

Machine Learning Summer School: Day 1

David Carlson

High-Level Introduction to Deep Learning

Part I

What is happening today?

- This class is focused on “Deep Learning:”
 - What it is
 - Some of its many applications
 - How to set up code to *learn your own* deep networks
- These algorithms *learn* from data to automate tasks and make predictions
- Today, you will learn about what we mean by “deep learning” and can *learn* your first deep network

Deep Learning is state-of-the-art for *many* applications

- Deep Learning is **not new**—many of the techniques go back decades
- Recent resurgence due to *amazing* performance on benchmark tasks
- One key task was the ImageNet Challenge
 - Want to recognize what is in an image (1 of 1000 categories)
 - Have ~1 million example images
 - Very relevant for things such as image search
- Example images are shown on the right, with predicted categories beneath each image

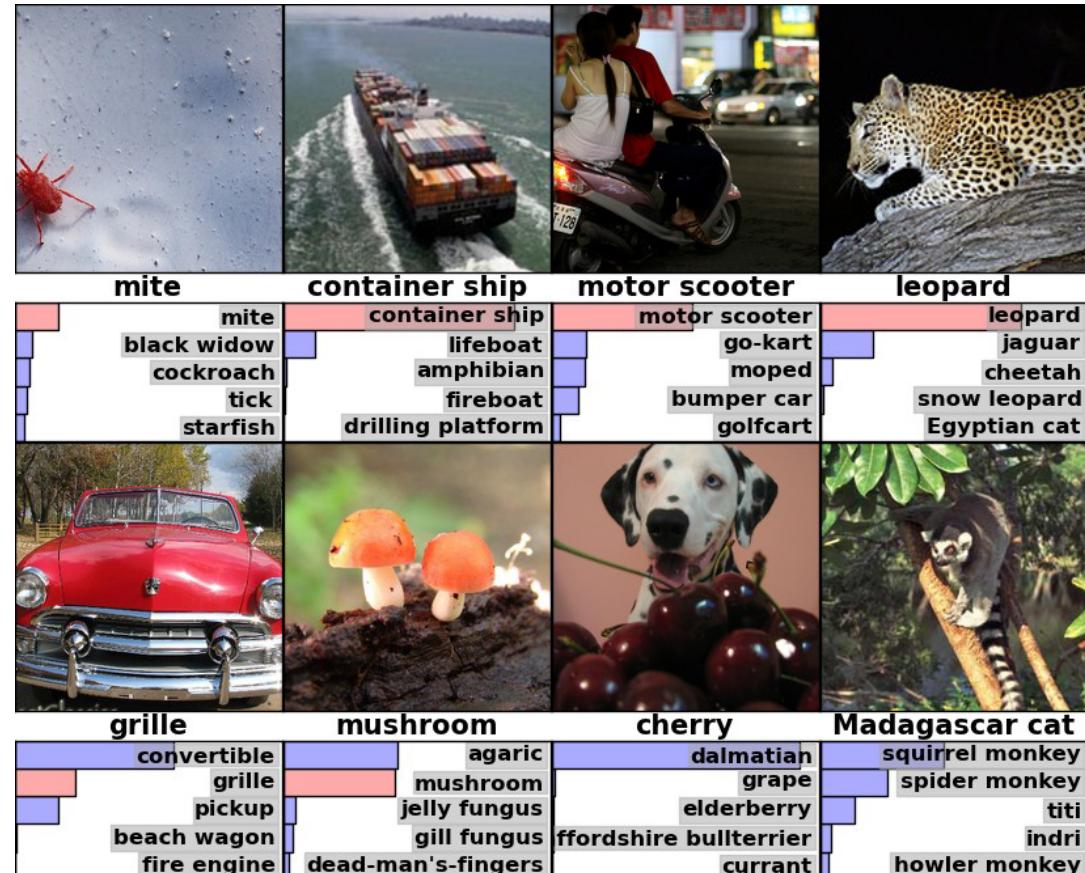
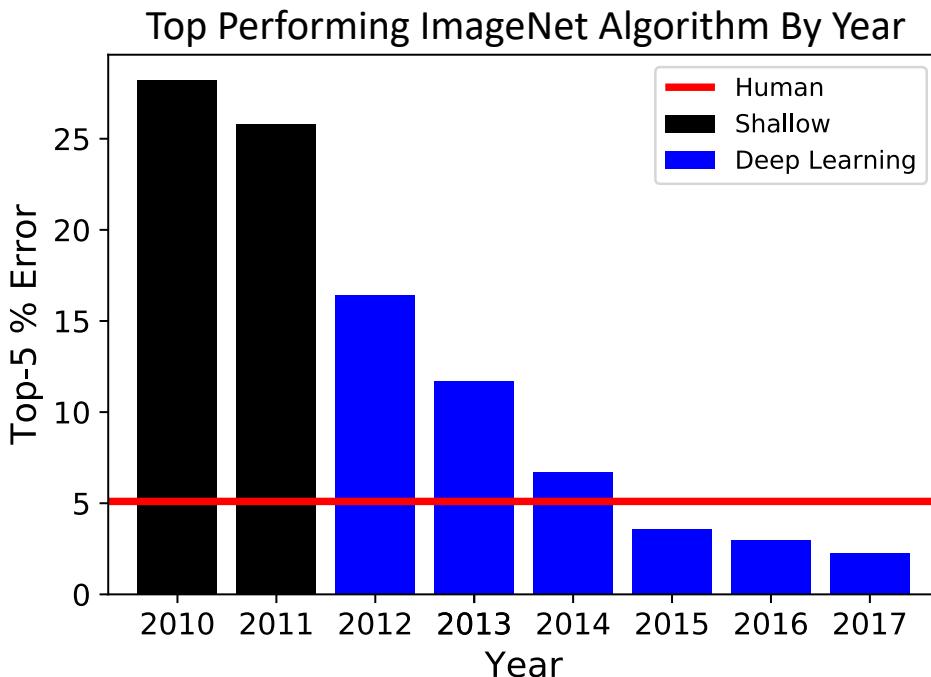


Figure from Krizhevsky et al 2012.

Deep Learning can surpass human performance

- For ImageNet:
 - Deep Learning was a *huge* jump forward
 - State-of-the-art systems **significantly outperform humans** on the same task
- These use “Convolutional Neural Networks,” which you will learn about tomorrow



Deep Learning beats human performance in many tasks

- Famously, Google DeepMind trained “AlphaGo” to beat the world champion Go player (a complex game)
- AlphaGo’s largely works by Deep Learning techniques (learned by repeatedly playing the game)
- Many other examples:
 - Voice Recognition
 - Object Detection
 - Text Translation
 - Etc
- So what is Deep Learning?



We need to understand “shallow” learning before “deep” learning

A “SHALLOW” NETWORK

Learning a Predictive Model Based on Labeled Data



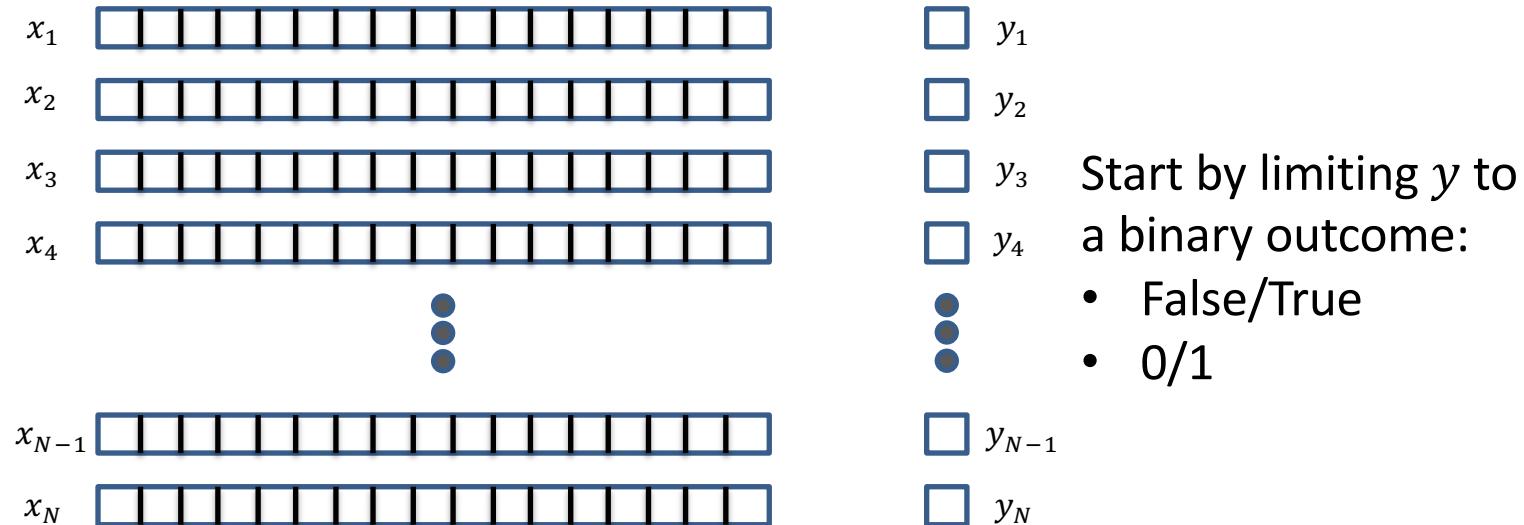
x , data/features for
a subject



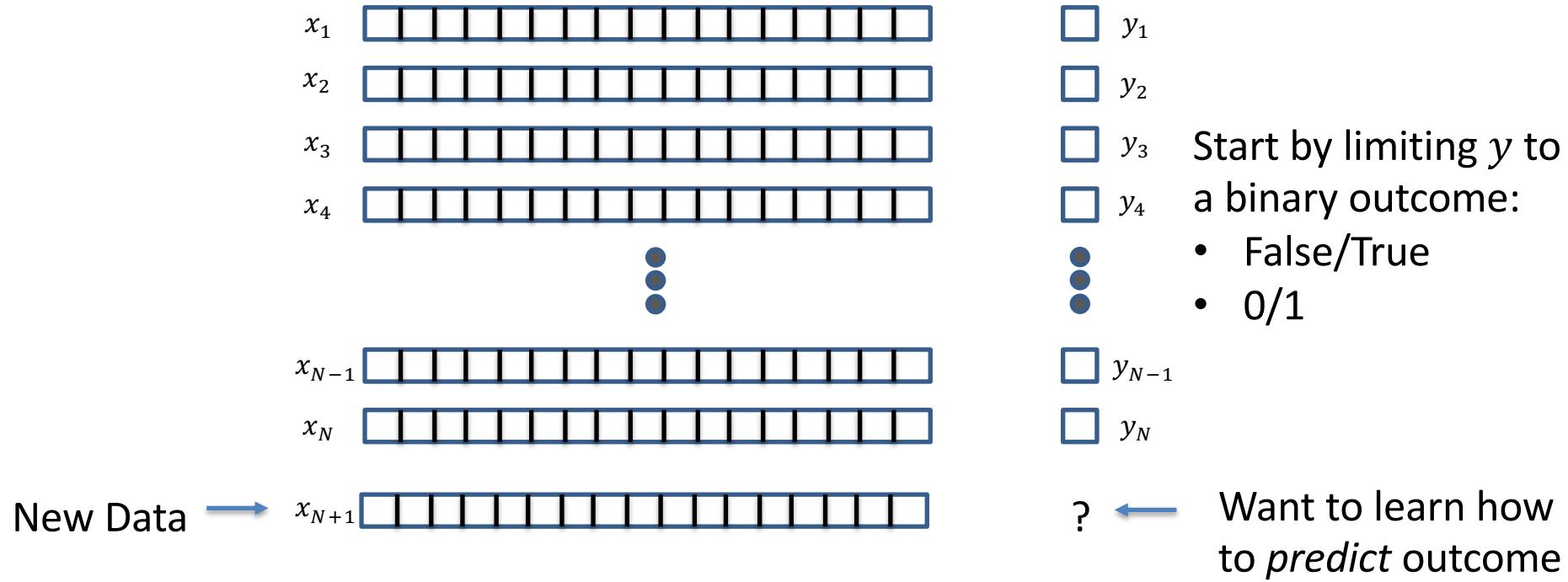
y , associated label 0/1

End goal: *predict* y from x

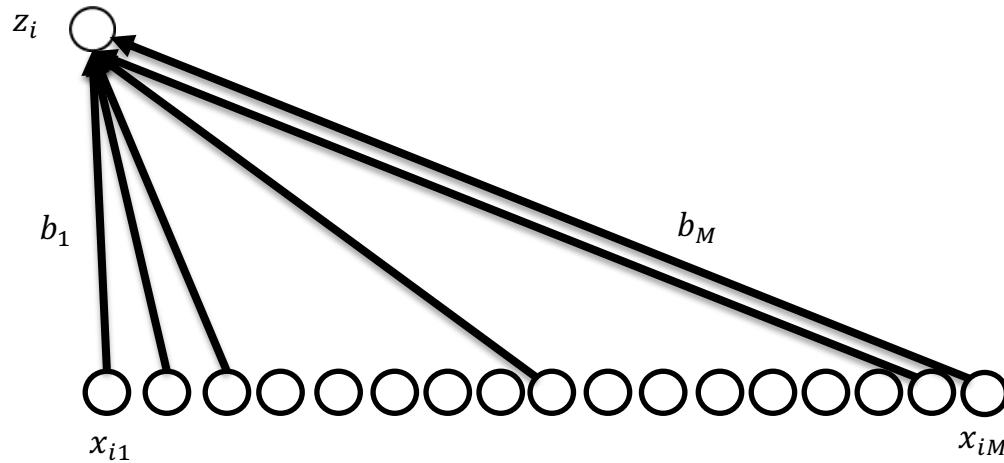
Training Set (Historical Data)



Making Predictions



Linear Predictive Model



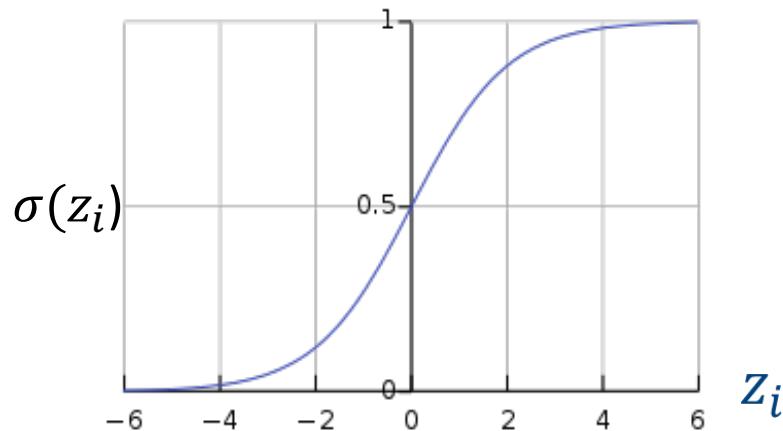
$$z_i = (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + (b_M \times x_{iM})$$

Convert to a Probability

$$z_i = (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + (b_M \times x_{iM}) + b_0$$

$$p(y_i = 1|x_i) = \sigma(z_i)$$

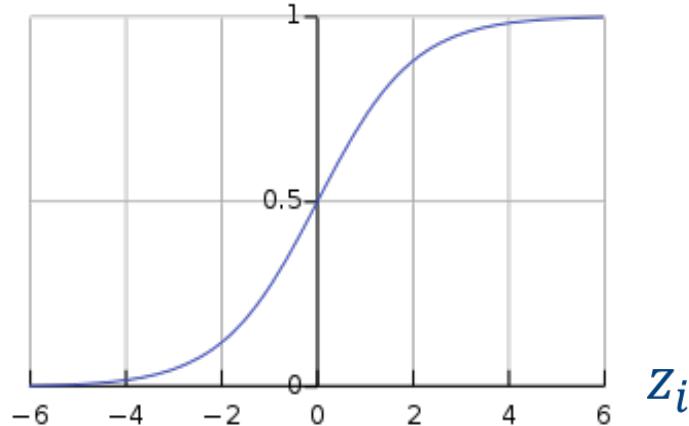
Extra Constant



Convert to a Probability

$$z_i = (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + (b_M \times x_{iM}) + b_0$$

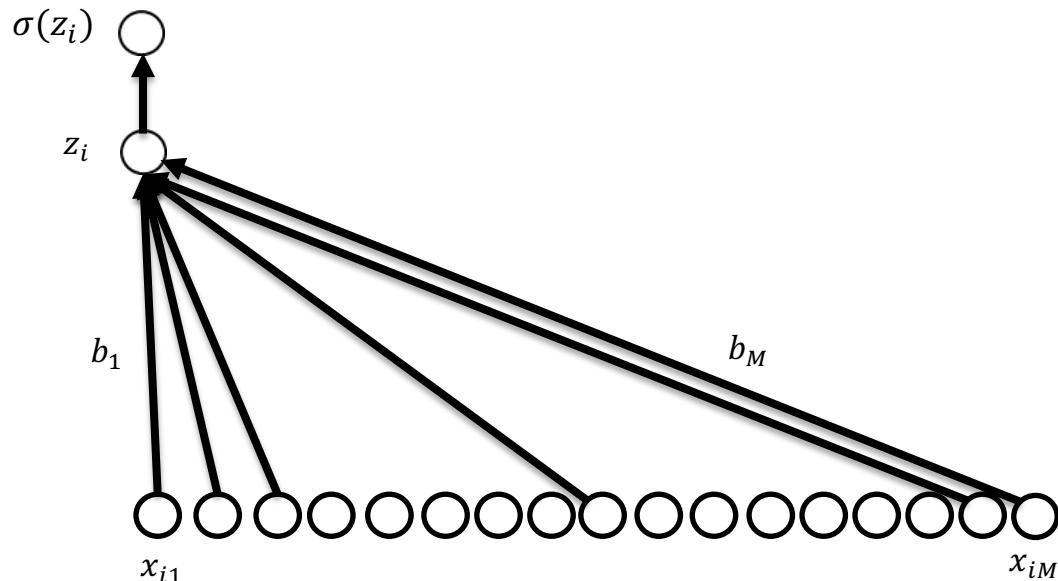
$$p(y_i = 1|x_i) = \sigma(z_i) = \frac{\exp(z_i)}{1+\exp(z_i)}$$



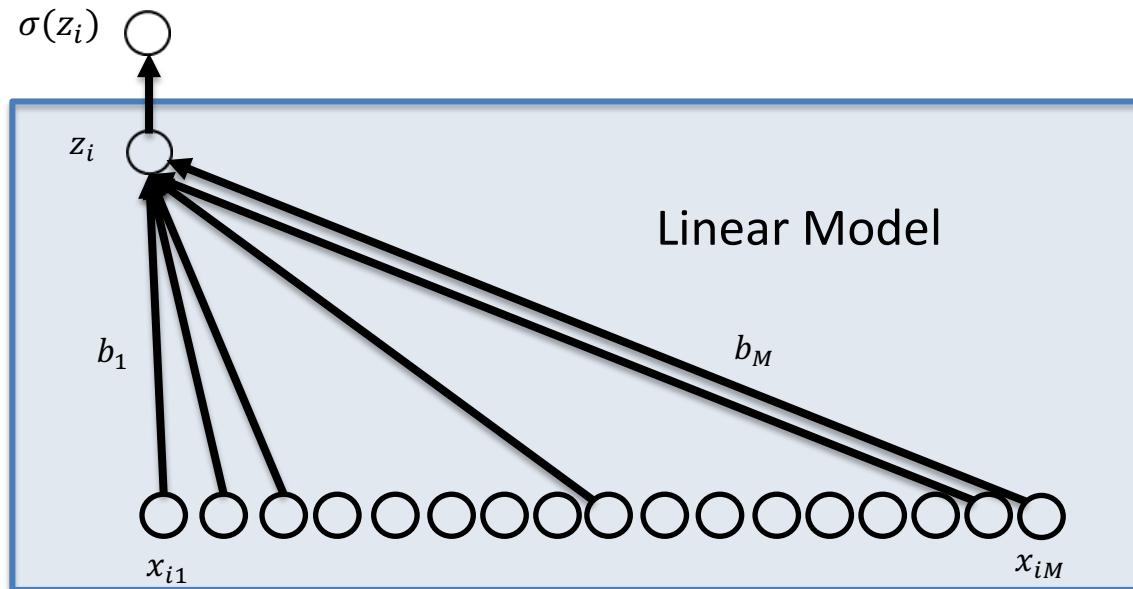
- Large and positive z_i indicates that event $y_i = 1$ is likely

- Large and negative z_i indicates that event $y_i = 0$ is likely

Logistic Regression

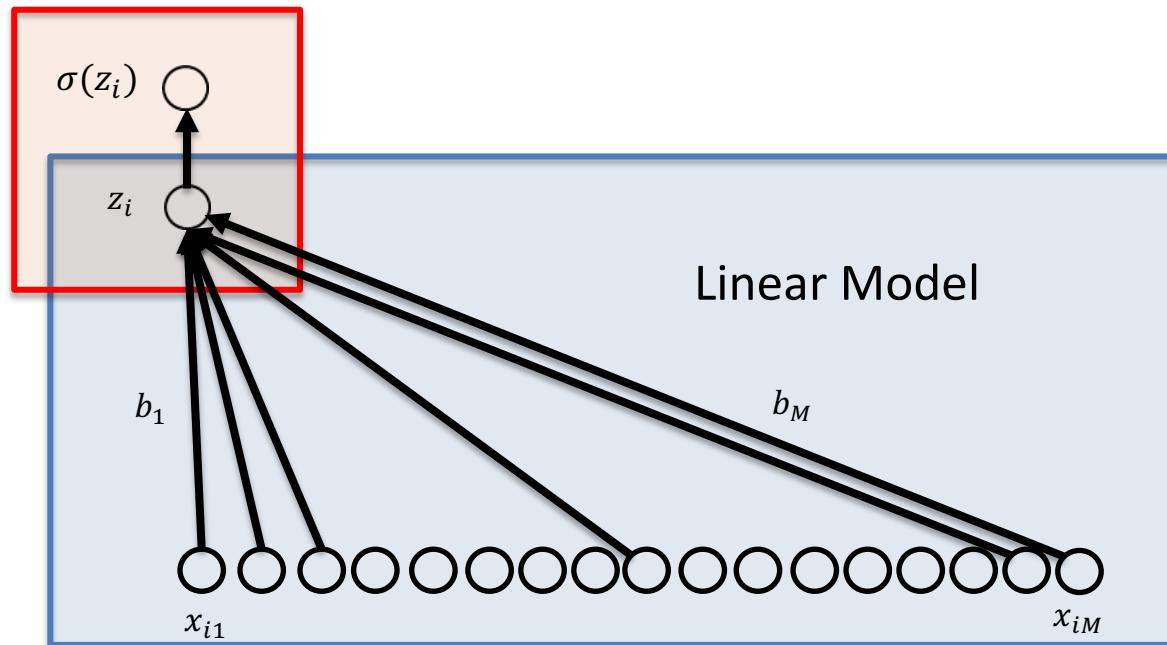


Logistic Regression



Logistic Regression

Convert to
Probability



What do the parameters and model mean?

AN EXAMPLE

Example

- Outcome:
 - $y_i = 1$, it rains on day i ;
 - $y_i = 0$, it does not rain on day i
- Features:
 - On day i what is the *{cloud cover, humidity, temperature, air pressure, ...}*



x_i , features for day i

y_i , did it rain on day i

Example

- **Outcome:** $y_i = 1$, it rains on day i ; $y_i = 0$, it does not rain on day i
- **Features:** On day i what is the {1: *cloud cover*, 2: *humidity*, 3: *temperature*, ... }

$$z_i = (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + (b_M \times x_{iM}) + b_0$$

Cloud Cover Humidity

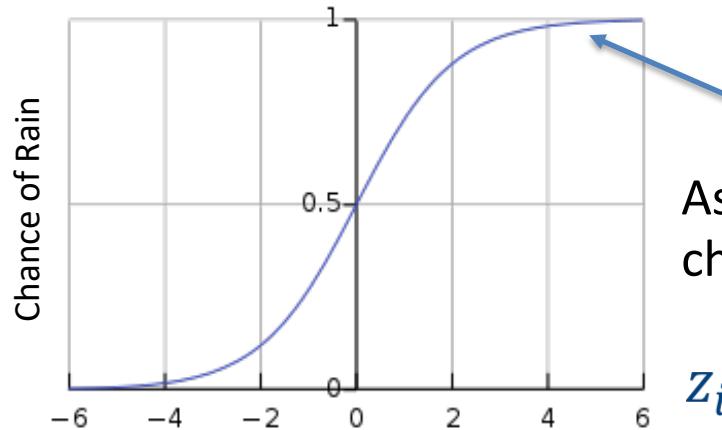
- If cloud cover is positively related to rainfall, b_1 should be positive

Impact on the Sigmoid Function

$$z_i = (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \dots + (b_M \times x_{iM}) + b_0$$

Cloud Cover *Humidity*

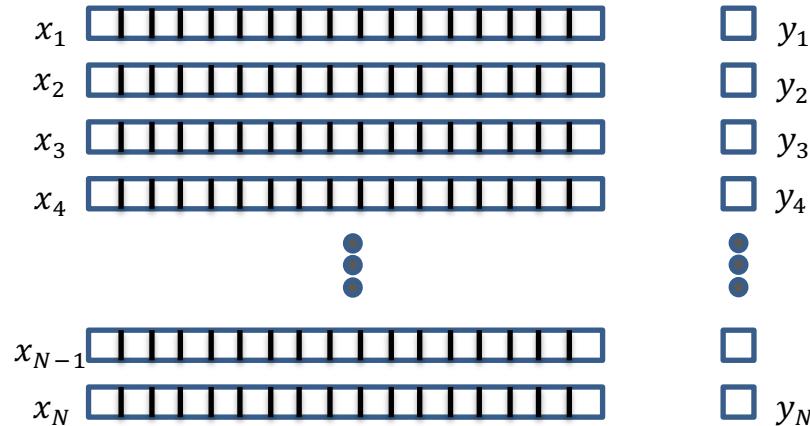
$$p(y_i = 1|x_i) = \sigma(z_i)$$



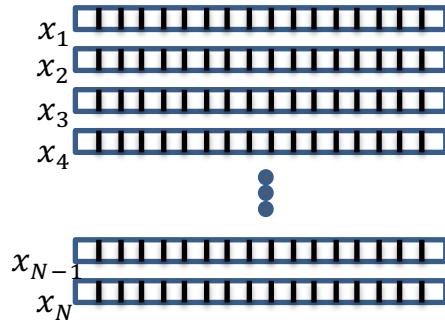
As the value z_i increases, the chance of rain increases

Building the Training Set

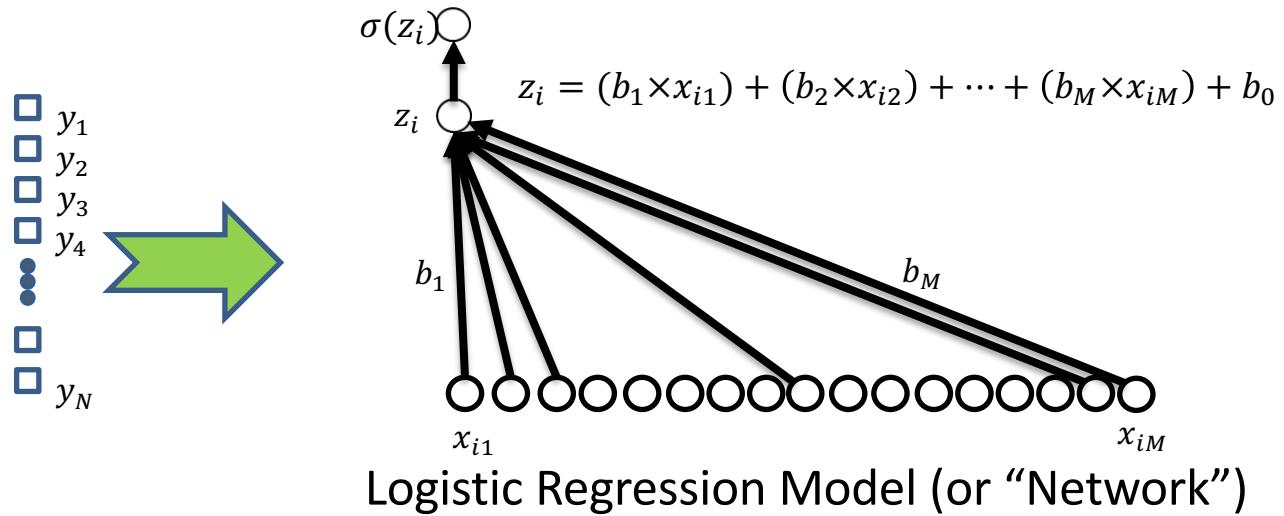
- Need to learn the parameters
- Requires *training data*
- Record data from N days
 - Capture features: {*cloud cover, humidity, temperature, ...*}
 - Did it rain?



Learning Model Parameters



Training Set



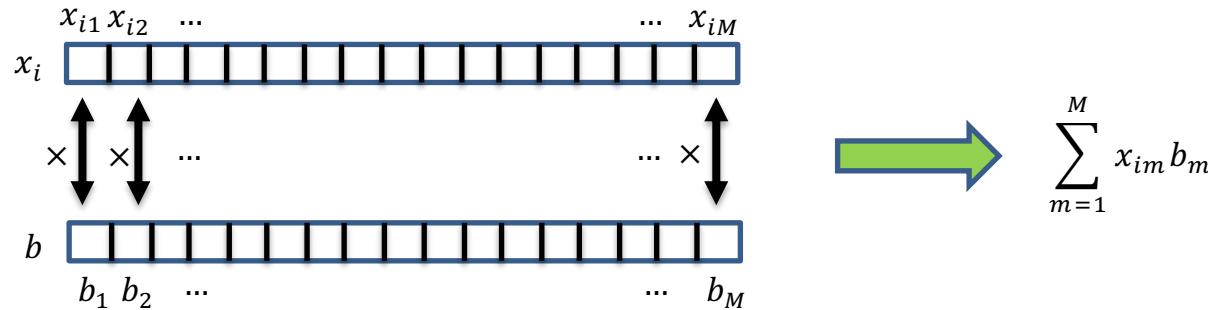
Logistic Regression Model (or “Network”)

Learned
Parameters (b_0, b_1, \dots, b_N)



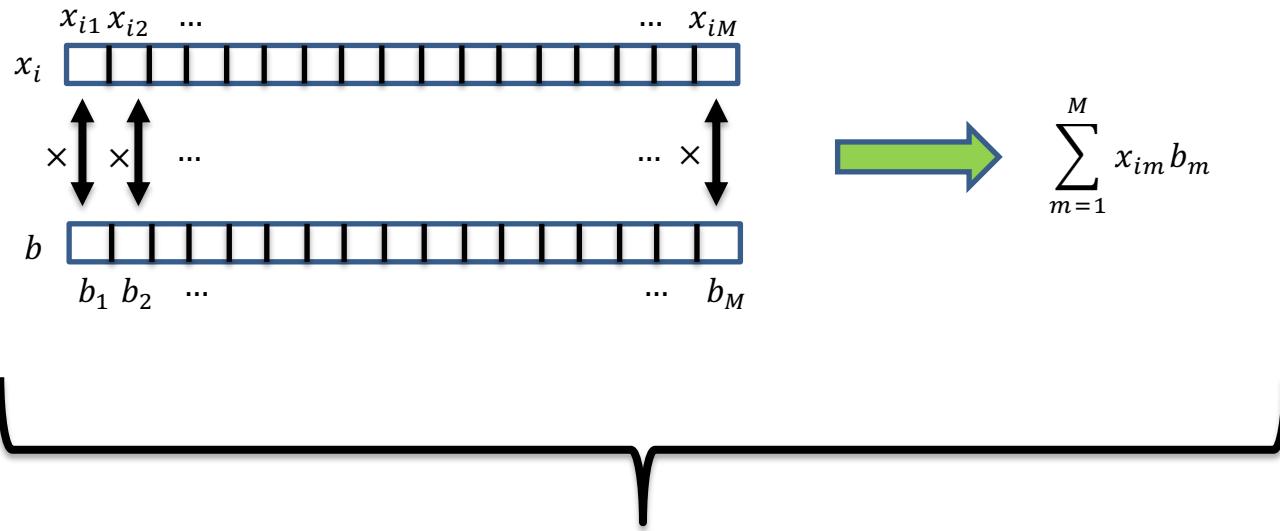
Interpretation of Logistic Regression

$$z_i = (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + (b_M \times x_{iM})$$



Interpretation of Logistic Regression

$$z_i = (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + (b_M \times x_{iM})$$



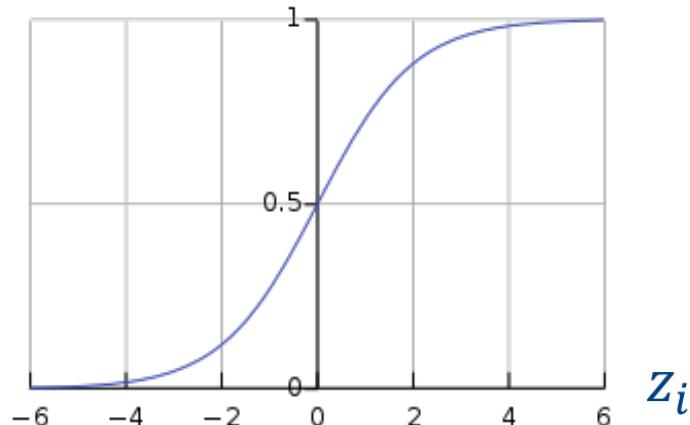
Compact Notation: $x_i \odot b$ (or “inner product”)

Interpretation of Logistic Regression

$$z_i = b_0 + (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + (b_M \times x_{iM})$$

$$= b_0 + x_i \odot b$$

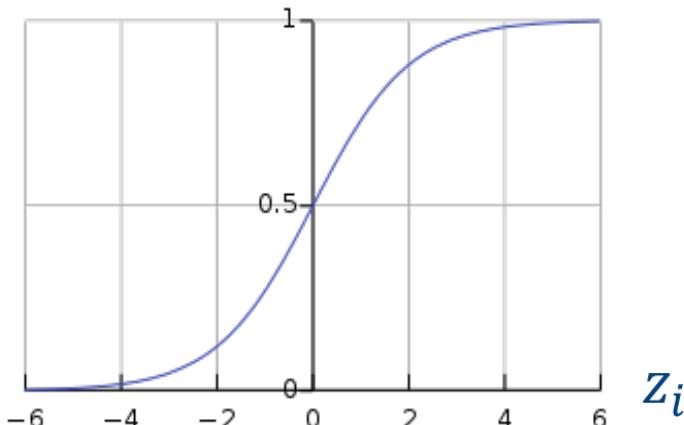
$$p(y_i = 1|x_i) = \sigma(z_i)$$



Interpretation of Logistic Regression

$$\begin{aligned}z_i &= b_0 + (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + (b_M \times x_{iM}) \\&= b_0 + x_i \odot b\end{aligned}$$

$$p(y_i = 1|x_i) = \sigma(z_i)$$



- May think of vector b as a template or filter (will visualize to make clear)
- If x_i is aligned/matched with b , then $x_i \odot b$ will be large
- The parameter b_0 is a bias to correct for class prevalence

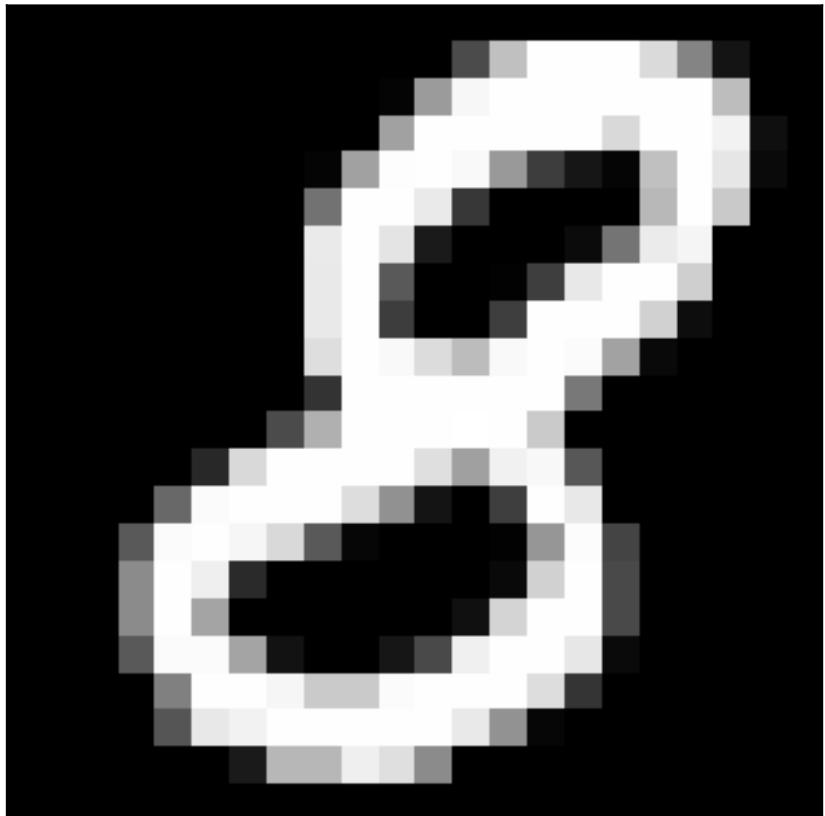
RECOGNIZING NUMERICAL DIGITS

The MNIST Dataset

- The Modified National Institute of Standards and Technology (MNIST) contains pictures of handwritten digits (0,1,2,...)
- Want to be able to tell what digit each image is (e.g. optical character recognition)

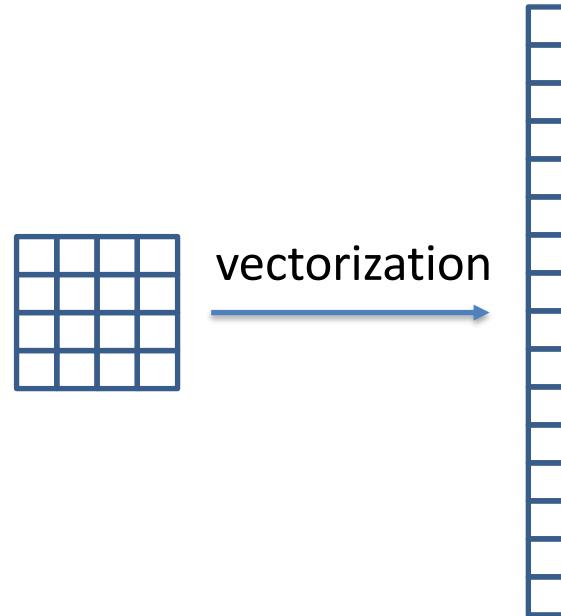


Images are Encoded as Numbers



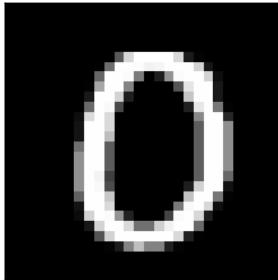
Vectorization

- We will start talking about deep learning *without* using the structure of the image
- We will return to this in a future lecture
- To convert an image into an unstructured set of numbers, we *vectorize it*

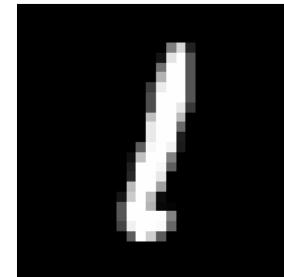
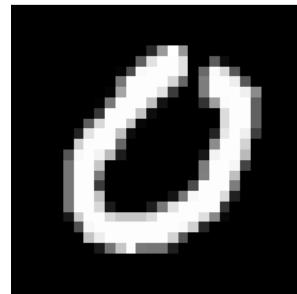
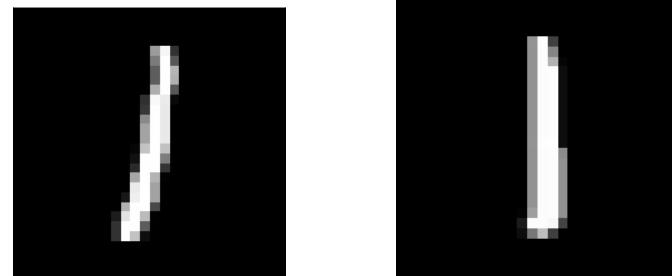


Start With The Binary Case

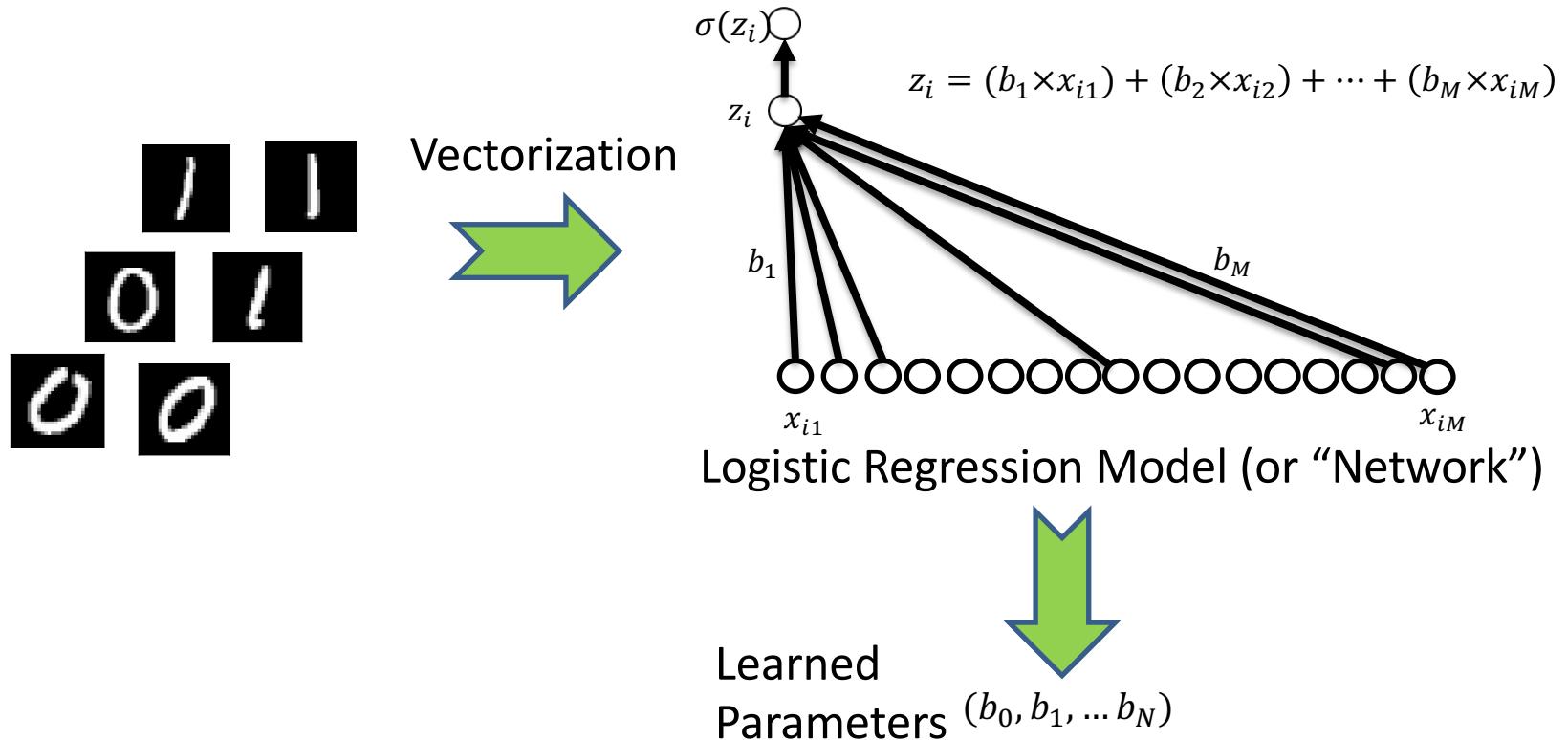
Zeros



Ones

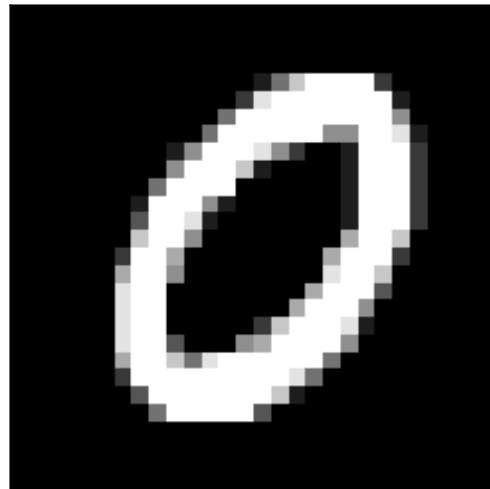


Learning on MNIST

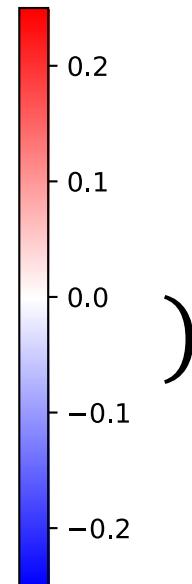
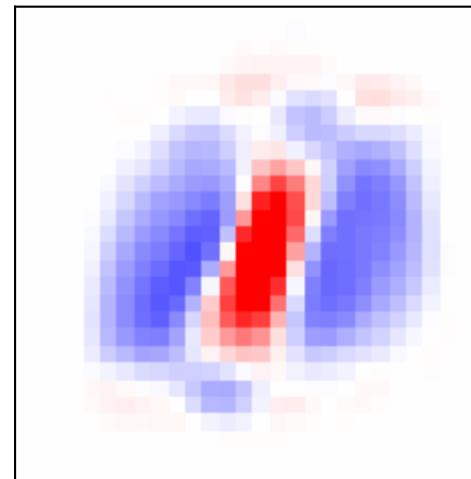


Zooming in on 0/1

$\sigma($



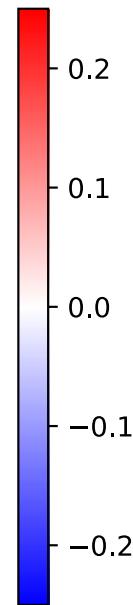
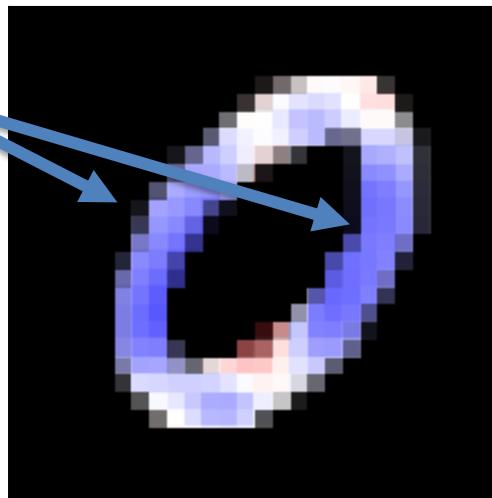
\times



Zooming in on 0/1

Negative Sections

$\sigma($

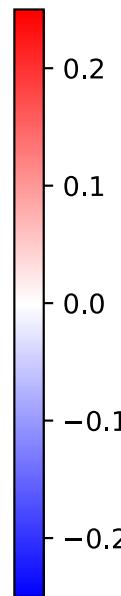
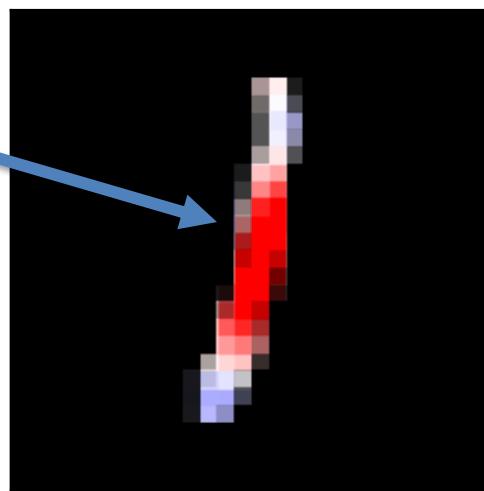


) = 0.006

Zooming in on 0/1

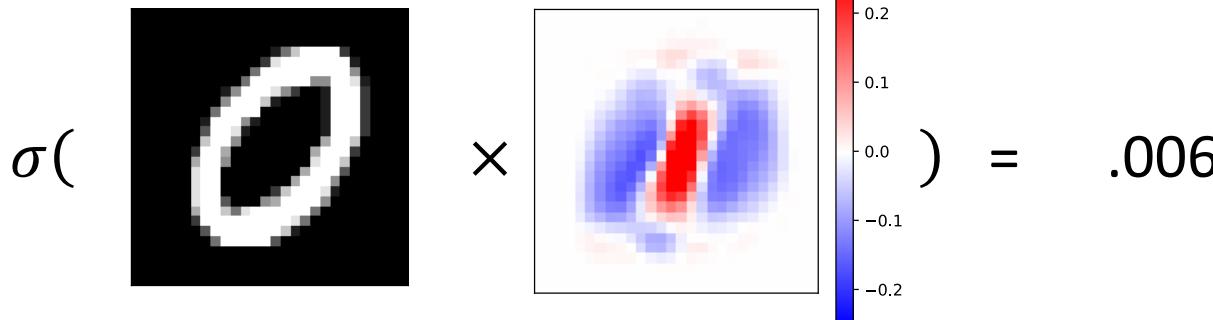
Positive Section

$\sigma($

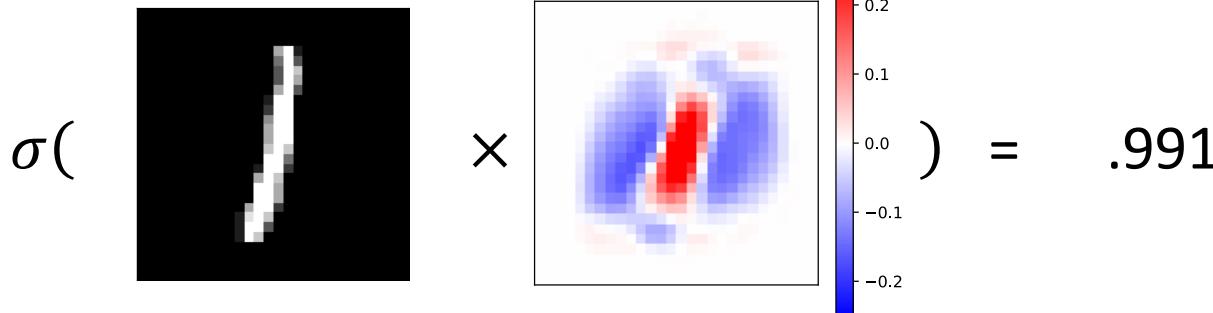


) = .991

Learned Weights for 0/1



We think that
this is a “zero”
(.6% chance it
is a “one”)



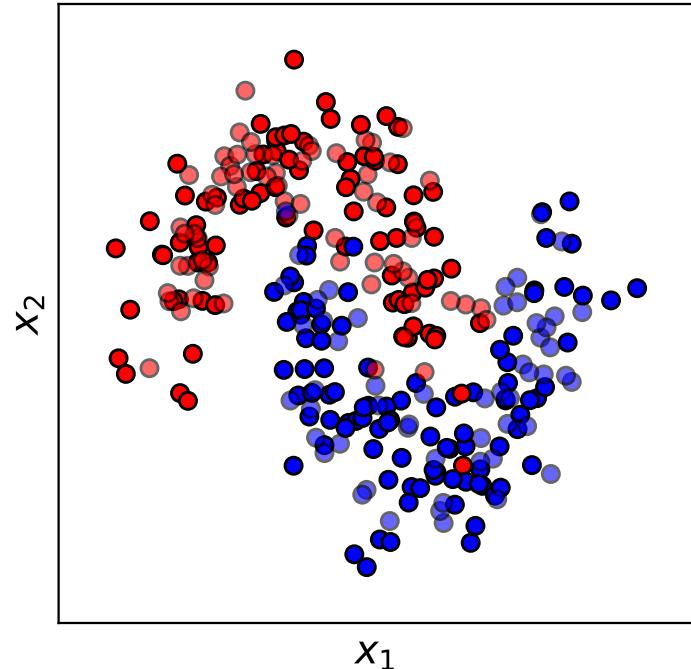
We think that
this is a “one”
(99.1% chance
it is a “one”)

From Shallow to Deep Learning

GENERALIZING LOGISTIC REGRESSION

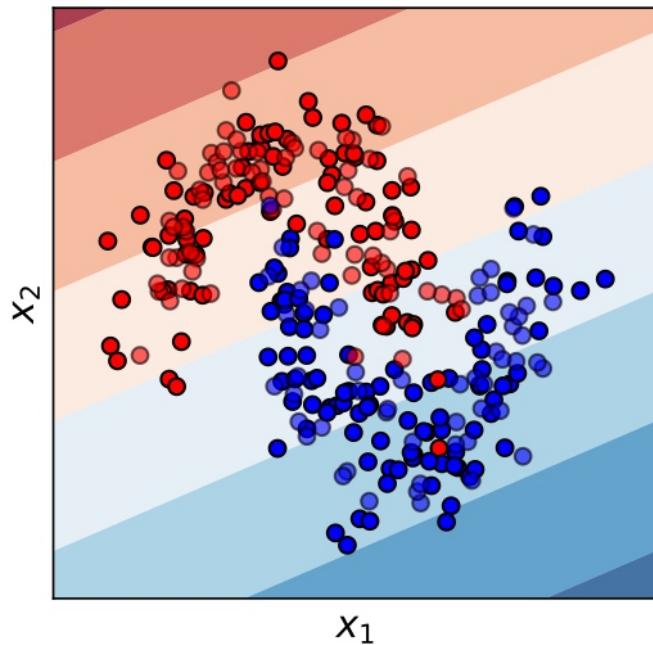
Logistic Regression is a “Linear” Classifier

- Also referred to as a generalized linear model
- Can only split data by linear trends



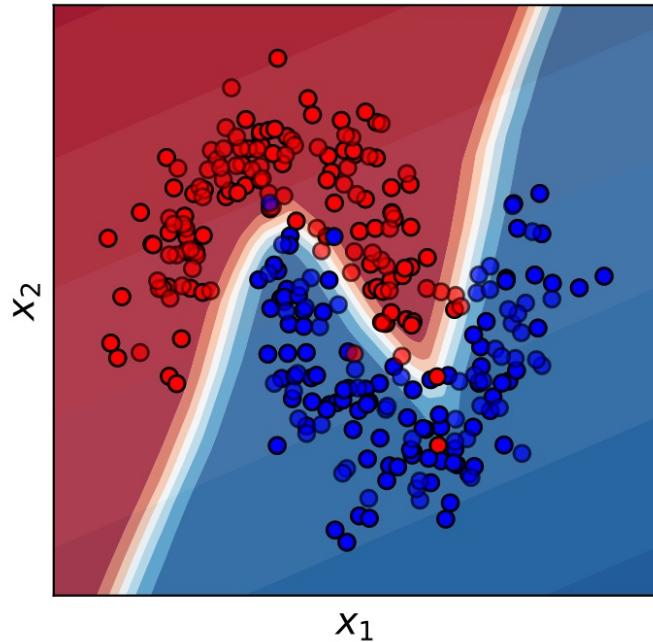
Limitations of Linear Classifiers

- Linear classifiers can only represent limited relationships
- Often want to use a classifier that can handle non-linearities.
- Many different ways of creating non-linear classifiers

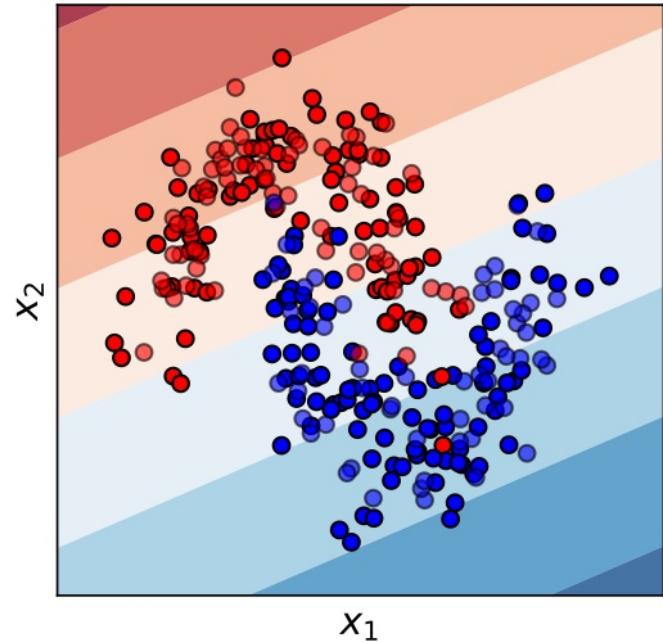


This is what we actually want!

Non-Linear Classifier

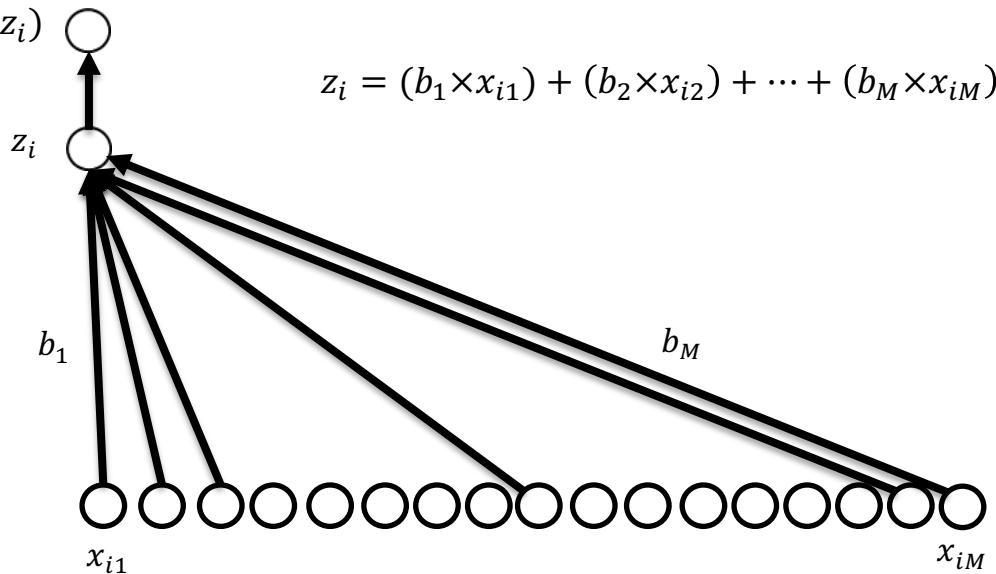


Linear Classifier

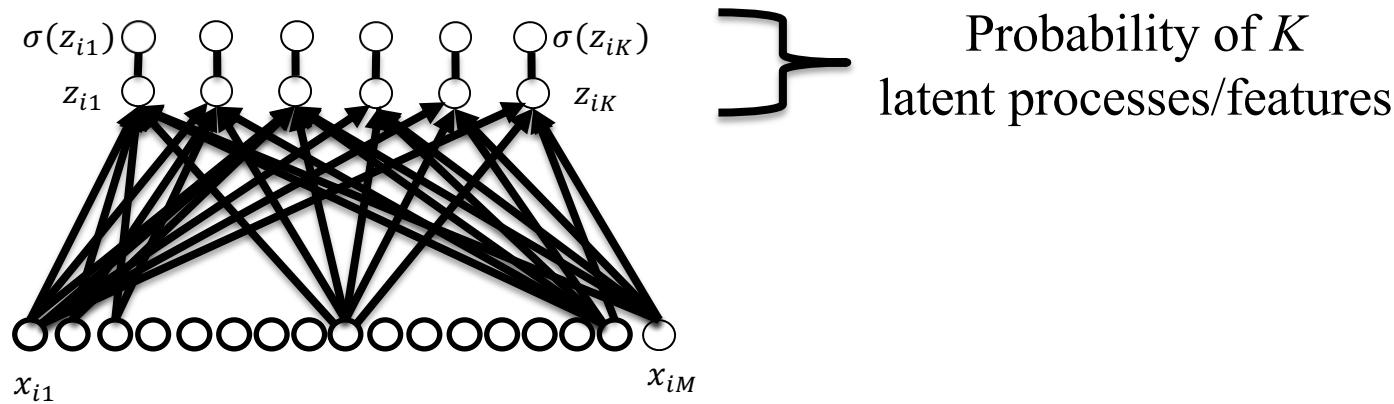


Can we modify Logistic Regression to learn complex structures?

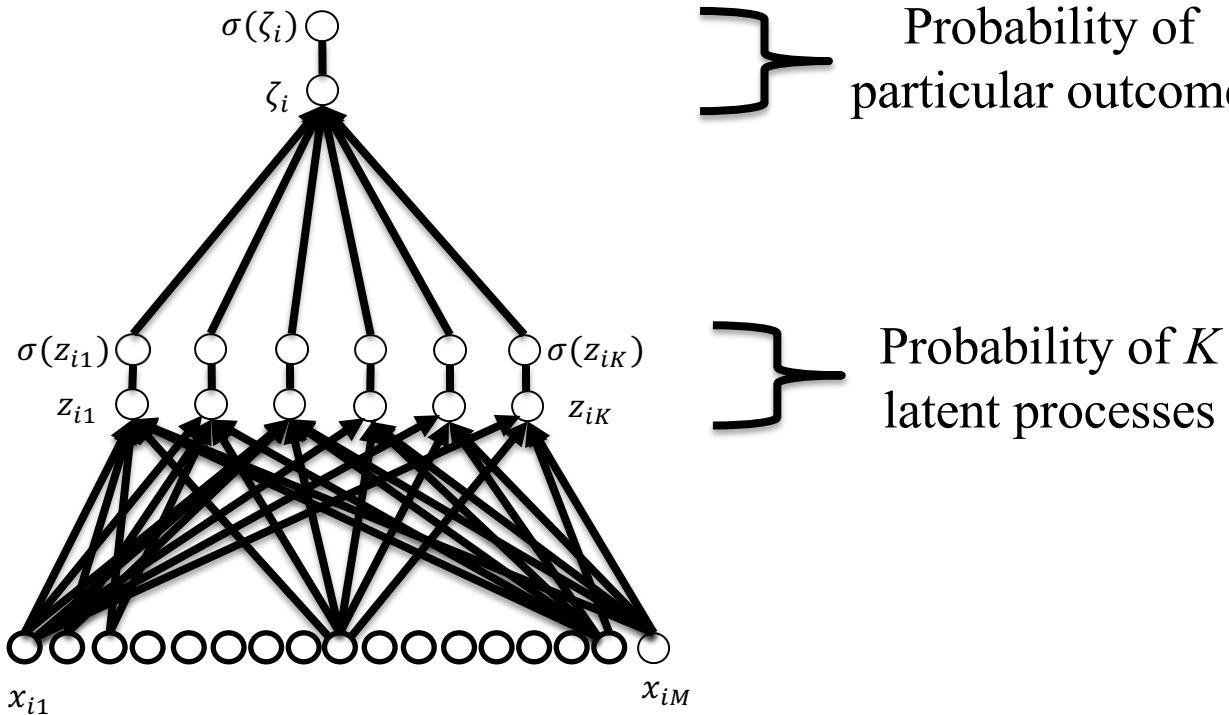
Probability of
a single
outcome



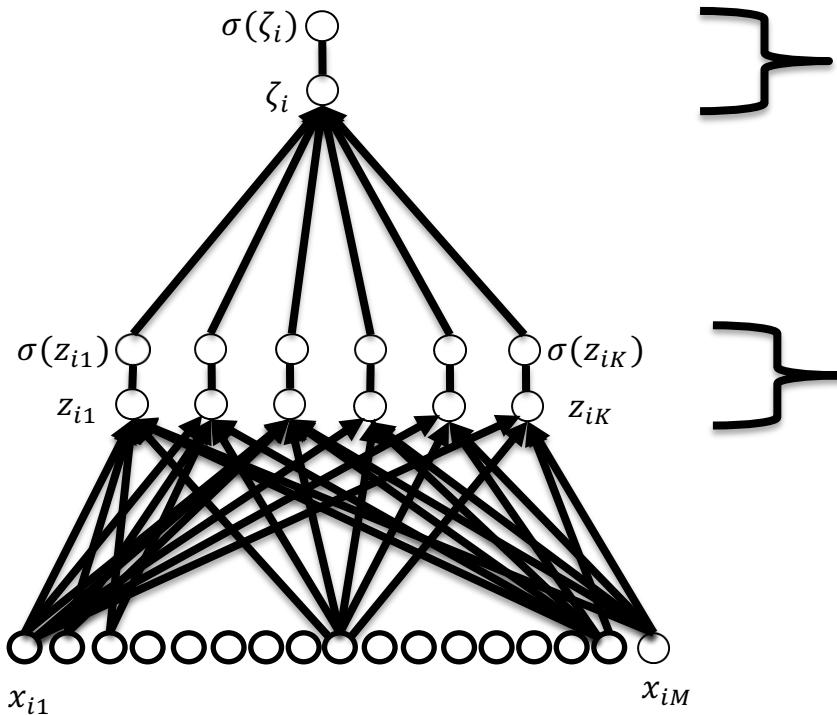
Generalization of Logistic Regression: Learned Features



Extended Logistic Regression



Extended Logistic Regression



Probability of
particular outcome

Probability of K
latent processes

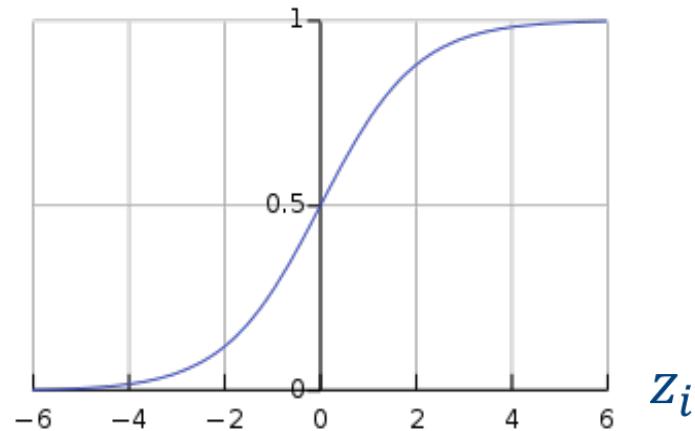
May be viewed as logistic
regression on K latent features,
rather than directly on the M
components of raw data

Recall Logistic Regression

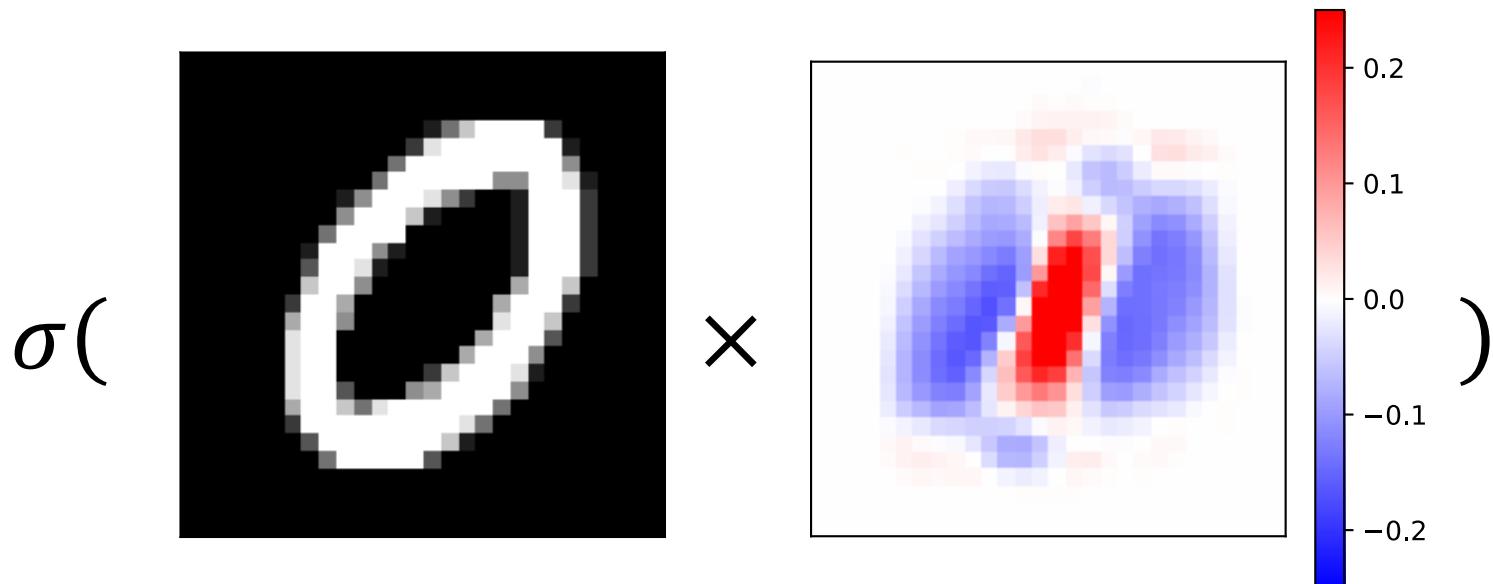
$$z_i = b_0 + (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + (b_M \times x_{iM})$$

$$= b_0 + x_i \odot b$$

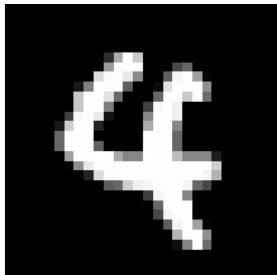
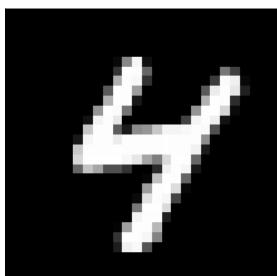
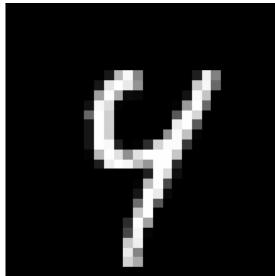
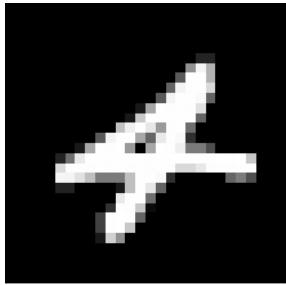
$$p(y_i = 1|x_i) = \sigma(z_i)$$



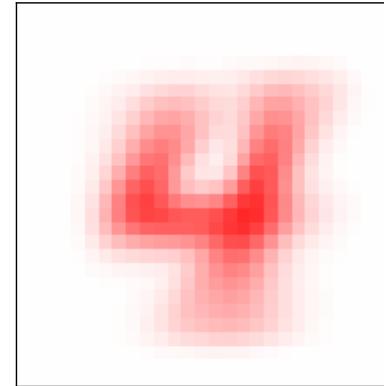
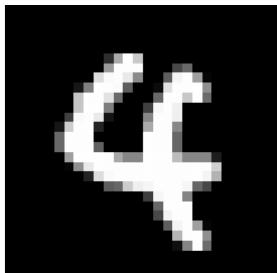
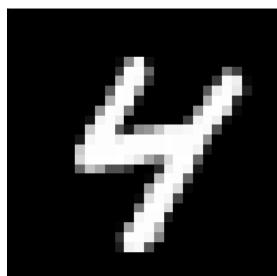
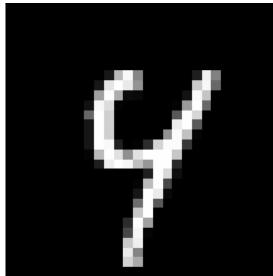
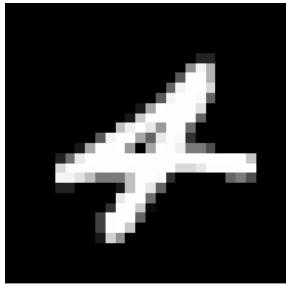
Why Limit Ourselves to Only One Filter?



Return to MNIST: Many ways of writing “4”

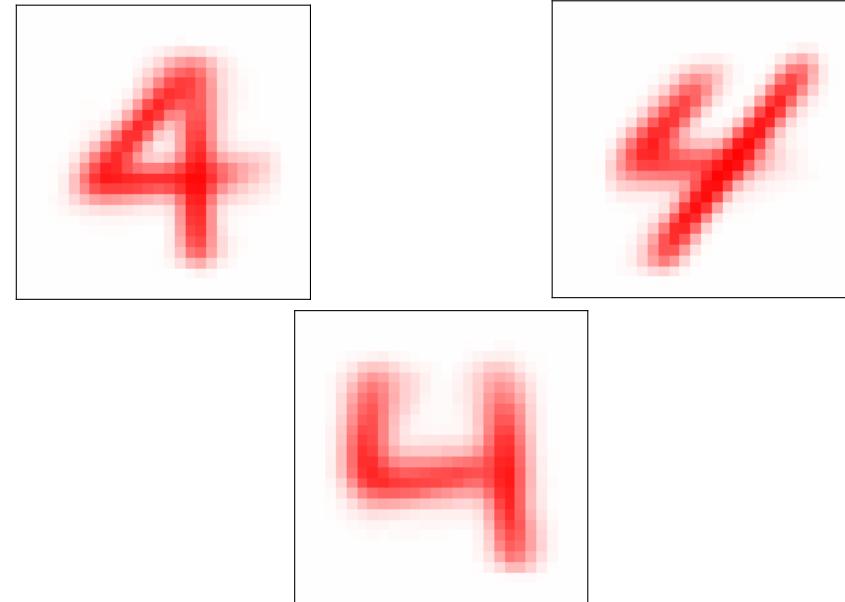


Return to MNIST: Many ways of writing “4”



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

Return to MNIST: Many ways of writing “4”



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

Why Limit Ourselves to Only One Filter?

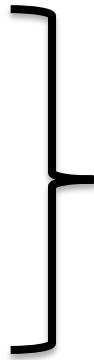
Introduce K Filters

$$z_{i1} = b_{01} + x_i \odot b_1$$

$$z_{i2} = b_{02} + x_i \odot b_2$$

⋮
⋮

$$z_{iK} = b_{0K} + x_i \odot b_K$$



Project data x_i onto K filters: b_1, \dots, b_K

Why Limit Ourselves to Only One Filter?

Introduce K Filters

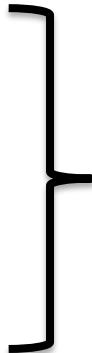
$$z_{i1} = b_{01} + x_i \odot b_1$$

$$z_{i2} = b_{02} + x_i \odot b_2$$

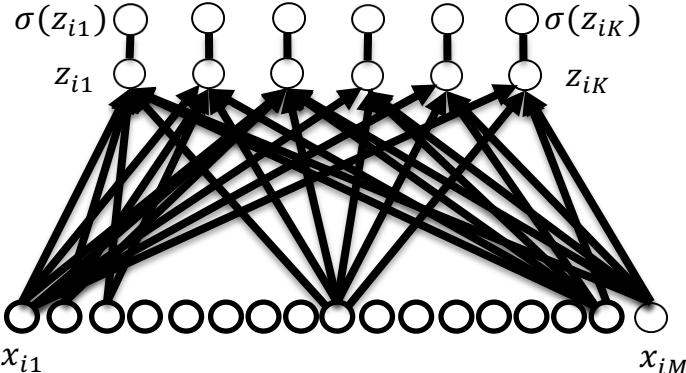
\vdots

\vdots

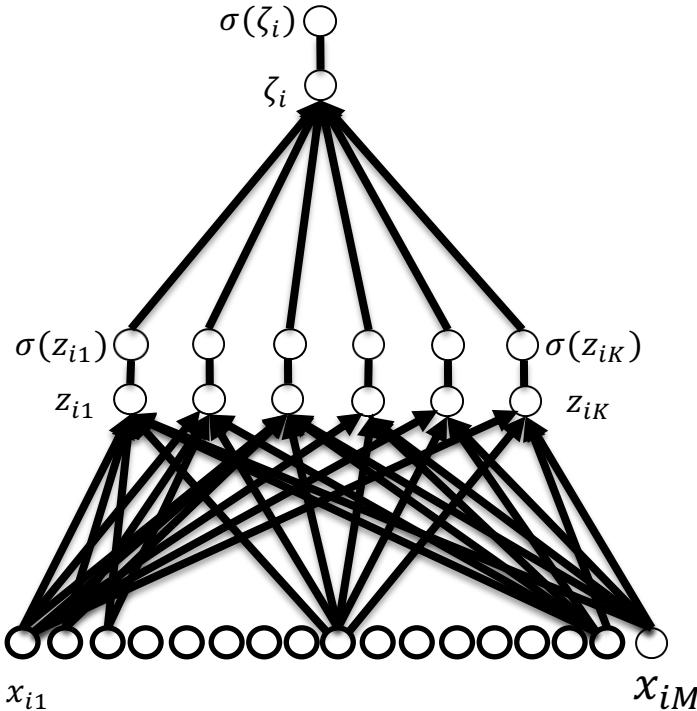
$$z_{iK} = b_{0K} + x_i \odot b_K$$



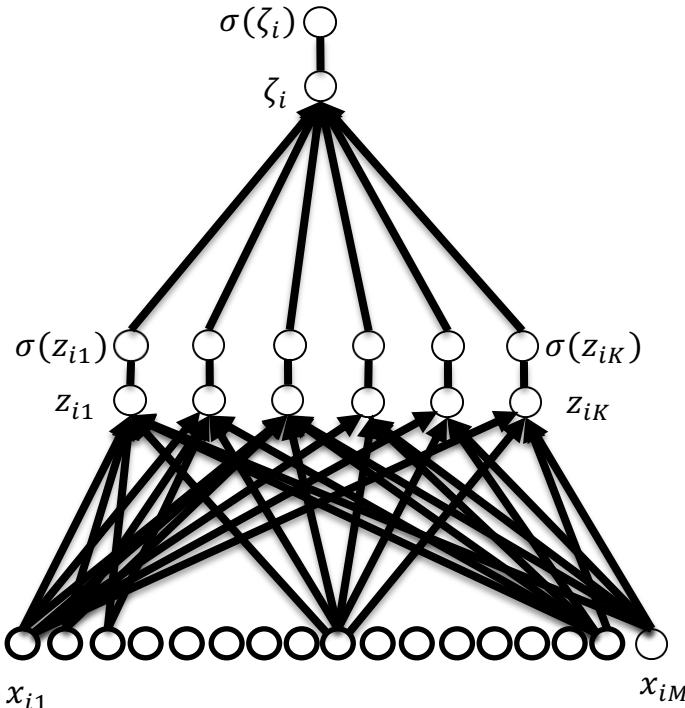
Project data x_i onto K filters: b_1, \dots, b_K



Extended Logistic Regression



Extended Logistic Regression



$$z_{i1} = b_{01} + x_i \odot b_1$$

$$z_{i2} = b_{02} + x_i \odot b_2$$

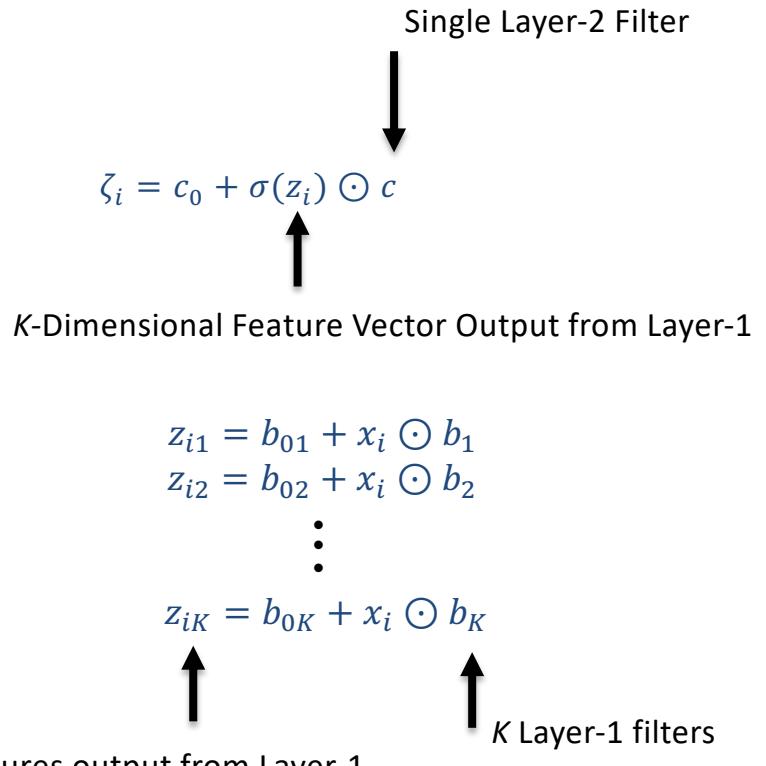
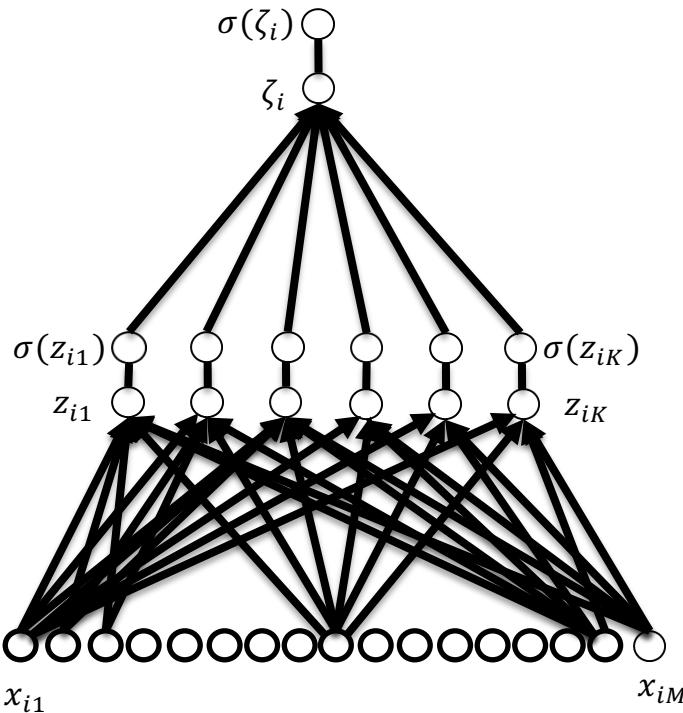
⋮

$$z_{iK} = b_{0K} + x_i \odot b_K$$

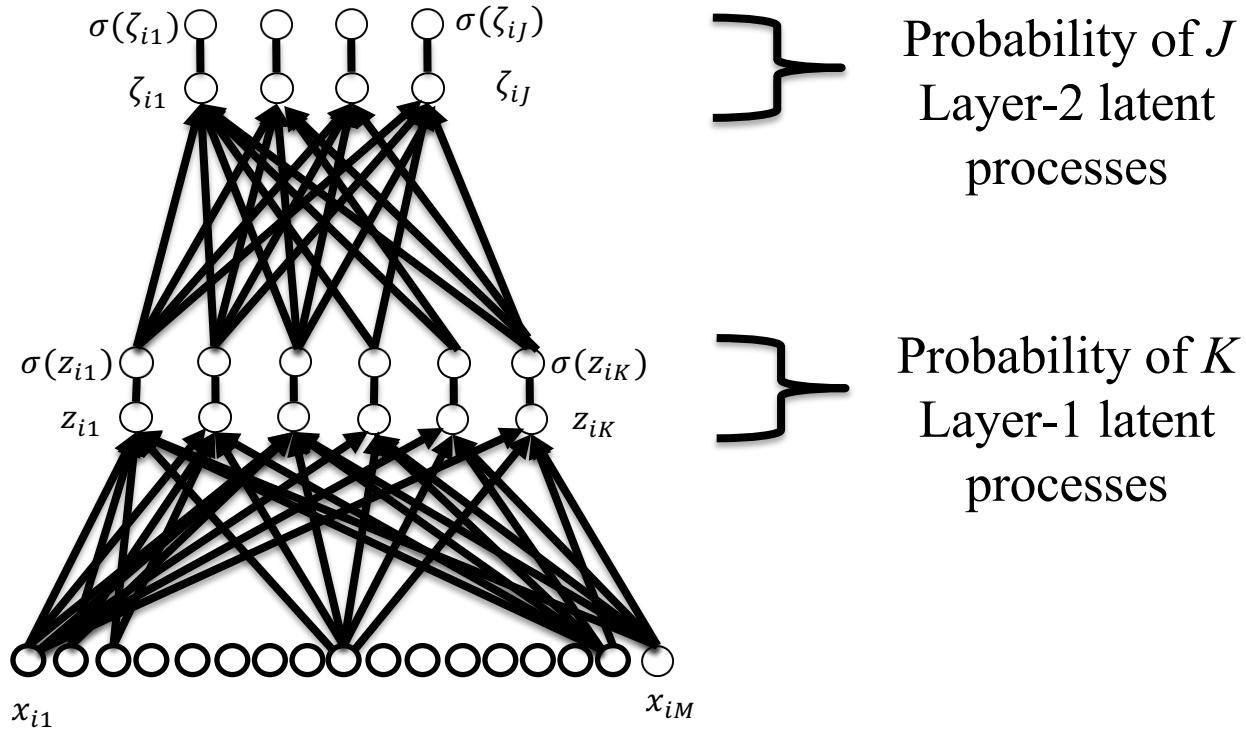
K Features output from Layer-1

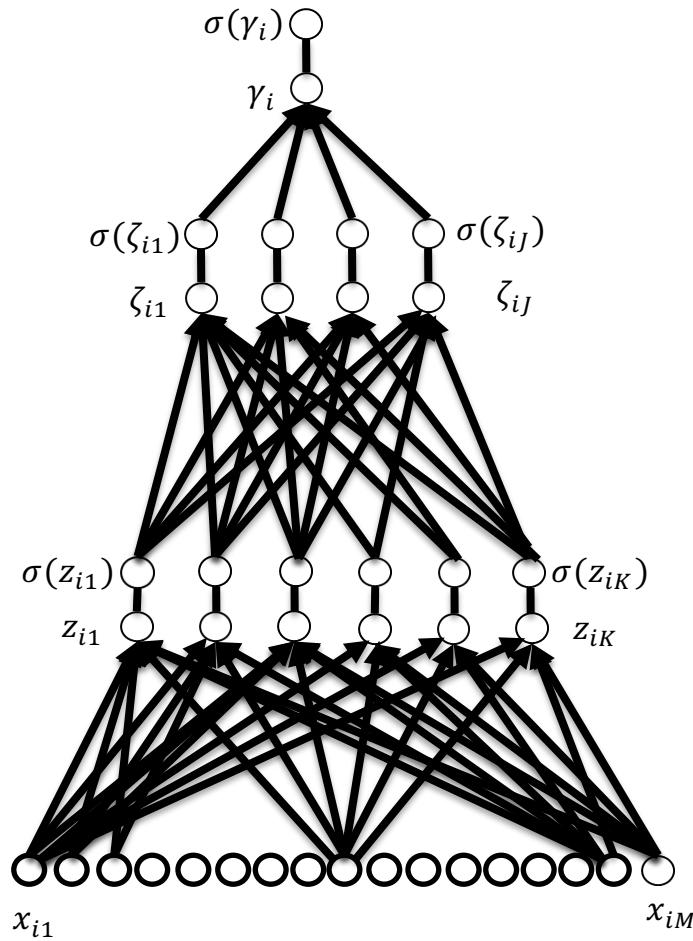
K Layer-1 filters

Extended Logistic Regression



Going Deeper

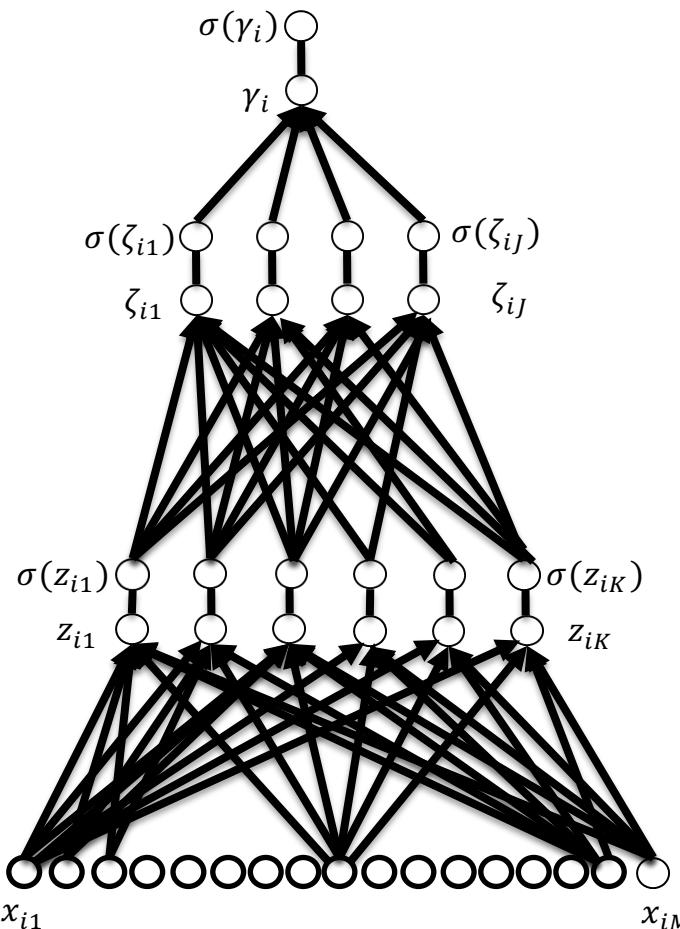




Probability of particular outcome

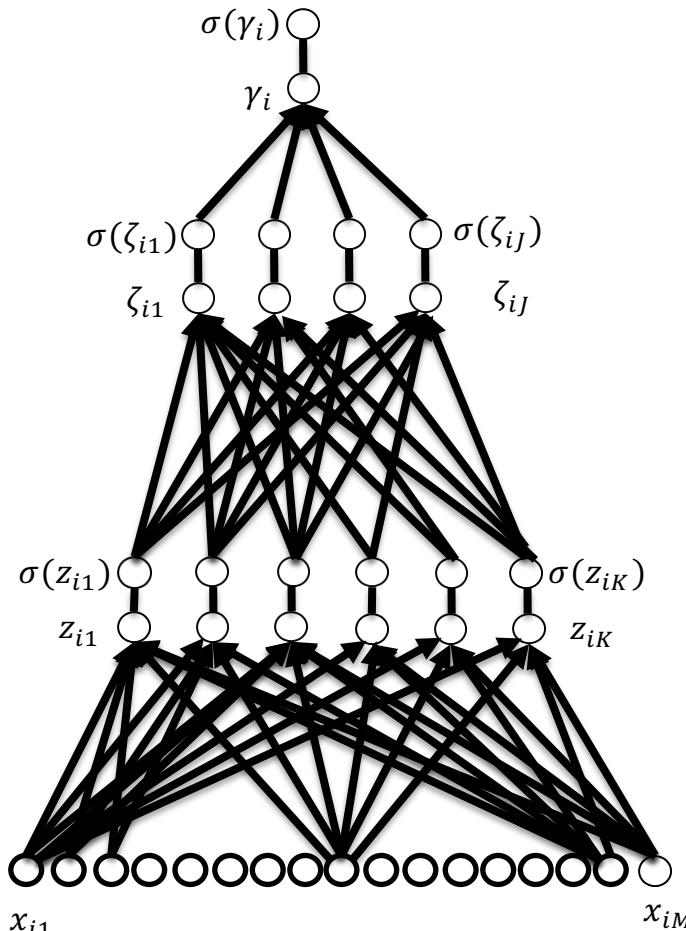
Probability of J Layer-2 latent processes

Probability of K Layer-1 latent processes



$$\begin{aligned}
 z_{i1} &= b_{01} + x_i \odot b_1 \\
 z_{i2} &= b_{02} + x_i \odot b_2 \\
 &\vdots \\
 z_{iK} &= b_{0K} + x_i \odot b_K
 \end{aligned}$$

Features of the
Data with Layer-1
Filters

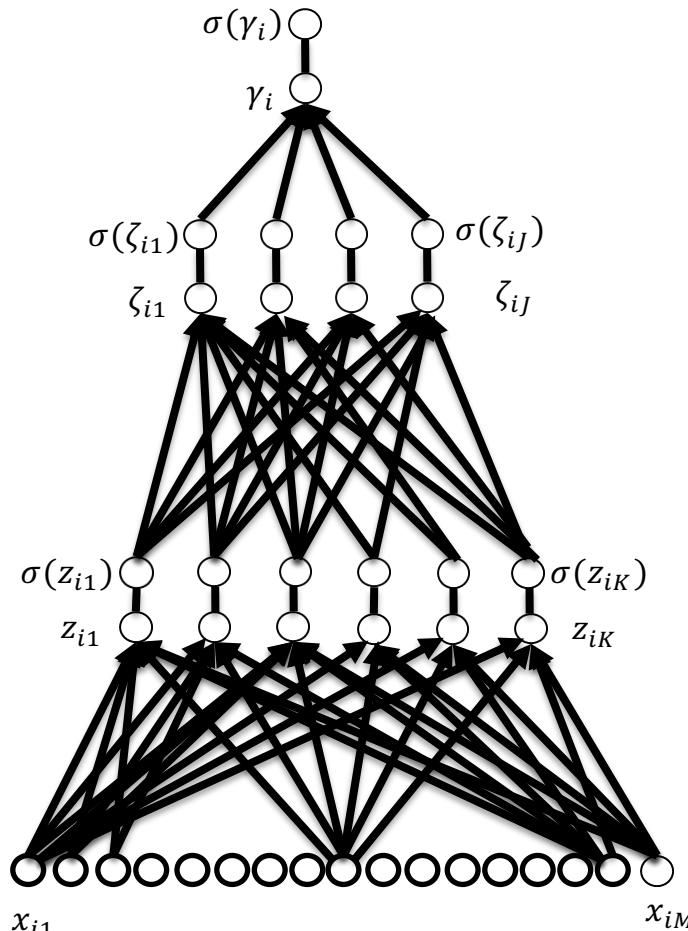


$$\begin{aligned}\zeta_{i1} &= c_{01} + \sigma(z_i) \odot c_1 \\ \zeta_{i2} &= c_{02} + \sigma(z_i) \odot c_2 \\ &\vdots \\ \zeta_{iK} &= c_{0J} + \sigma(z_i) \odot c_J\end{aligned}$$

$$\begin{aligned}z_{i1} &= b_{01} + x_i \odot b_1 \\ z_{i2} &= b_{02} + x_i \odot b_2 \\ &\vdots \\ z_{iK} &= b_{0K} + x_i \odot b_K\end{aligned}$$

Features of the
Layer-1 Features,
with Layer-2 Filters

Features of the
Data with Layer-1
Filters



$$\gamma_i = d_0 + \sigma(\zeta_i) \odot d$$

Logistic Regression
Based on the Layer-2
Features

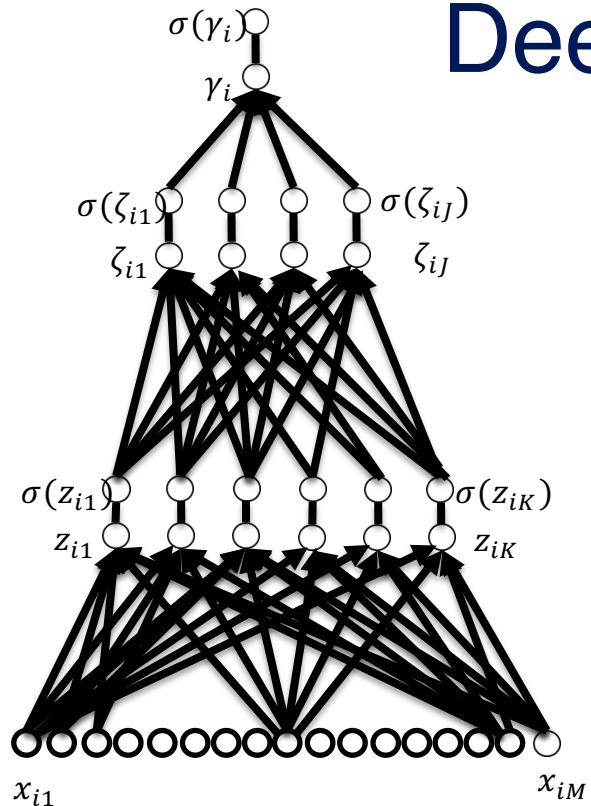
$$\begin{aligned}\zeta_{i1} &= c_{01} + \sigma(z_i) \odot c_1 \\ \zeta_{i2} &= c_{02} + \sigma(z_i) \odot c_2 \\ &\vdots \\ \zeta_{iK} &= c_{0J} + \sigma(z_i) \odot c_J\end{aligned}$$

Features of the
Layer-1 Features,
with Layer-2 Filters

$$\begin{aligned}z_{i1} &= b_{01} + x_i \odot b_1 \\ z_{i2} &= b_{02} + x_i \odot b_2 \\ &\vdots \\ z_{iK} &= b_{0K} + x_i \odot b_K\end{aligned}$$

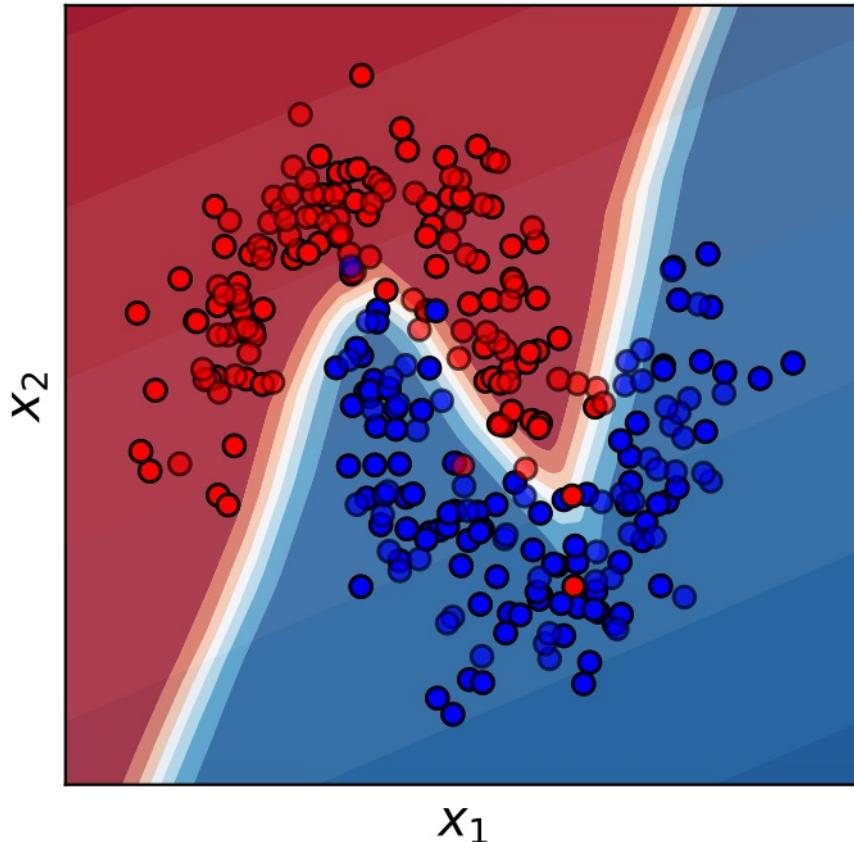
Features of the
Data with Layer-1
Filters

Multilayer Perceptron: Neural Network Deep Learning

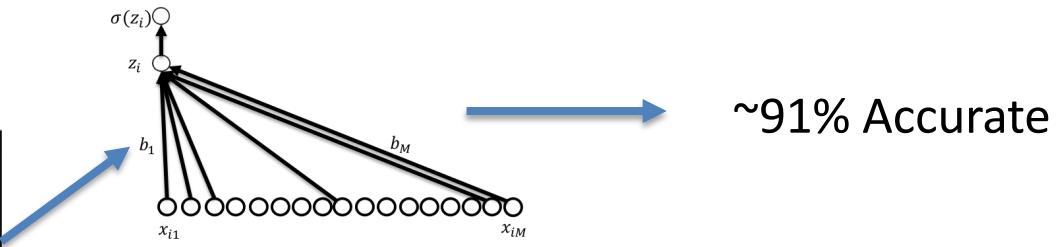


Can learn non-linear classifications

By learning multiple layers and transformation, it is possible to have a non-linear classifier capable of more accurately capturing the data.

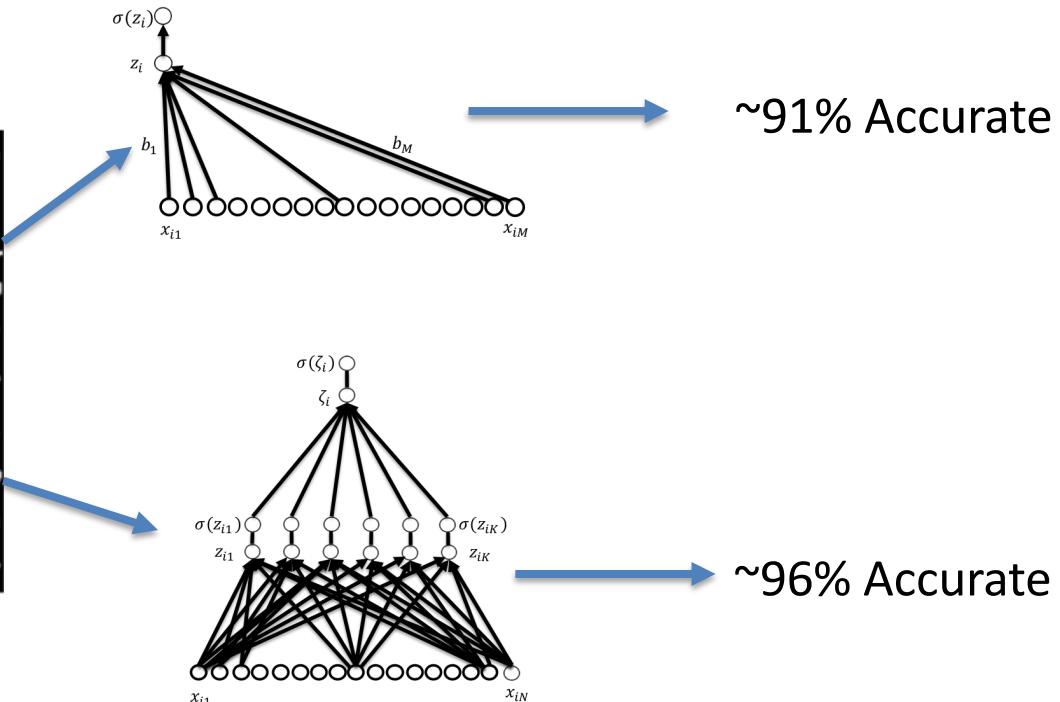


Does this work with MNIST?



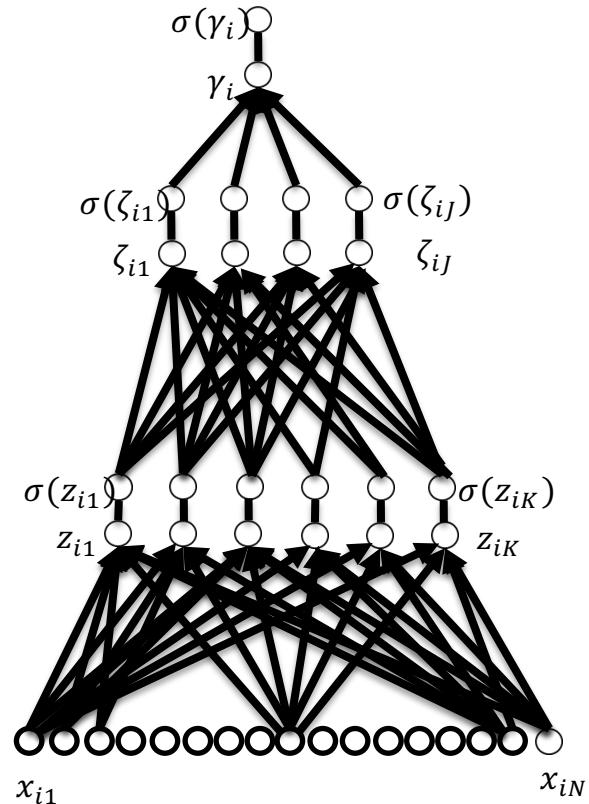
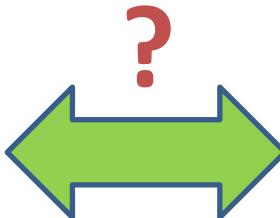
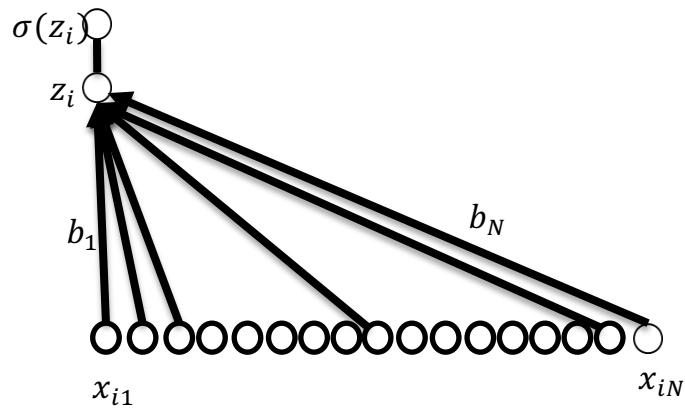
$\sim 91\%$ Accurate

Does this work with MNIST?



Disclaimer: will need a slight modification to the network to predict 1 of 10 digits.

Do we always want to increase complexity?



Thanks for your attention!

**BREAK; WHEN WE RETURN WE'LL
DISCUSS HOW THE NETWORK IS
LEARNED**