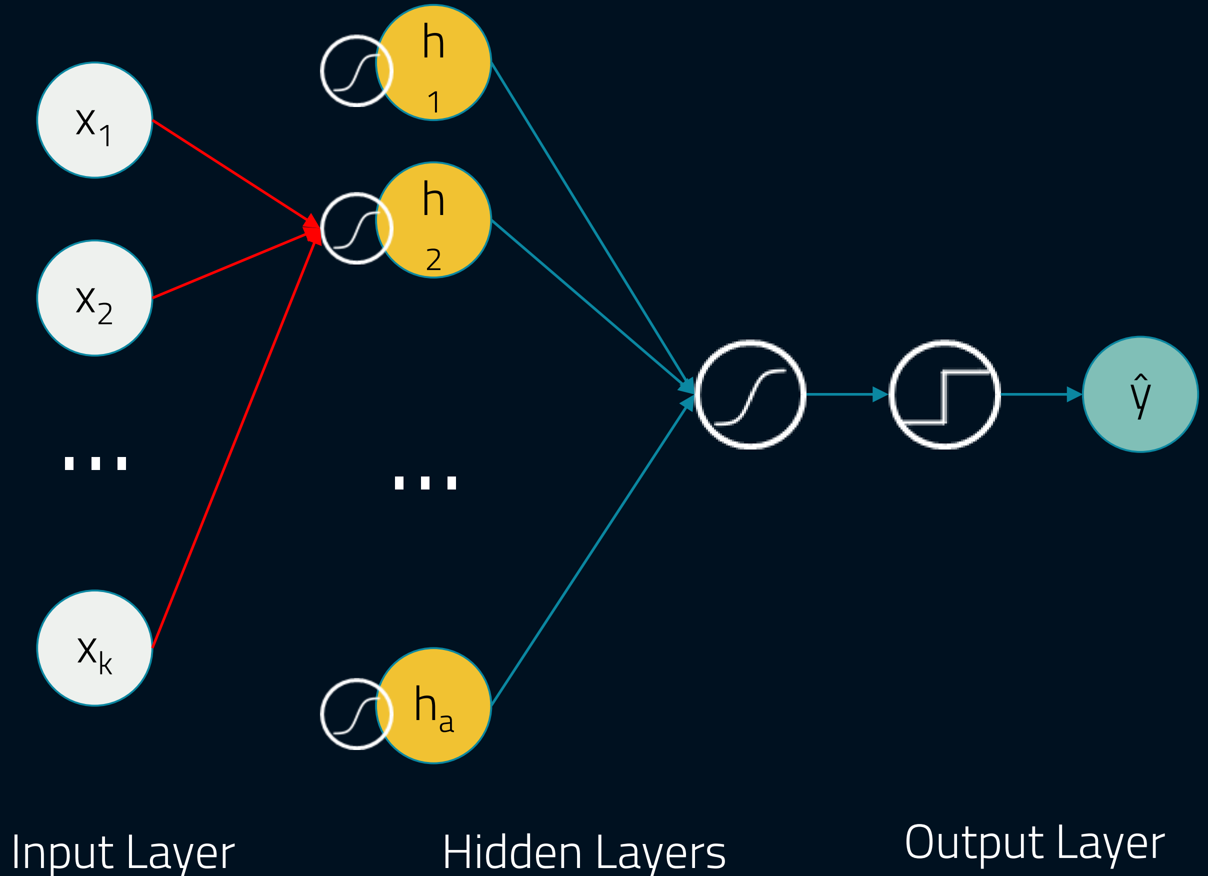
Hidden Layers

Output Layer

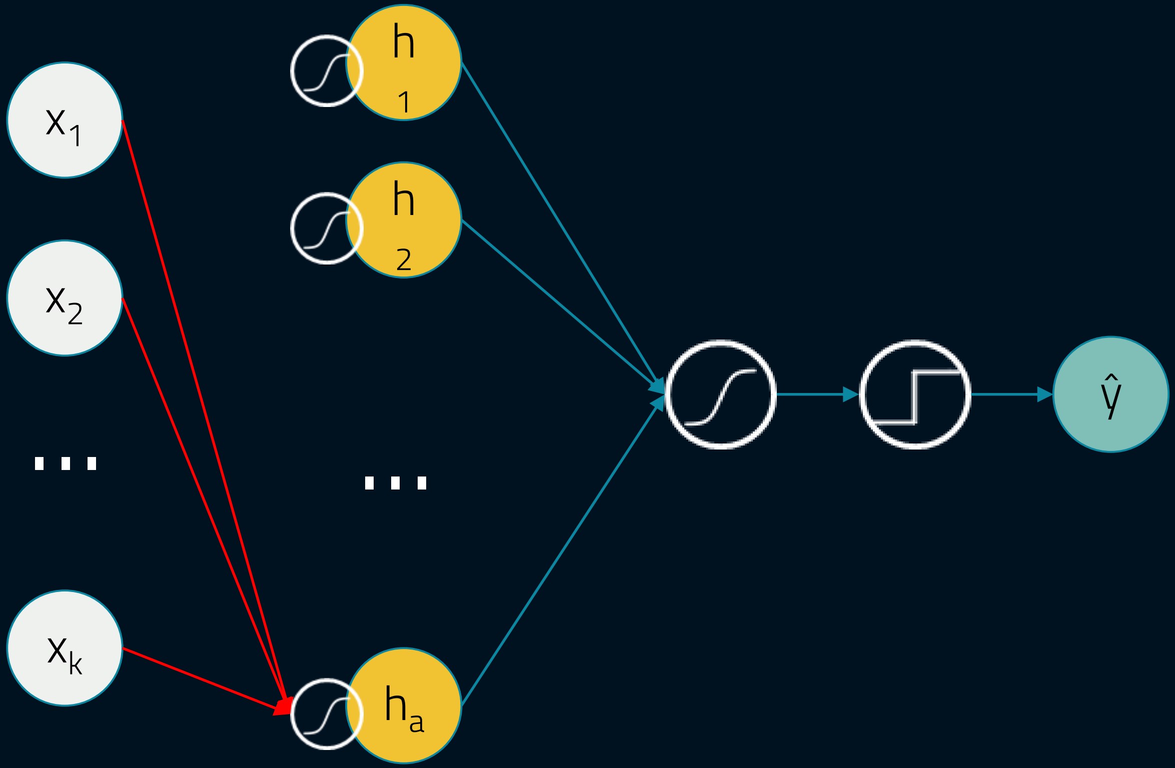
Neural Network With 1 Hidden Layer



Neural Network With 1 Hidden Layer



Neural Network With 1 Hidden Layer

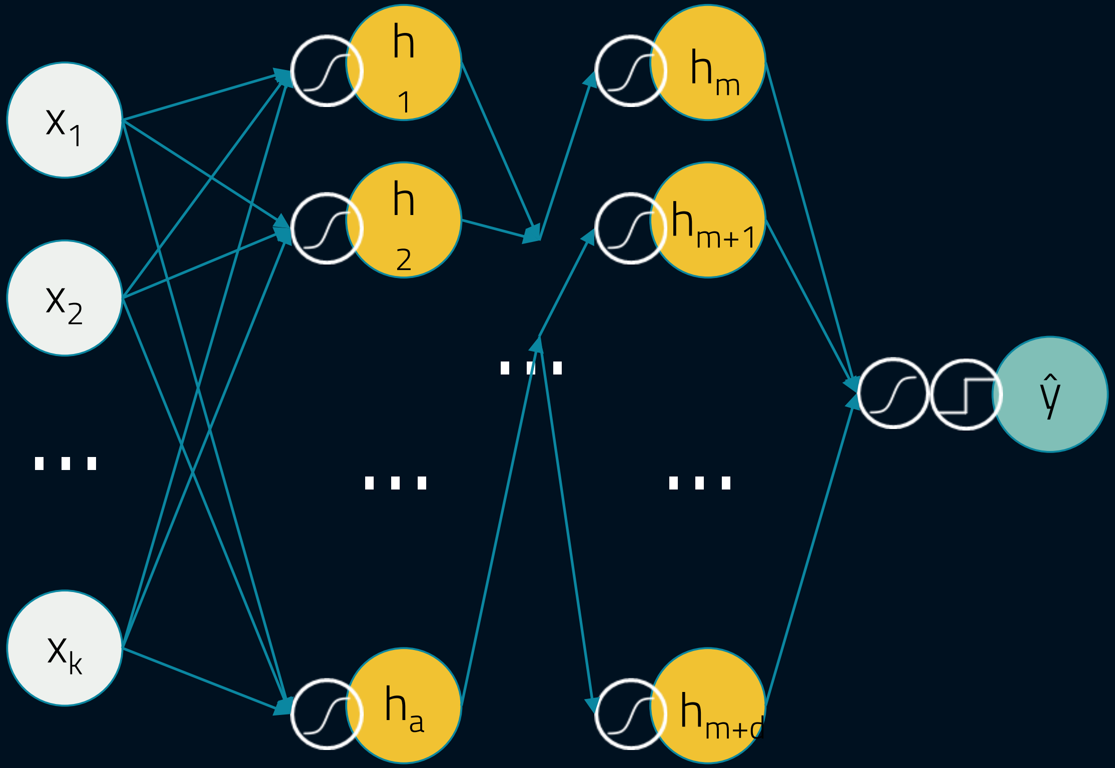


Input Layer

Hidden Layers

Output Layer

Generalized Neural Network



Input Layer

Hidden Layers

Output Layer

What's Left to Address

- How do we actually train neural networks?
  - Gradient descent
  - Backpropagation
- What is the utility of different activation functions?
- The importance of regularization for neural networks
- How to move to more than 2 categories for our prediction
- A short look at commonly used models:
  - Convolutional Neural Networks
  - Recurrent Neural Networks

Likelihood Estimate (In Detail)

- For a given datapoint i:
- The probability that we correctly predict y_i to be 1 when y_i is actually 1 is $p_i^{y_i}$
- The probability that we correctly predict y_i to be 0 when y_i is actually 0 is $(1 - p_i)^{1-y_i}$
- This makes sense because if y_i is 1, then we will get out p_i and if it is 0, we will get out $1-p_i$.
- The other factor will always be 1 & this is useful.

Likelihood Estimate cont.

- So our total likelihood estimate is

$$LE = \prod_{i=1}^{n} p_i^{y_i}(1 - p_i)^{1-y_i}$$

- Taking the log of this, we get the log-likelihood:

$$LLE = \sum_{i=1}^{n} y_i log(p_i) + (1 - y_i) log(1 - p_i)$$