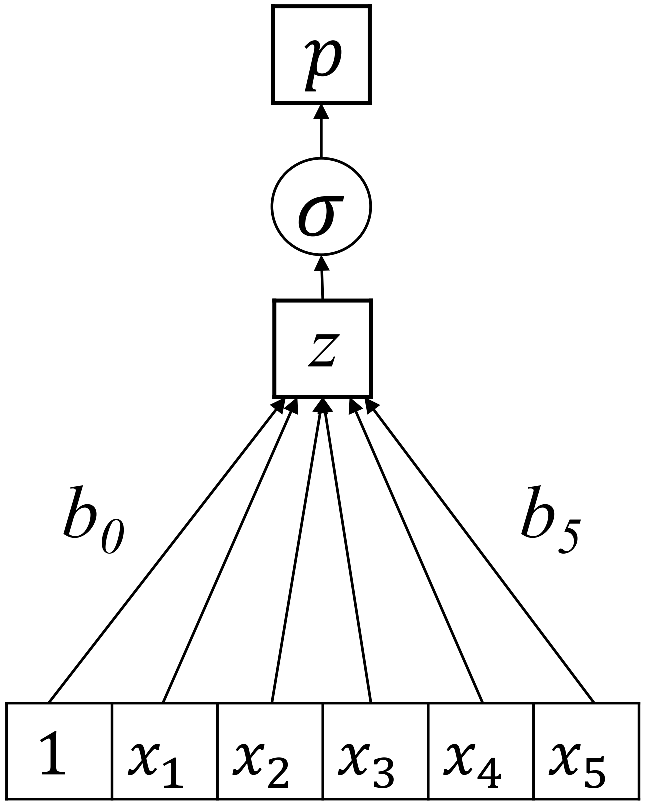
The Multilayer Perceptron

(in other words, a *standard* neural network)

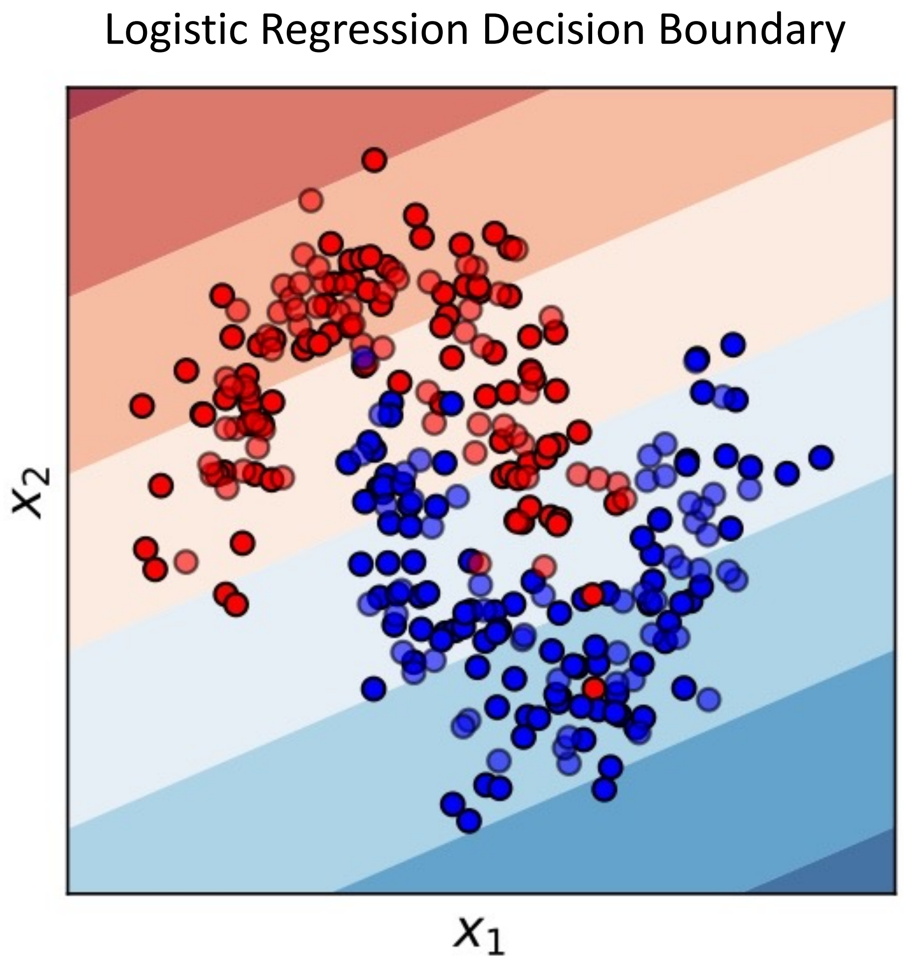
Matthew Engelhard

Today: How can we modify logistic regression to learn complex, nonlinear relationships?



We need more flexible, non-linear classifiers

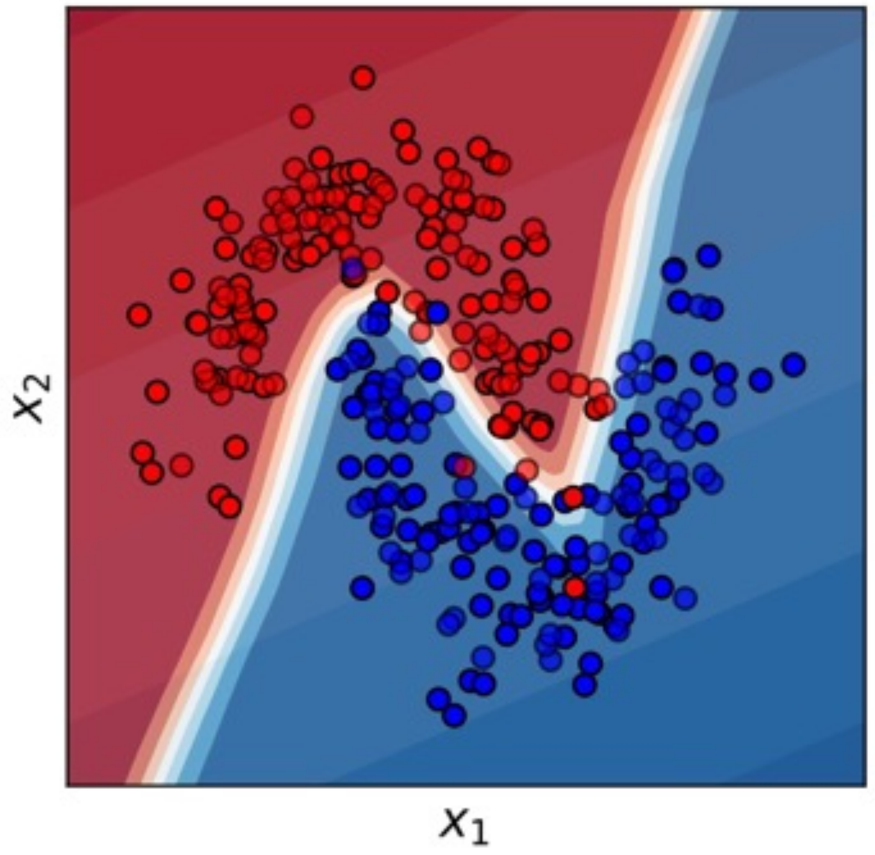
- Suppose x_1 and x_2 are biomarker values
- After biopsy:
  - Blue patients: benign
  - Red patients: malignant
- We need a model that can distinguish between the two, but logistic regression cannot: it can only draw **linear** decision boundaries.
- Today, we will see how we can "extend" logistic regression to form a multilayer perceptron (MLP) – in other words, a neural network – that can draw **nonlinear** decision boundaries



We need more flexible, non-linear classifiers

- Suppose x_1 and x_2 are biomarker values
- After biopsy:
  - Blue patients: benign
  - Red patients: malignant
- We need a model that can distinguish between the two, but logistic regression cannot: <u>it can only draw **linear** decision boundaries</u>.
- Today, we will see how we can "extend" logistic regression to form a multilayer perceptron (MLP) – in other words, a neural network – that can draw **nonlinear** decision boundaries

MLP (i.e. neural network) decision boundary



Let’s break logistic regression.

Breaking logistic regression: simple example 1

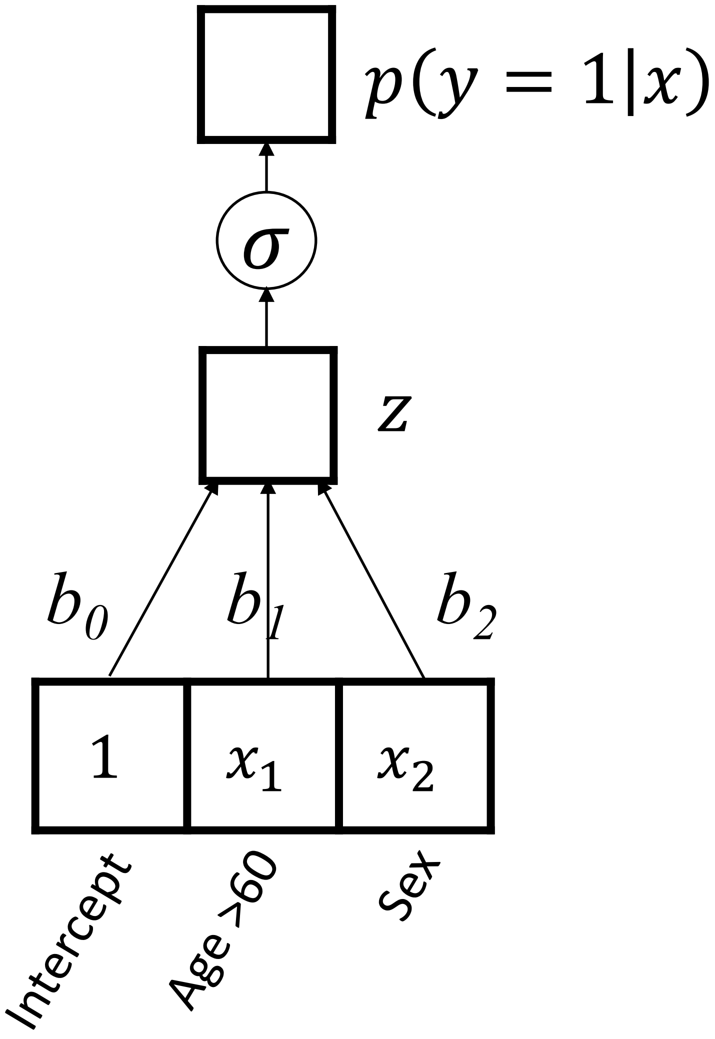
- Suppose there’s a new disease. We’re trying to develop a predictive model to determine who it will affect.

- For whatever reason, it turns out the disease affects only two groups of people:
  - Females under 60
  - Males over 60

- Can logistic regression learn to predict this?

Breaking logistic regression: simple example 1

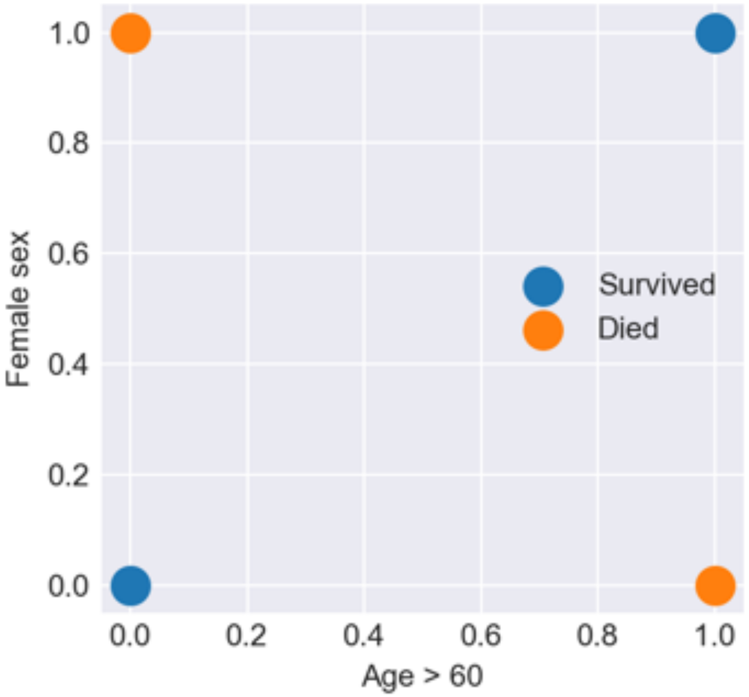
- Predictor 1: Age
  - $x_1 = 1$ if age > 60
  - $x_1 = 0$ if age $\leq$60
- Predictor 2: Sex
  - $x_2 = 1$ if female
  - $x_2 = 0$ if male
- Goal: predict high log-odds only for
  - Females under 60
  - Males over 60
- What should the parameters (b_1, b_2, and b_3) be?



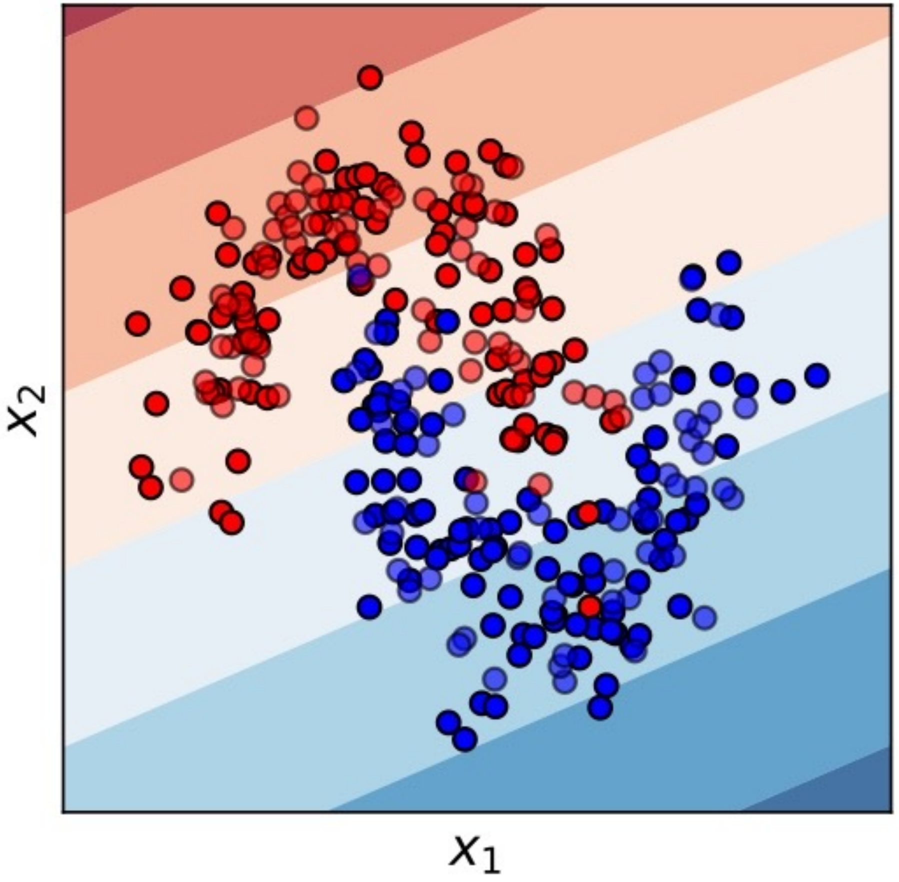
Breaking logistic regression: simple example 1

Can we solve this with a linear decision boundary?

In other words, **can we draw a line separating those who lived from those who died?**



Example of a Logistic Regression Decision Boundary



Breaking logistic regression: simple example 2

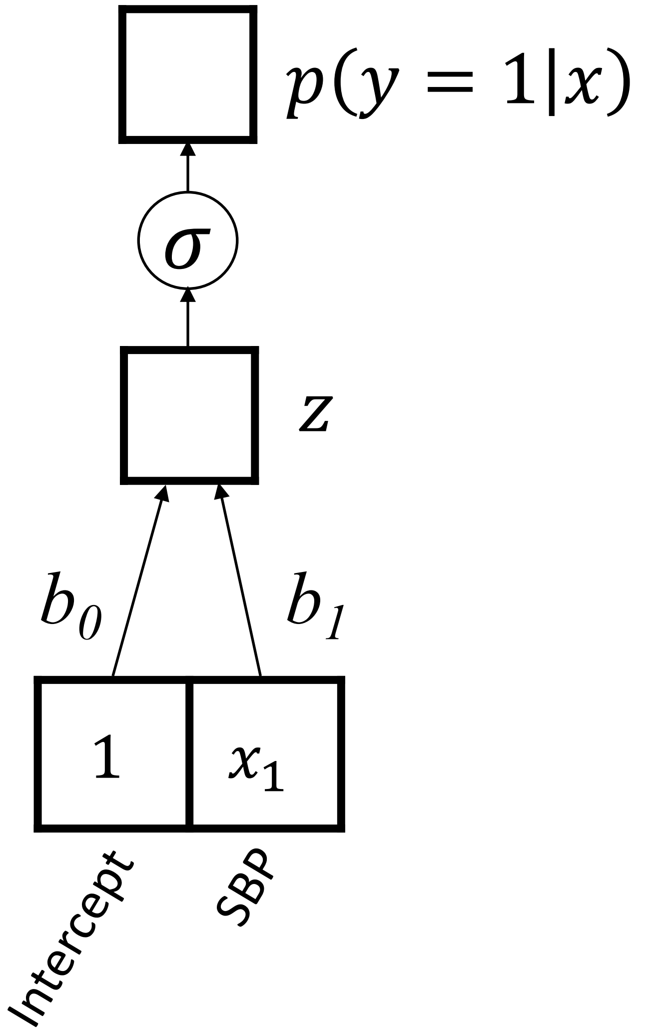
- In the ED, we're trying to develop a model that predicts mortality from systolic blood pressure upon arrival.

- It turns out, you're at high risk of dying if you:
  a) You have very <u>high</u> blood pressure, OR
  b) You have very <u>low</u> blood pressure

- Can logistic regression learn to predict this?

Breaking logistic regression: simple example 2

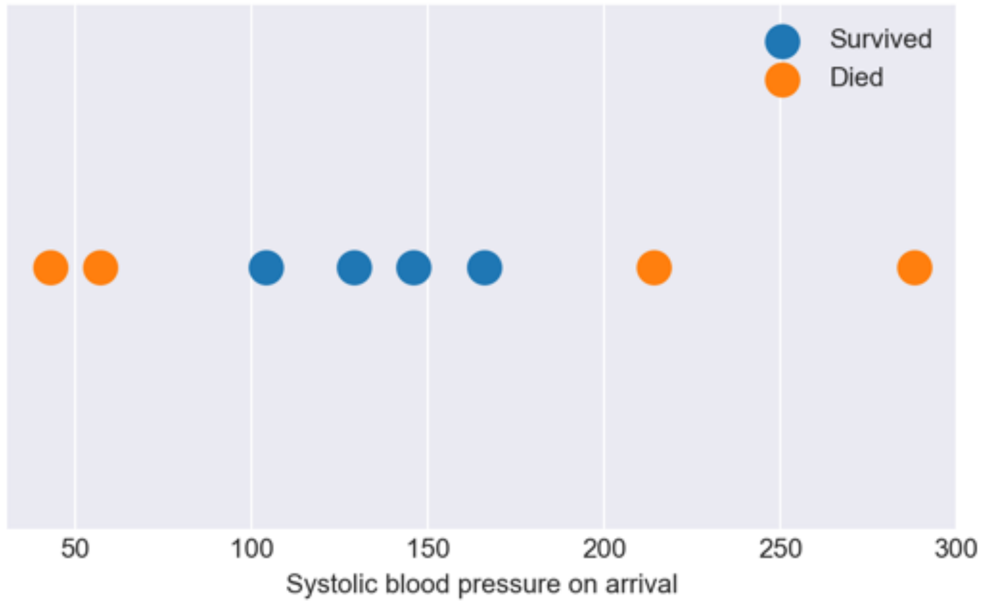
- Predictor 1: Systolic blood pressure
- Goal: predict high log-odds only for
  - SBP > 200
  - SBP < 60
- What should the parameters (b_1 and b_2) be?



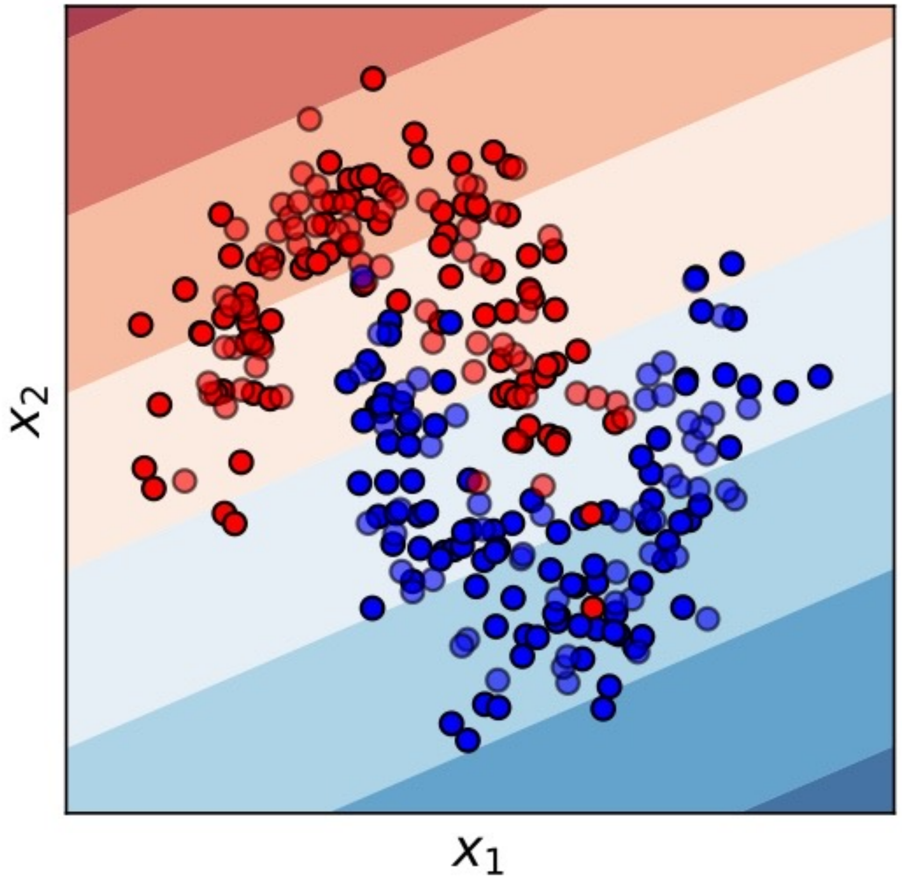
Breaking logistic regression: simple example 2

Can we solve this with a linear decision boundary?

In other words, **can we draw a line separating those who lived from those who died?**



Example of a Logistic Regression Decision Boundary



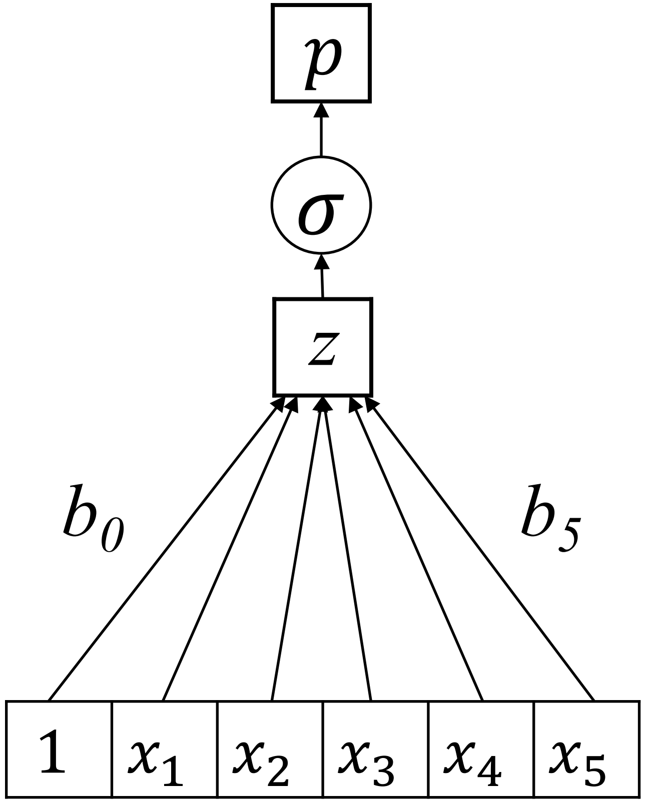
In general, there can be:

- Nonlinear affects (e.g. *high* and *low* blood pressure both increase risk; middle/normal blood pressure is OK)

- Interactions: males over 60 are at risk, and females under 60 are at risk, but being male (or female) does not on its own increase risk

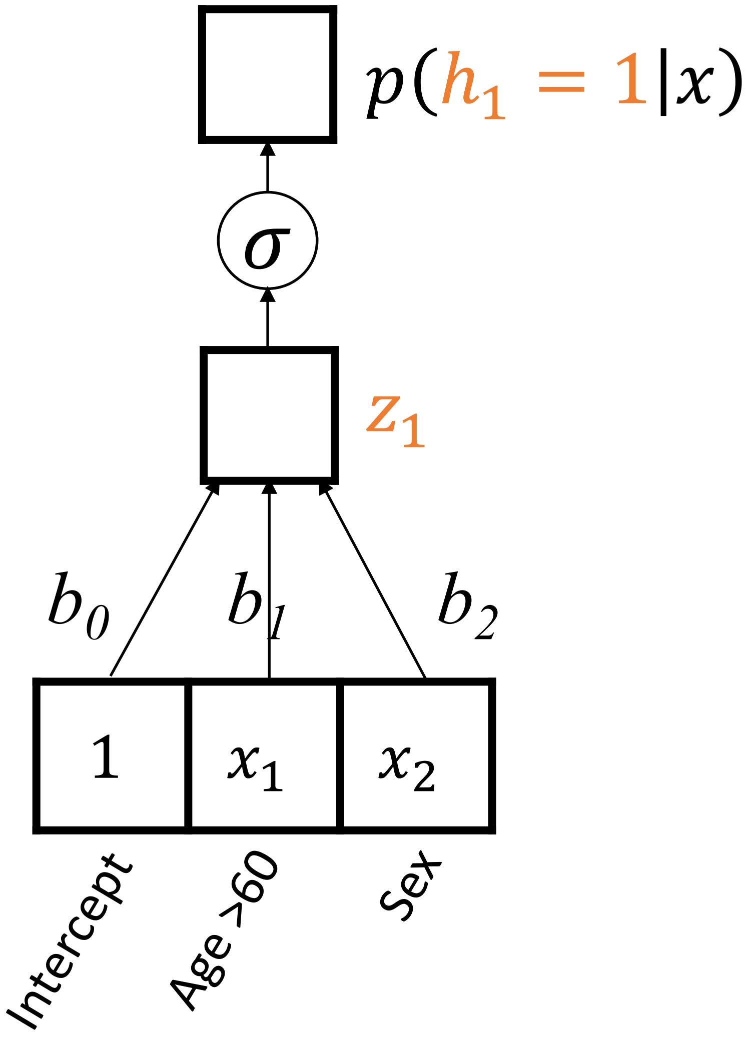
- We need models that can figure this stuff out… but how?

How can we modify logistic regression to learn complex, nonlinear relationships?



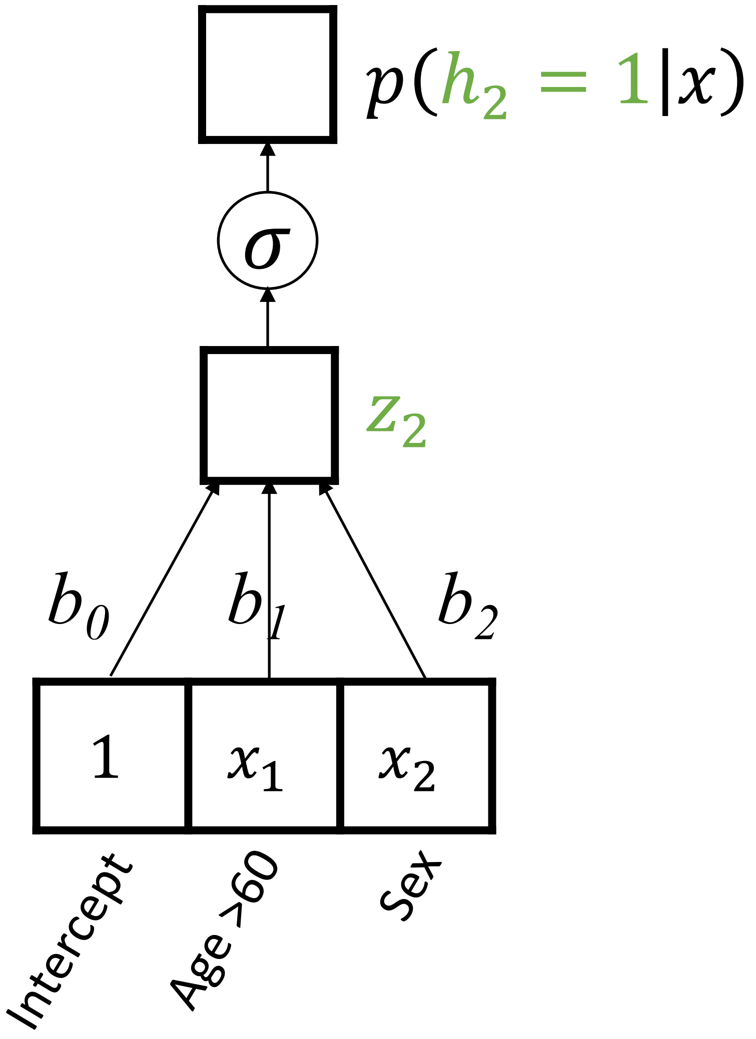
Let’s break the problem into simpler pieces.

- Predictor 1: Age
  - $x_1 = 1$ if age > 60
  - $x_1 = 0$ if age ≤60
- Predictor 2: Sex
  - $x_2 = 1$ if female
  - $x_2 = 0$ if male
- Can logistic regression identify
  - Females under 60 (h_1=1)?
  - Males over 60 (h_2=1)?
- What should the parameters (b_1, b_2, and b_3) be?



Let’s break the problem into simpler pieces.

- Predictor 1: Age
  - $x_1 = 1$ if age > 60
  - $x_1 = 0$ if age ≤60
- Predictor 2: Sex
  - $x_2 = 1$ if female
  - $x_2 = 0$ if male
- Can logistic regression identify
  - Females under 60 (h_1=1)?
  - Males over 60 (h_2=1)?
- What should the parameters (b_1, b_2, and b_3) be?

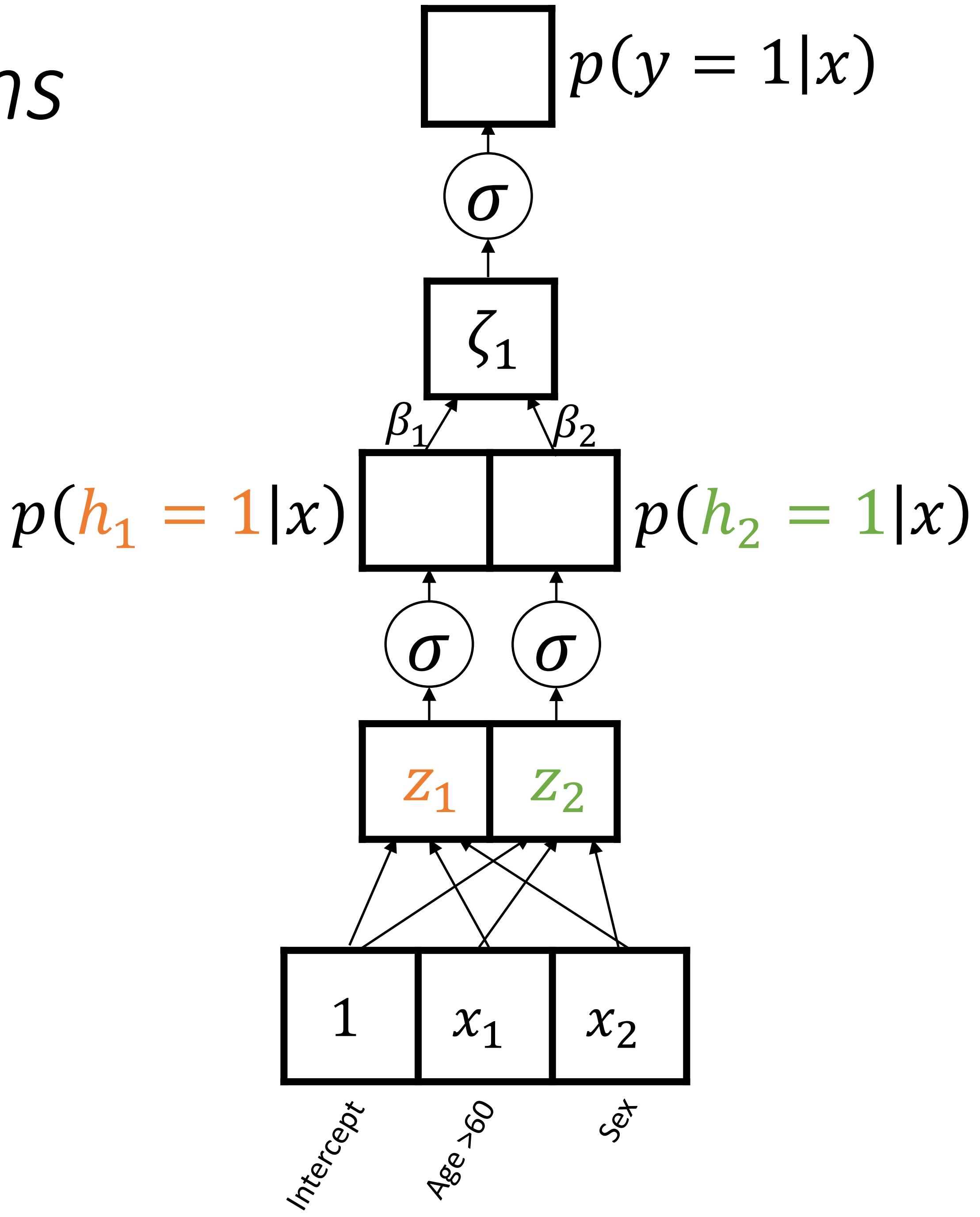


Now, *use these predictions* to make predictions

- Neuron 1 (h_1):
  - $h_1 = 1$ if (female ≤60)
  - $h_1 = 0$ otherwise

- Neuron 2 (h_2):
  - $h_2 = 1$ if (male >60)
  - $h_2 = 0$ otherwise

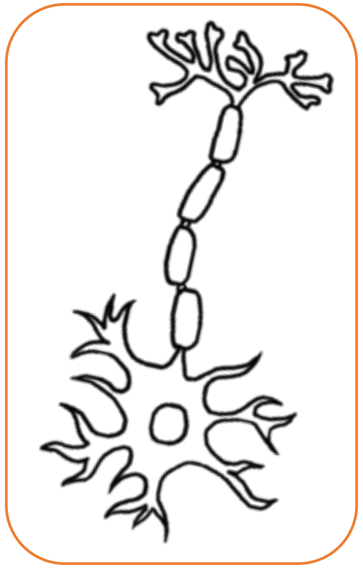
- What should the parameters (β_1 and β_2) be?



This is a neural network, or MLP.

- Each logistic regression is like a neuron

Neuron 1 (h_1):
Detects females under 60

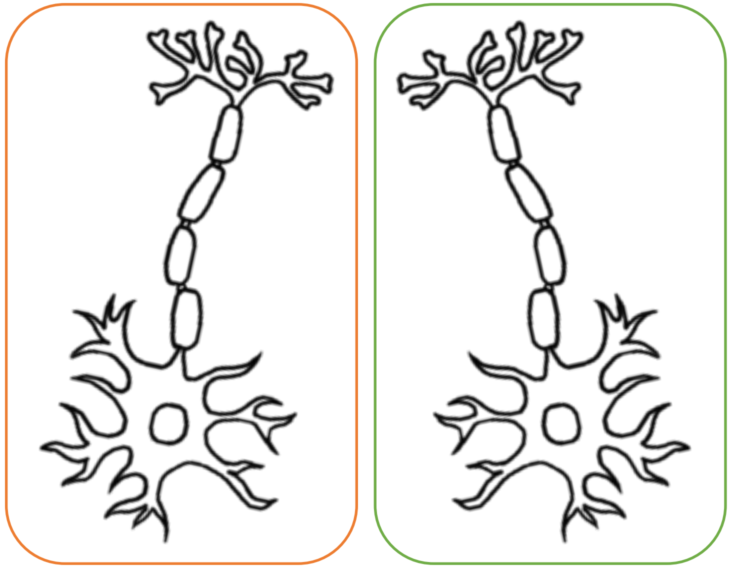


This is a neural network, or MLP.

- Each logistic regression is like a neuron

- Different neurons detect different *features*
  - *Feature 1*: female under 60
  - *Feature 2*: male over 60

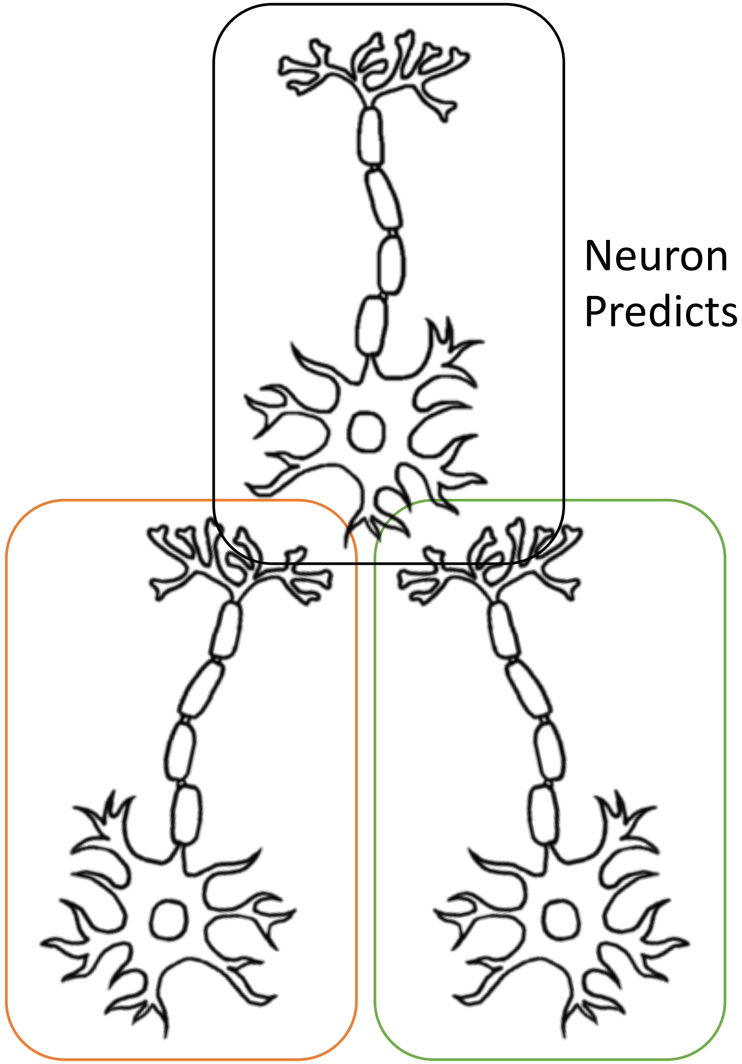
Neuron 1 (h_1):
Detects females under 60



Neuron 2 (h_2):
Detects males over 60

This is a neural network, or MLP.

- Each logistic regression is like a neuron
- Different neurons detect different *features*
  - *Feature 1*: female under 60
  - *Feature 2*: male over 60
- Predictions are made based on detected features rather than the original predictors



Neuron 3:
Predicts the outcome

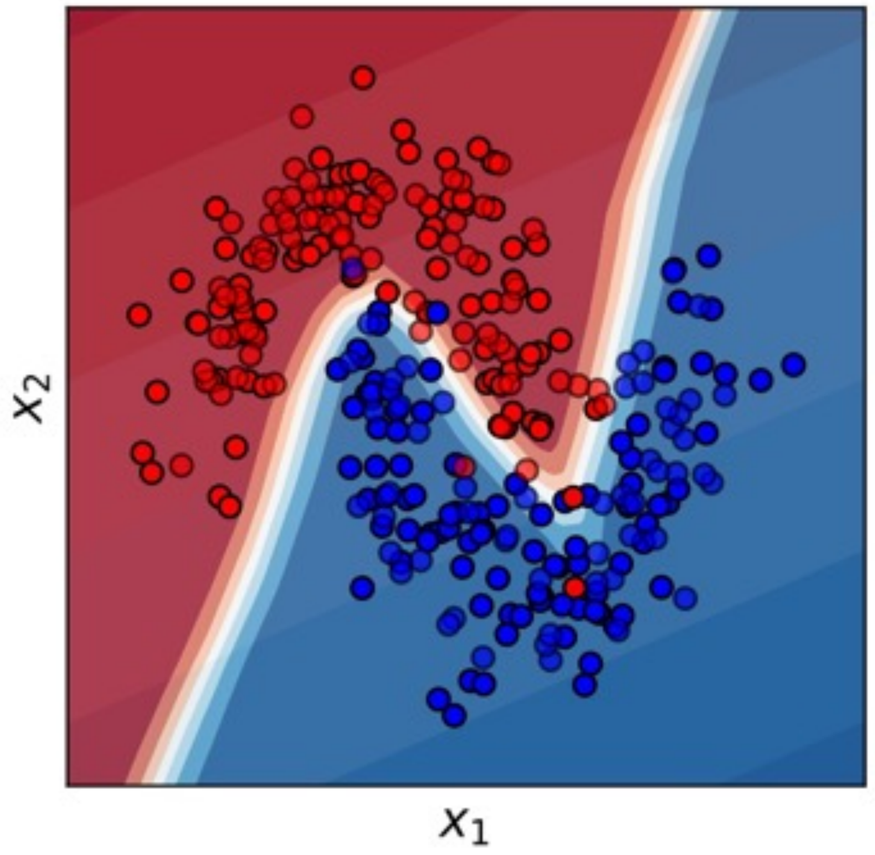
Neuron 1 (h_1):
Detects females under 60

Neuron 2 (h_2):
Detects males over 60

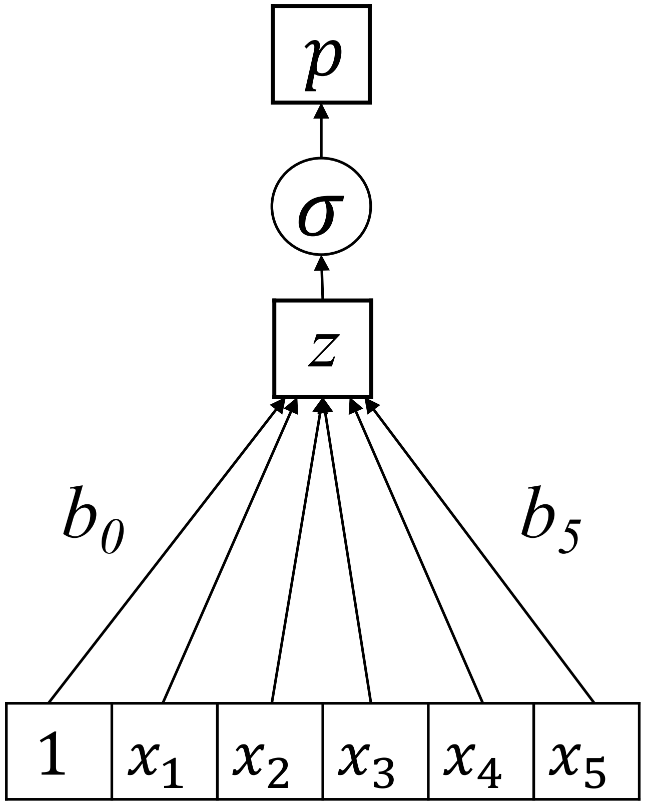
Neural networks provide unlimited flexibility.

- Become more flexible by:
  - Adding more layers of feature detectors
  - Adding more feature detectors per layer

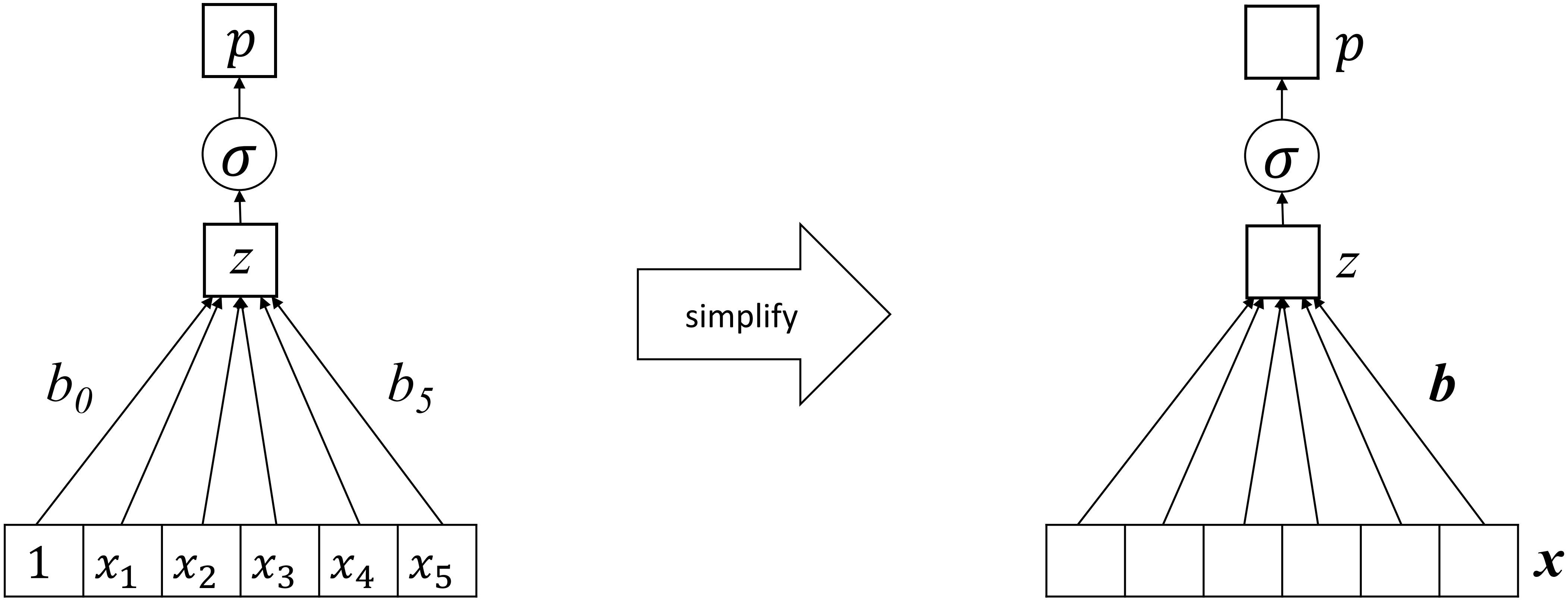
MLP (i.e. neural network) decision boundary

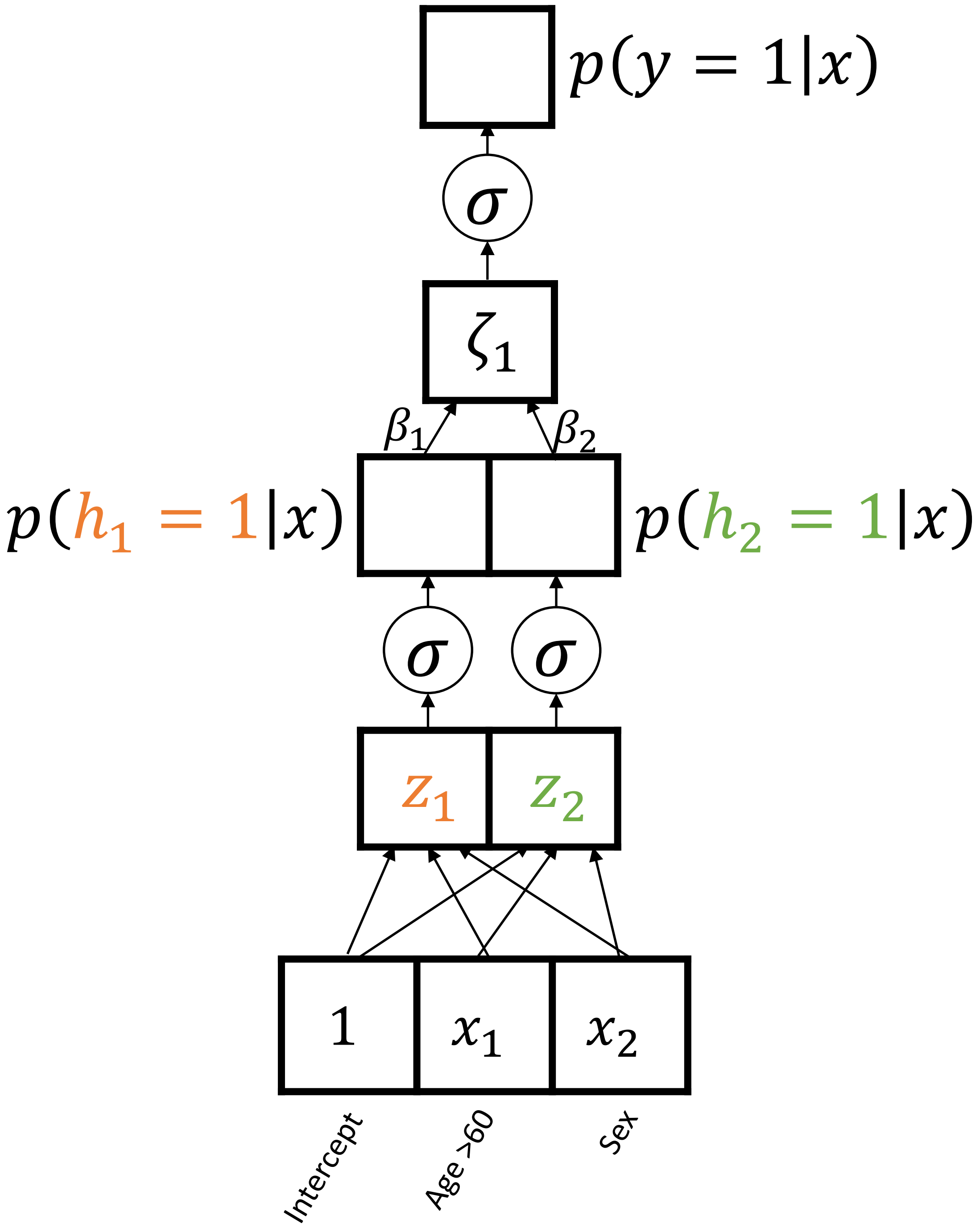


To show multiple layers of neurons, we need to simplify our logistic regression graph.

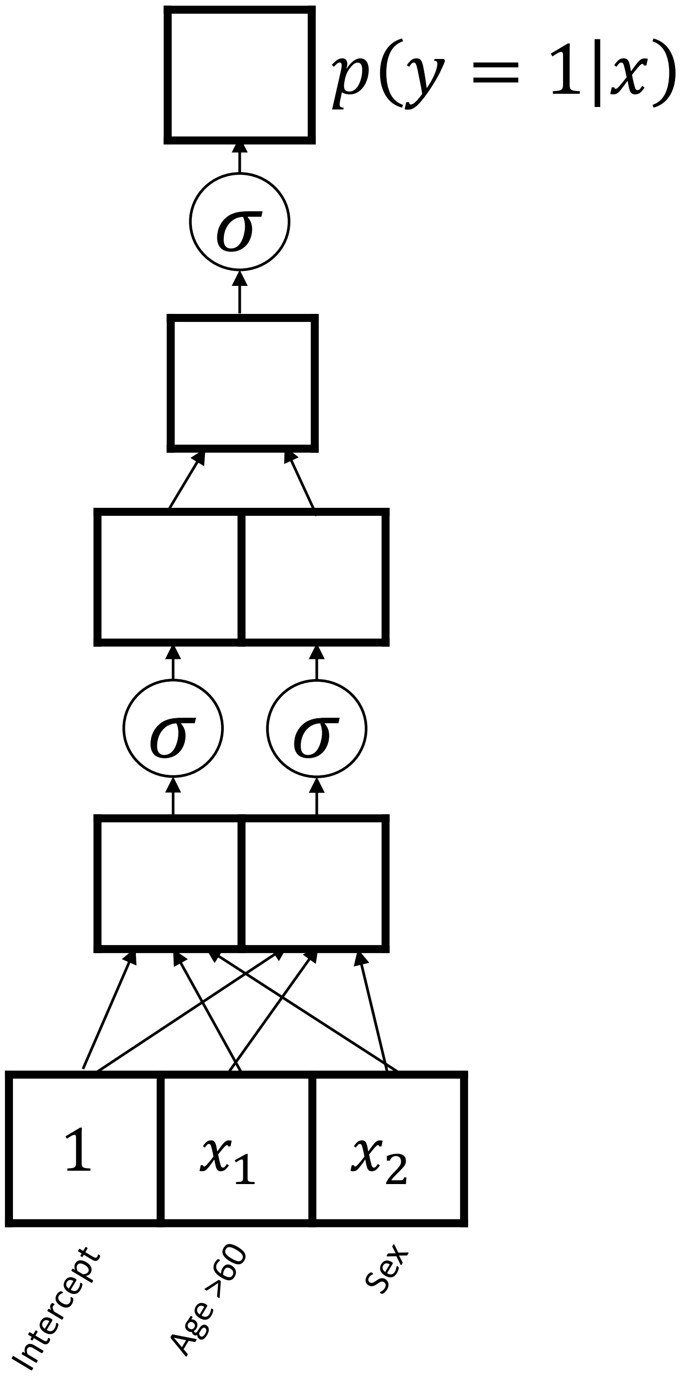


To show multiple layers of neurons, we need to simplify our logistic regression graph.



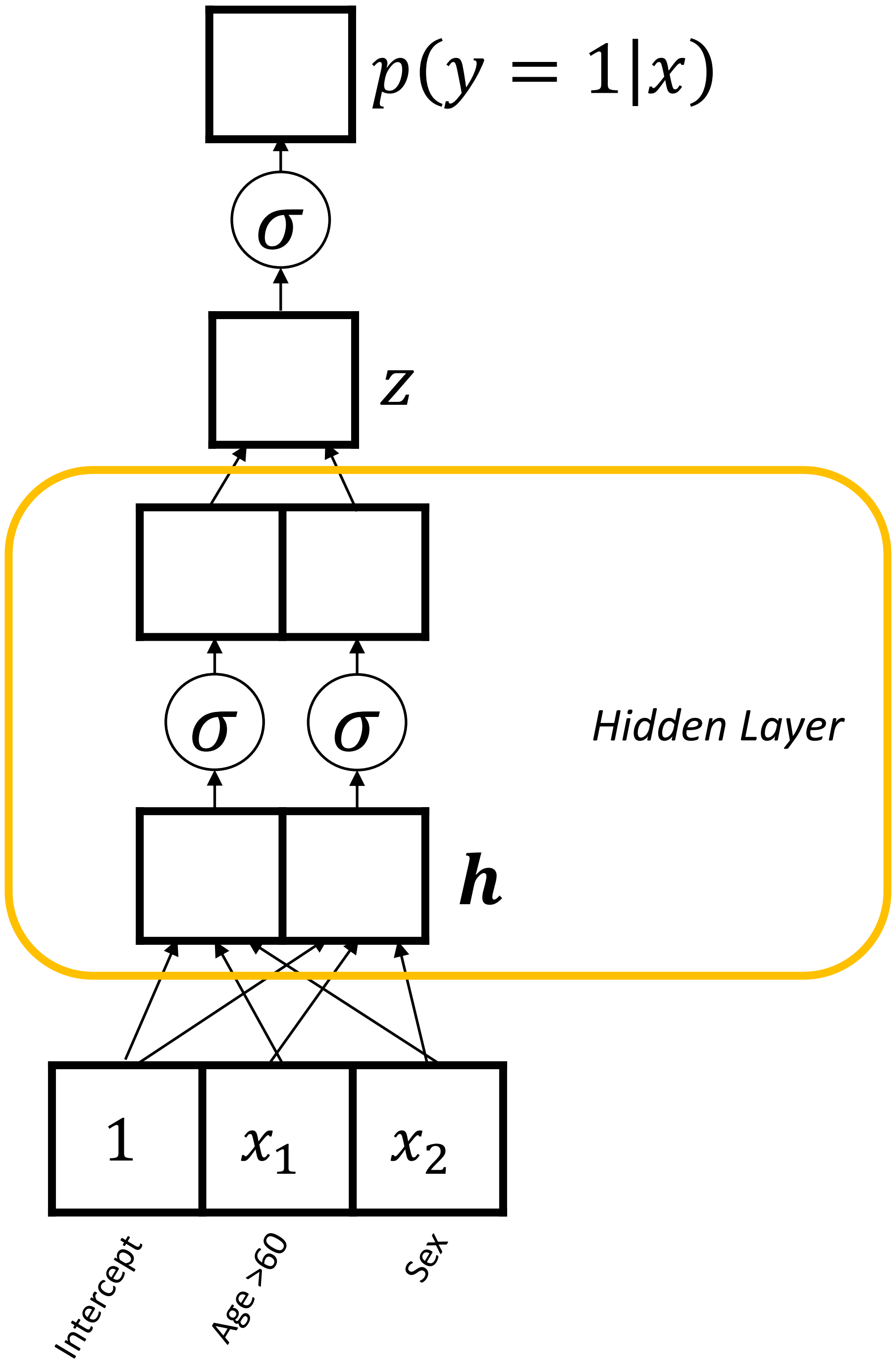


Let’s clean this up.



Let’s clean this up.

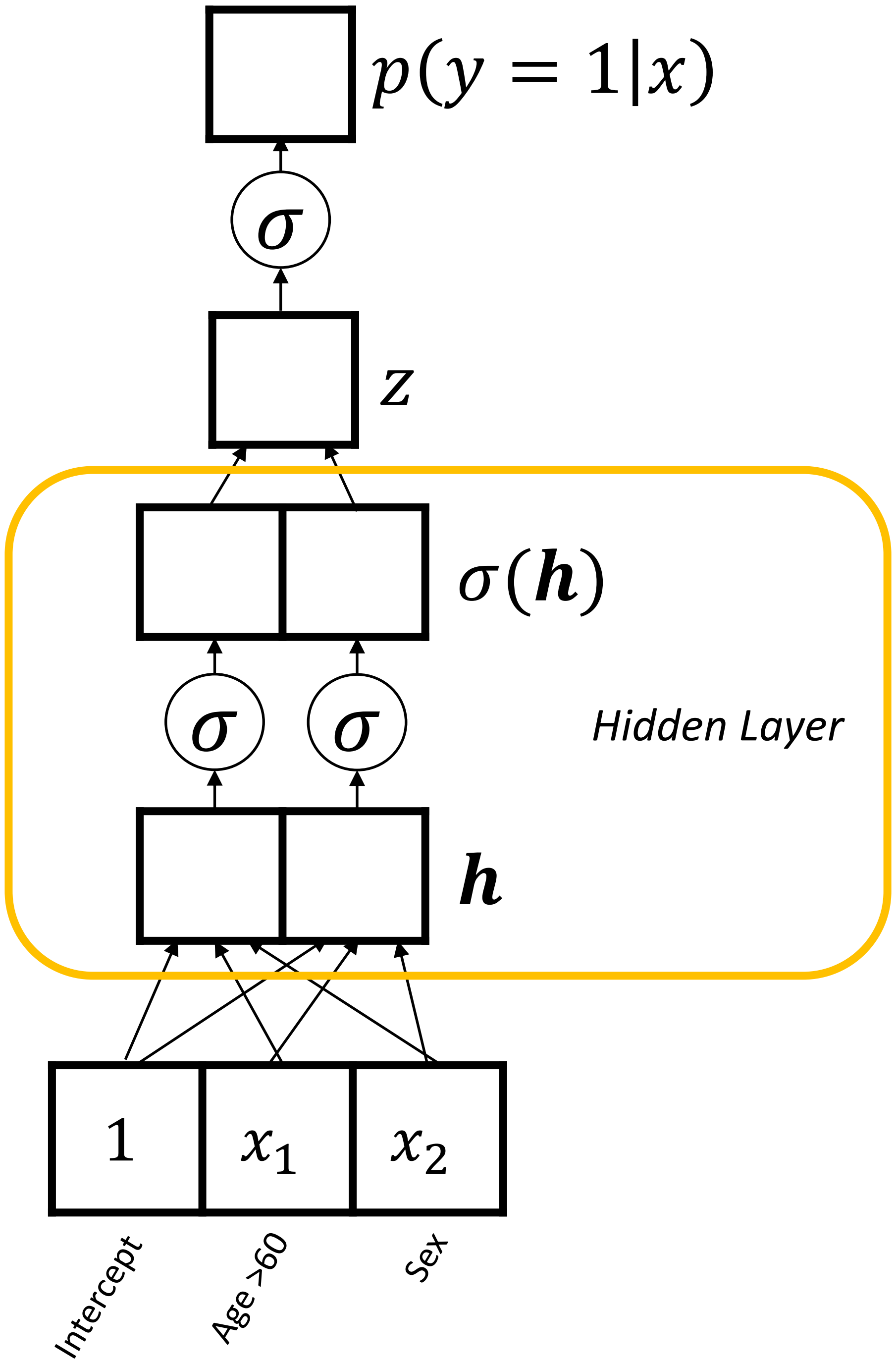
We’ll work toward standard NN/MLP terminology



Let’s clean this up.

We’ll work toward standard NN/MLP terminology

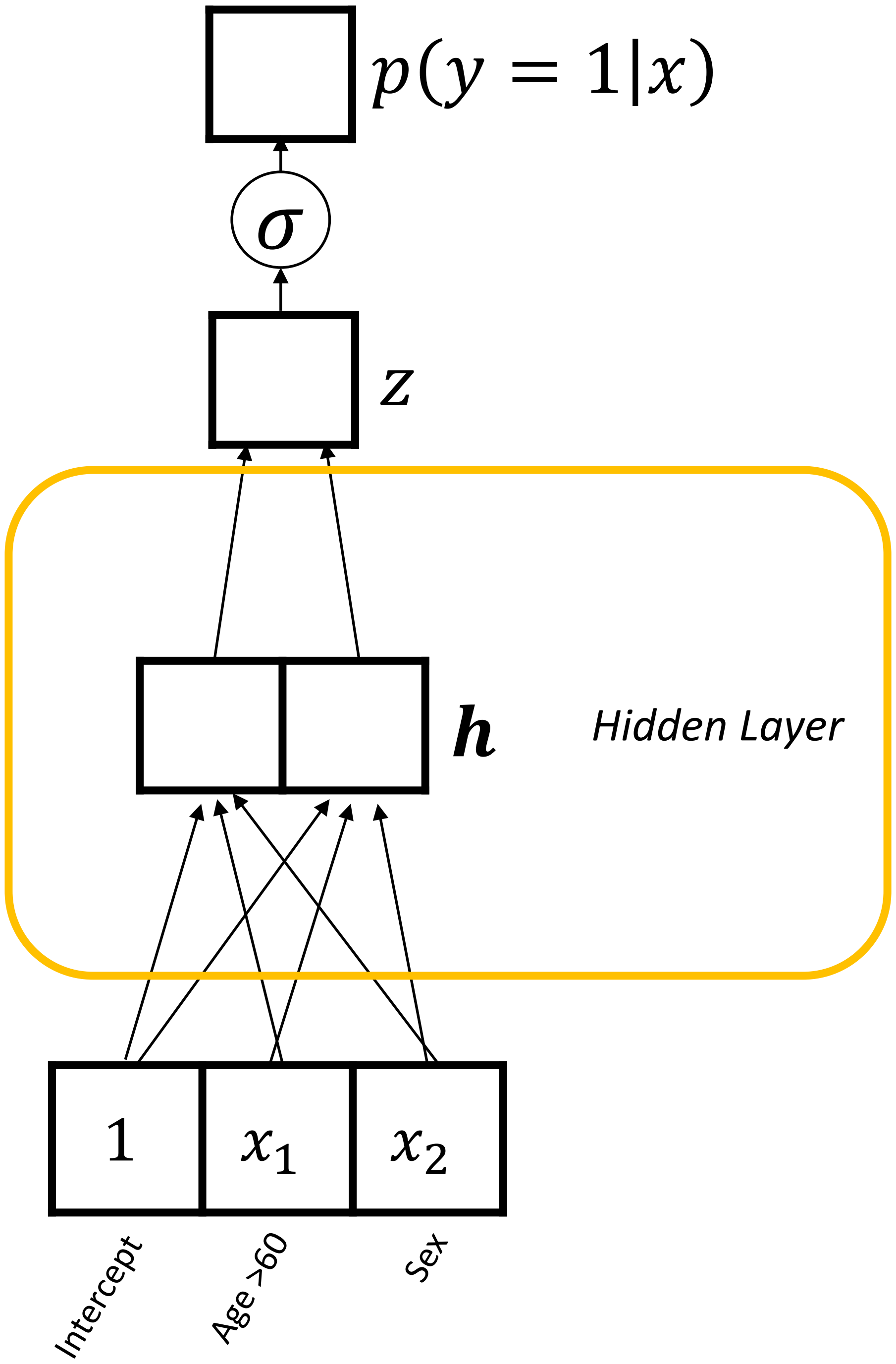
- *z*: the predicted log-odds of *y*
- ***h:*** the predicted log-odds of features in the hidden layer



Let’s clean this up.

We’ll work toward standard NN/MLP terminology

- *z*: the predicted log-odds of *y*
- ***h**:* the predicted log-odds of features in the hidden layer
- $\sigma(\boldsymbol{h})$: the predicted probabilities of features in the hidden layer

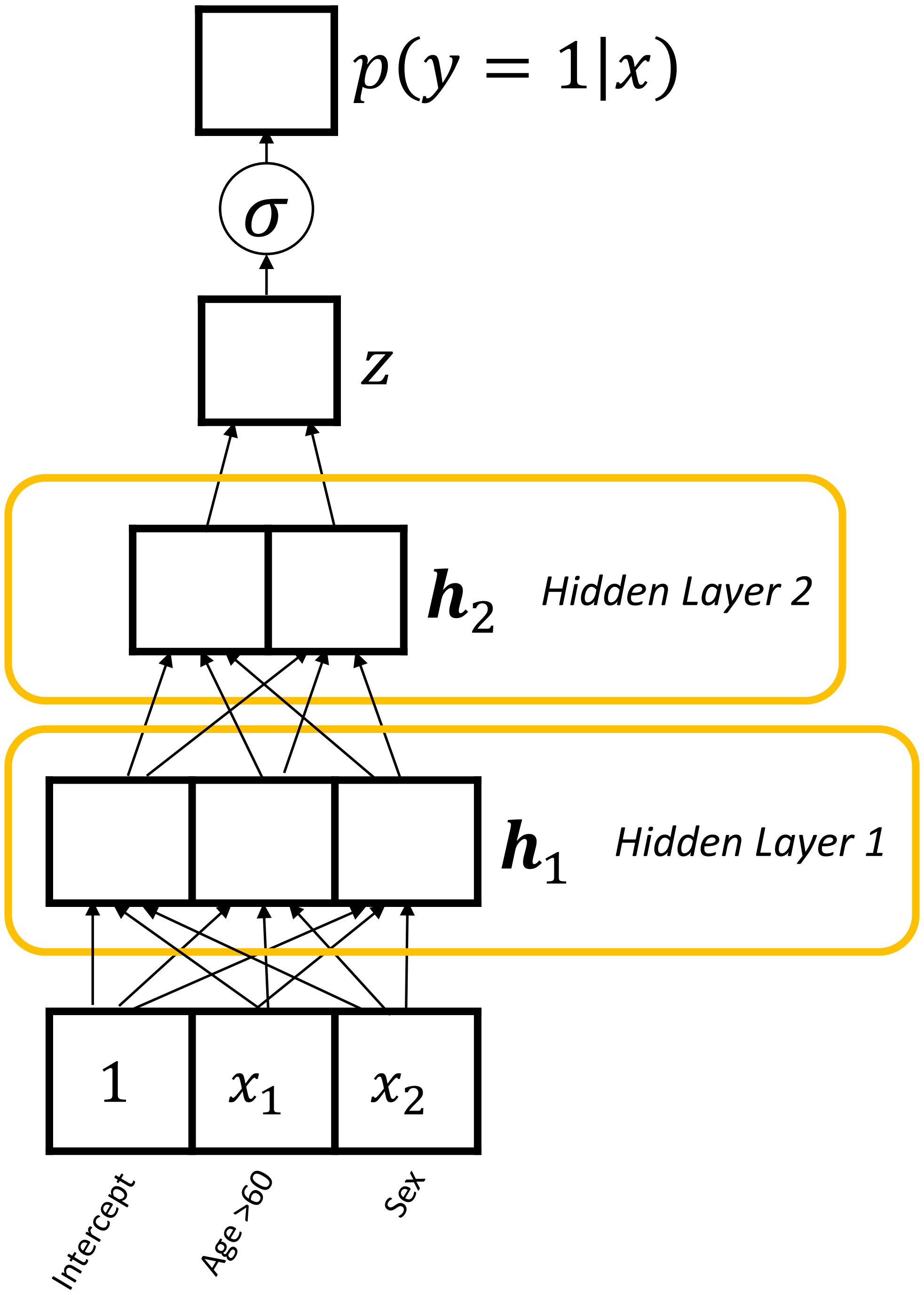


Let’s clean this up.

We’ll work toward standard NN/MLP terminology

- *z*: the predicted log-odds of *y*
- ***h****:* the predicted log-odds of features in the hidden layer
- $\sigma(\boldsymbol{h})$: the predicted probabilities of features in the hidden layer

- Sometimes we make the diagram smaller / neater by condensing our representation of the hidden layer



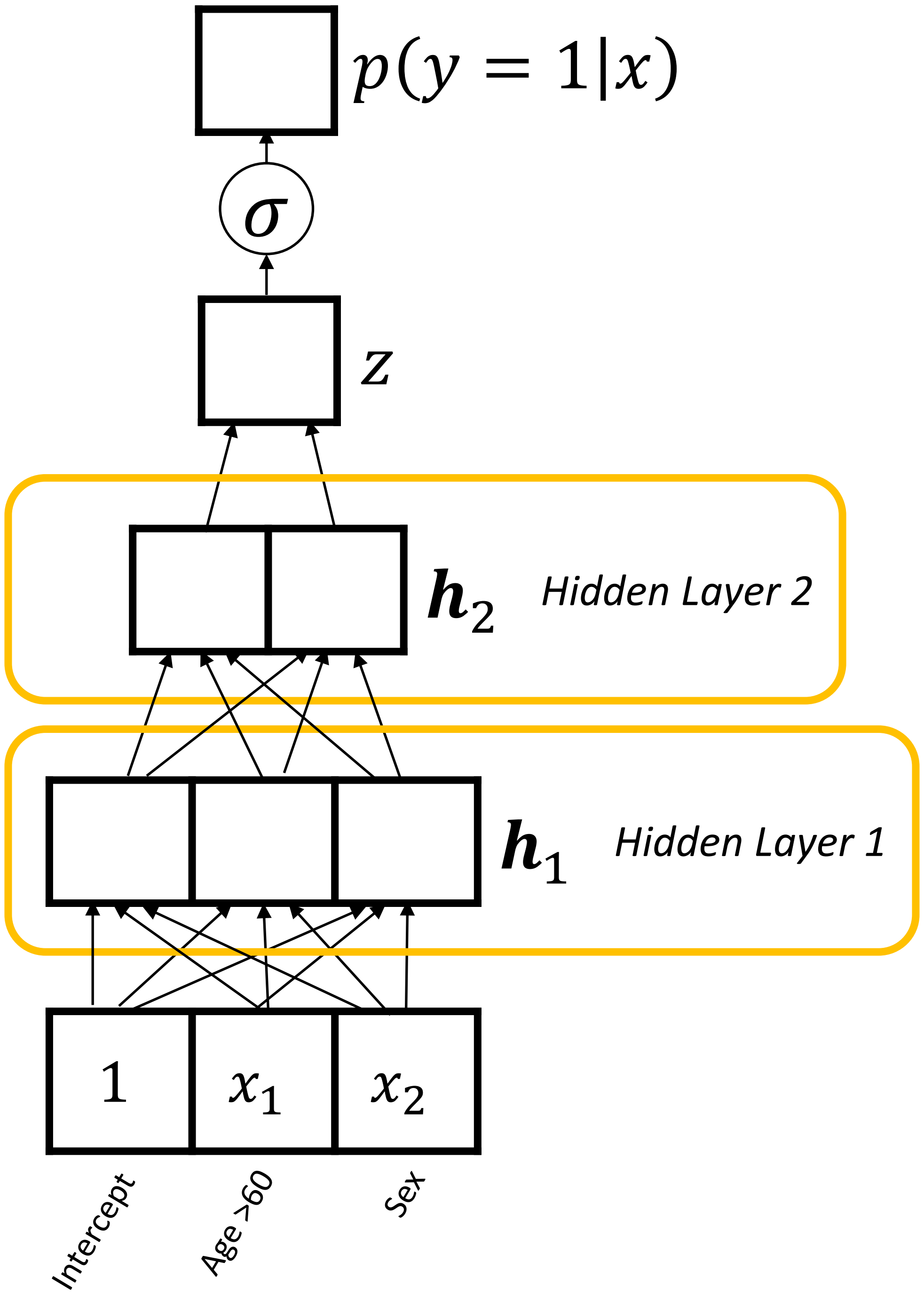
Let’s clean this up.

We’ll work toward standard NN/MLP terminology

- z: the predicted log-odds of y
- $\boldsymbol{h}$*:* the predicted log-odds of features in the hidden layer
- $\sigma(\boldsymbol{h})$: the predicted probabilities of features in the hidden layer

- Sometimes we make the diagram smaller / neater by condensing our representation of the hidden layer

- This makes it easier to show models with multiple hidden layers



Quiz:

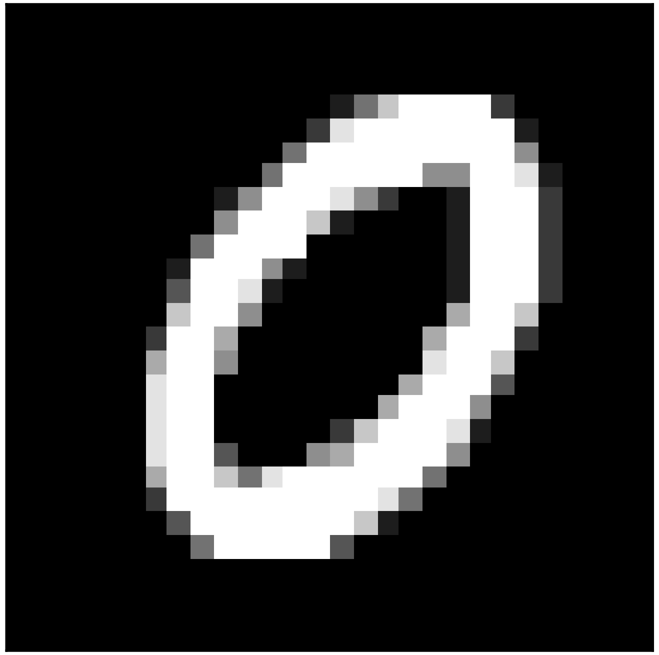
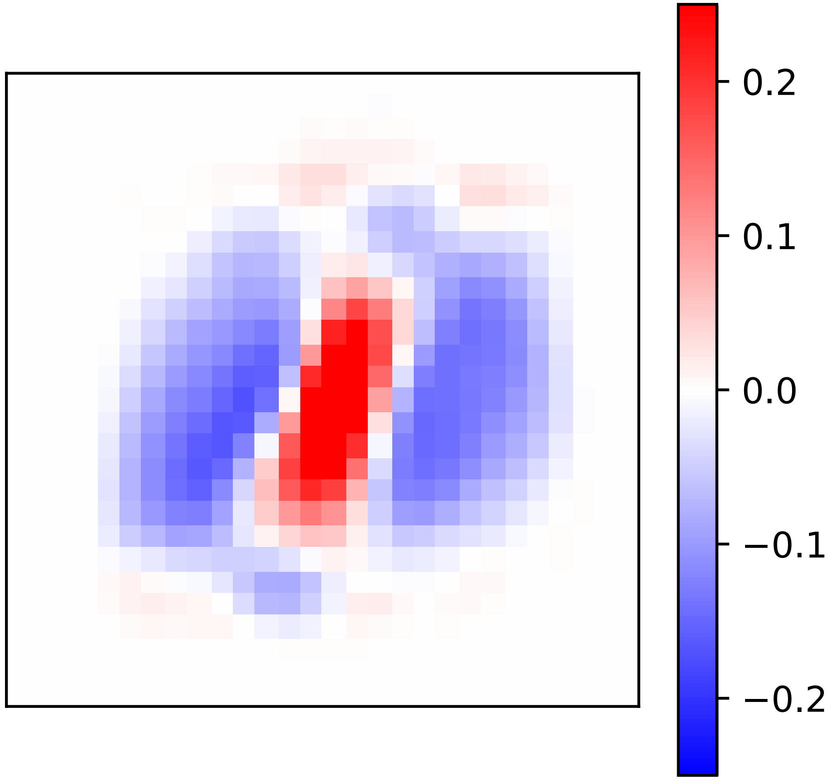
1. How many logistic regressions are in this neural network?

2. How many parameters are in this neural network?

MLP for MNIST

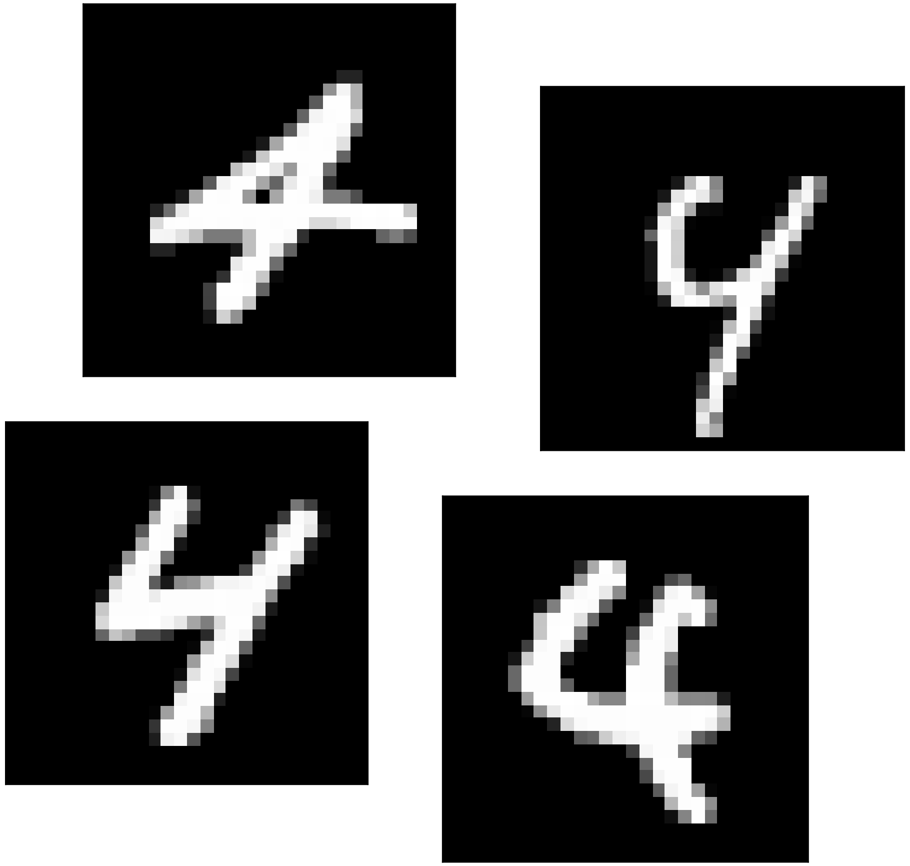
Moving toward computer vision

Why Limit Ourselves to Only One Filter?

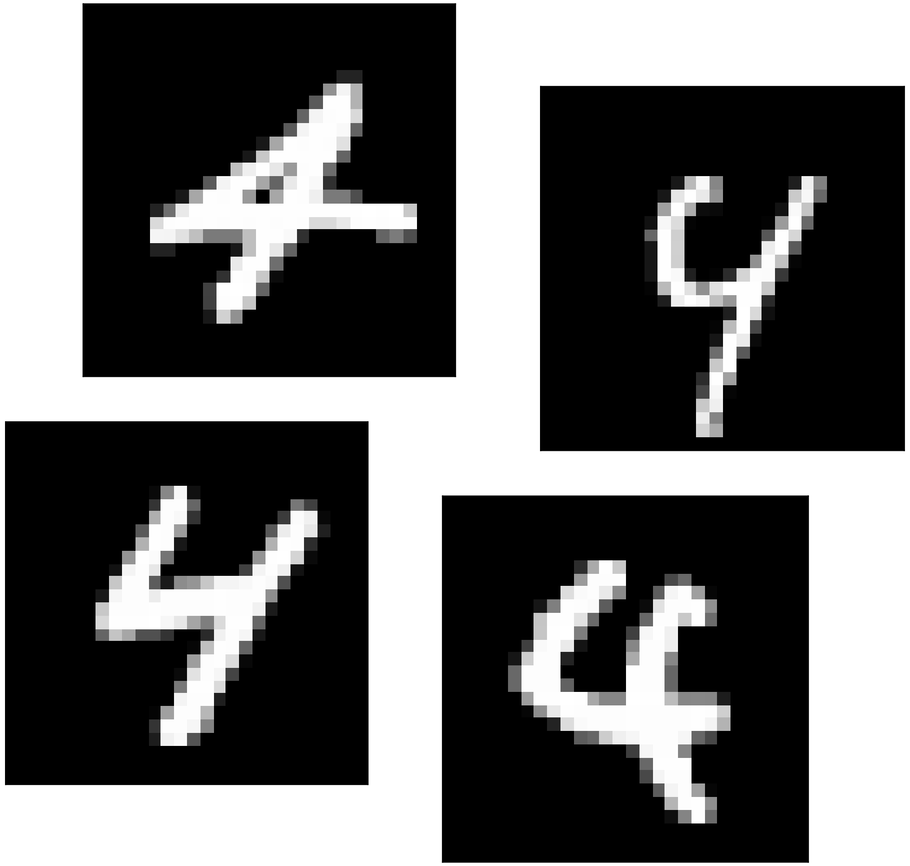
$\sigma($  $\times$ 

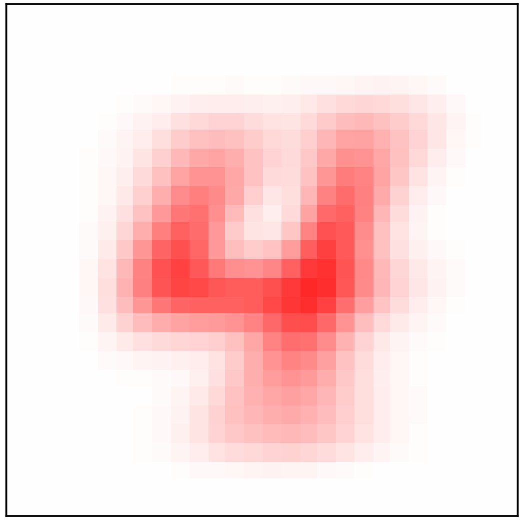
$)$

Return to MNIST: Many ways of writing "4"



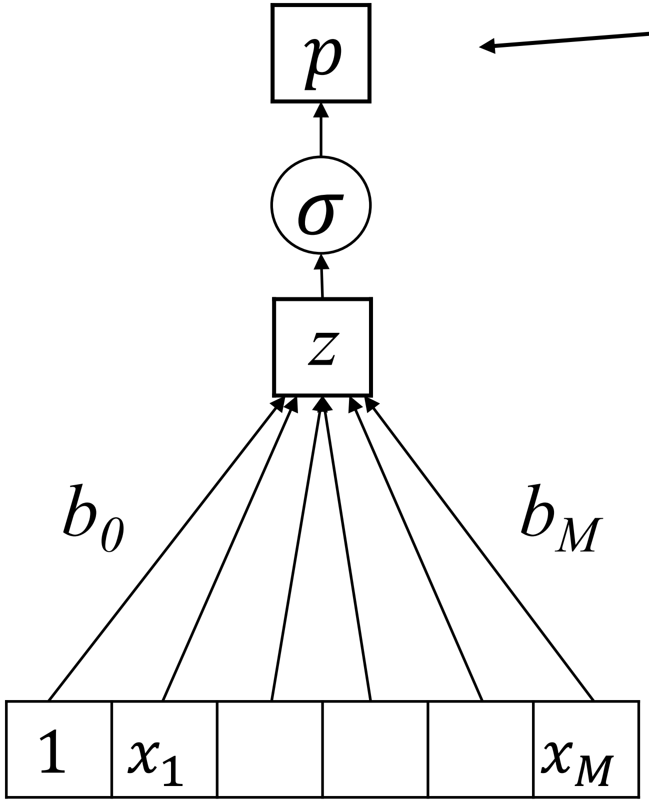
Return to MNIST: Many ways of writing "4"



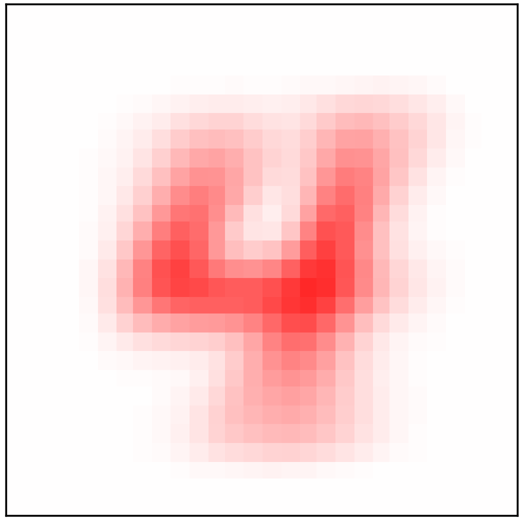


Single Filter (e.g. Logistic Regression/ "Shallow Learning") only uses one filter, looks for the average shape

A single '4' detector

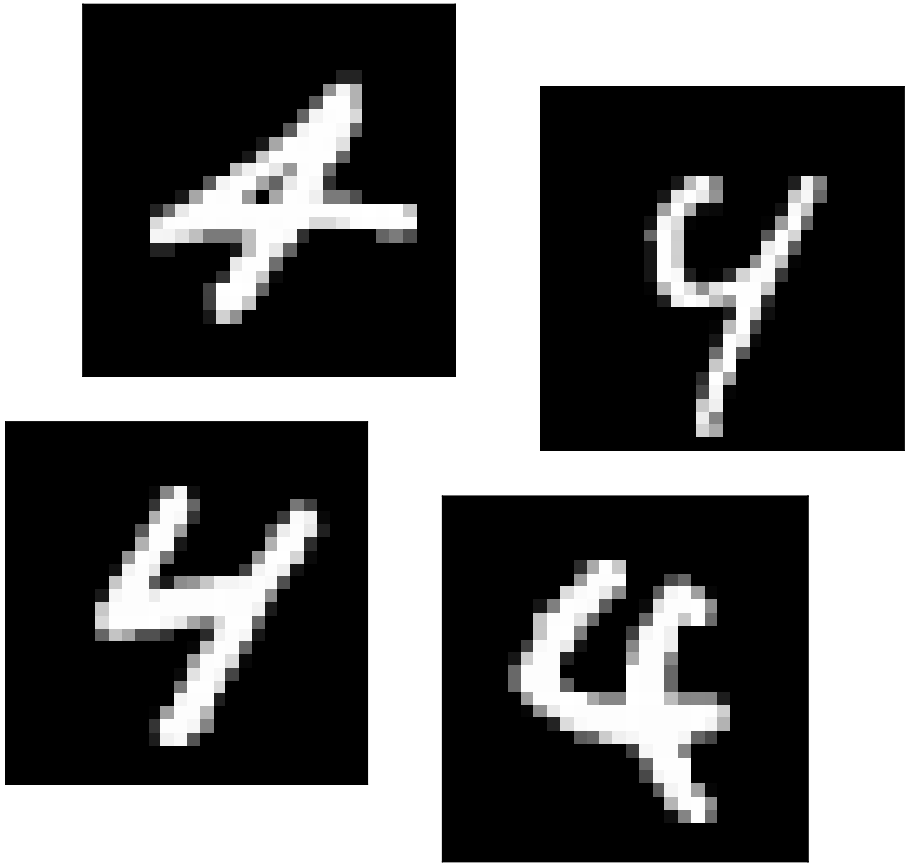


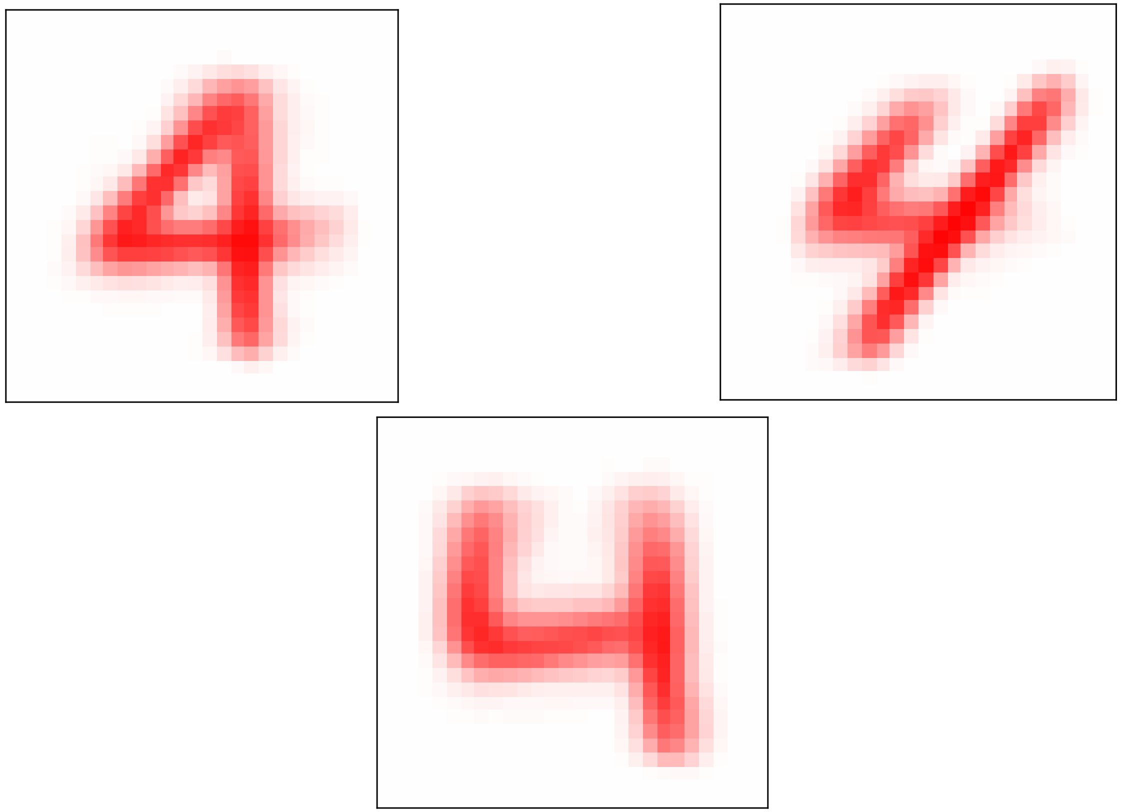
p detects images that look like this



The parameters $\boldsymbol{b}$ after reshaping

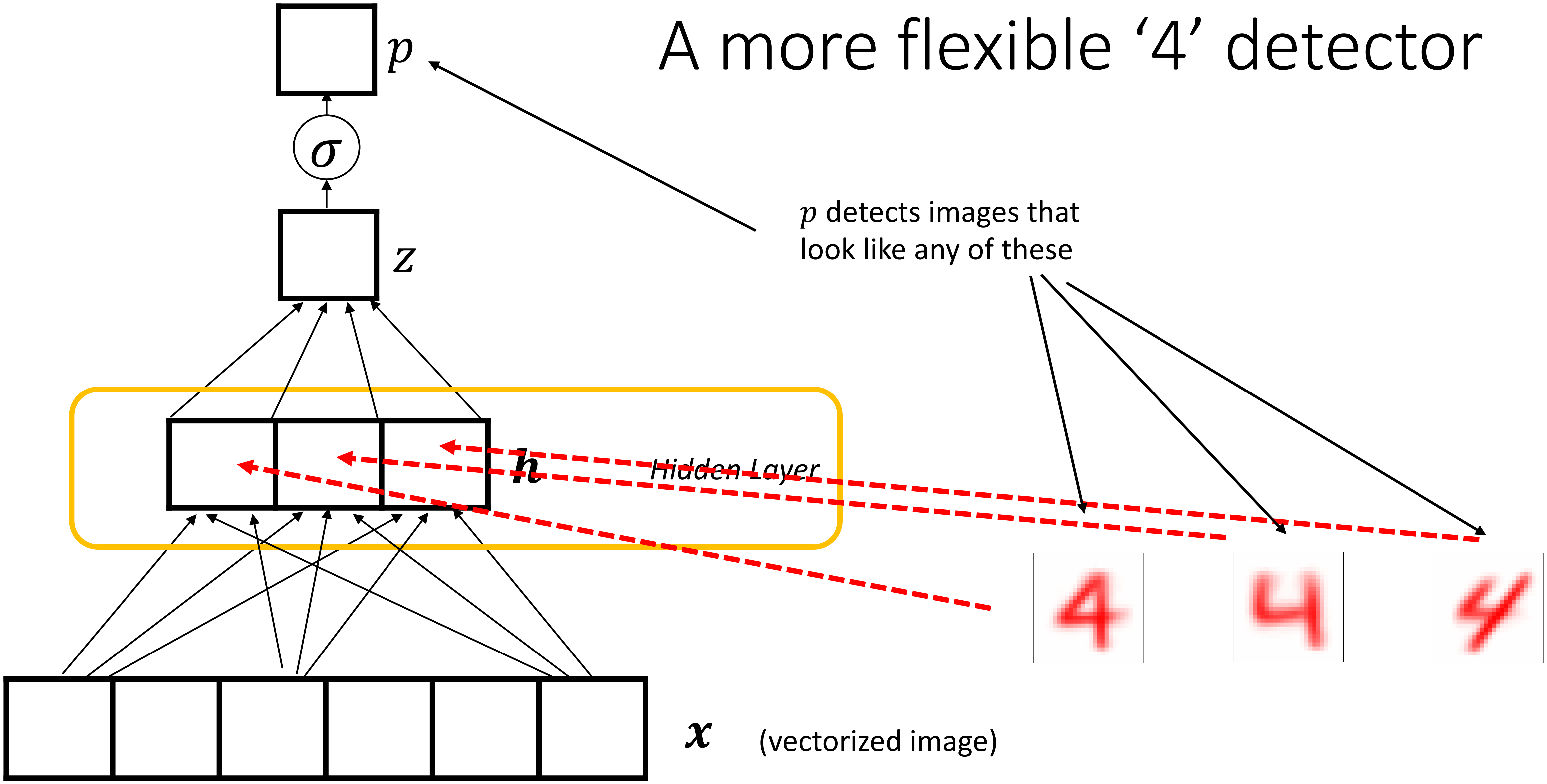
Return to MNIST:
Many ways of writing “4”





Multiple filters can look for *subtypes* indicative of different ways of writing “4”

A more flexible ‘4’ detector



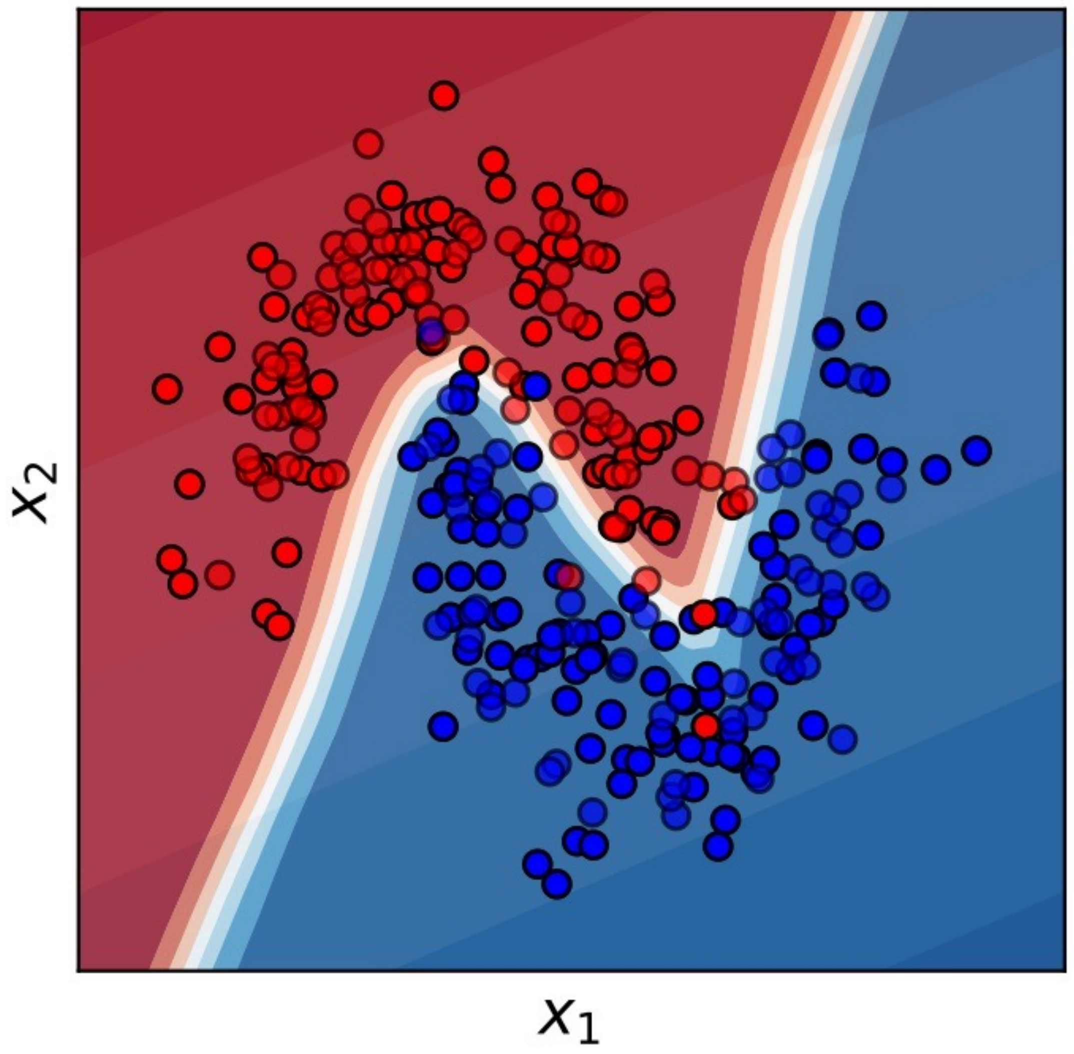
To increase flexibility/complexity, we can:

- Increase the width (i.e. number of hidden units) in one or more hidden layers (WIDE)

- Increase the number of hidden layers (DEEP)

- *Deep learning* refers to the latter; we are building a deep hierarchy of features

Learn Highly Non-Linear Decision Surfaces



Does this work with MNIST?

Logistic regression:
~91% Accurate

MLP with 1 hidden layer:
~96% Accurate

...maybe we can do even better…

Summary

- There are some binary classification tasks – in fact many binary classification tasks – that logistic regression just can't solve.
- However, by stacking many logistic regressions together, we are able to learn intermediate, *latent* features, thereby breaking up one complex problem into many simple ones.
- This is called a multilayer perceptron (MLP), or artificial neural network (ANN), or just neural network (NN).
- In NNs with multiple hidden layers, features detected by the hidden units become increasingly complex with each successive layer. We therefore say that the MLP learns a *hierarchy* of features.
- However, the MLP also has disadvantages. One disadvantage is that it typically requires more training data than logistic regression. We will discuss these more in later lectures.