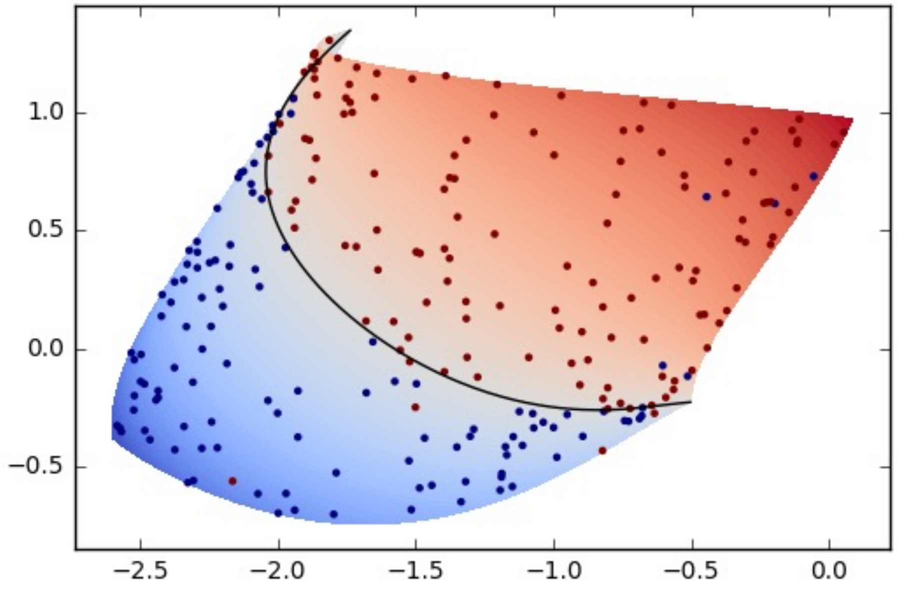
# Lecture 10: Logistic Classifiers



Gavin Kerrigan

Spring 2023

Some materials courtesy Padhraic Smyth, Alex Ihler.

## Announcements

- No in-person lectures this week (M, W, F)
  - Recorded and posted to Canvas

- HW2 due on Friday
  - Should be able to complete after today’s lecture

Midterm & Projects

Logistic Classifiers

Binary Cross-Entropy Loss

Gradient for Logistic Classifiers

## Midterm Exam

- Next Friday (5/5)
  - In class, during regular lecture time
  - 50 minute exam
  - Closed book, no notes
  - No electronic devices (calculator, phone, etc.)
- Sample exam + solutions will be available
- How should you study for this exam?
  - Review lecture slides, HWs, discussion materials
  - Be able to reproduce all derivations from lecture
  - Be able do all algorithms "by hand"
    - i.e. fit models, make predictions with models

## Class Projects

For the projects, you will:

- Create a team of 3 students
  - Worth 10% of your project grade to form a team on time
  - See Canvas page for team finding spreadsheet
- Select a dataset (all classification tasks)
  - One tabular dataset, two image datasets, one text dataset
  - More info next slide
- Train, evaluate, and produce some insight on this dataset with several different classification models
  - See Canvas for details and ideas

Important dates/deadlines:

- Team formation deadline: end of week 7 (Friday 5/19)
- Project deadline: Monday of finals week (6/12)

## Class Projects

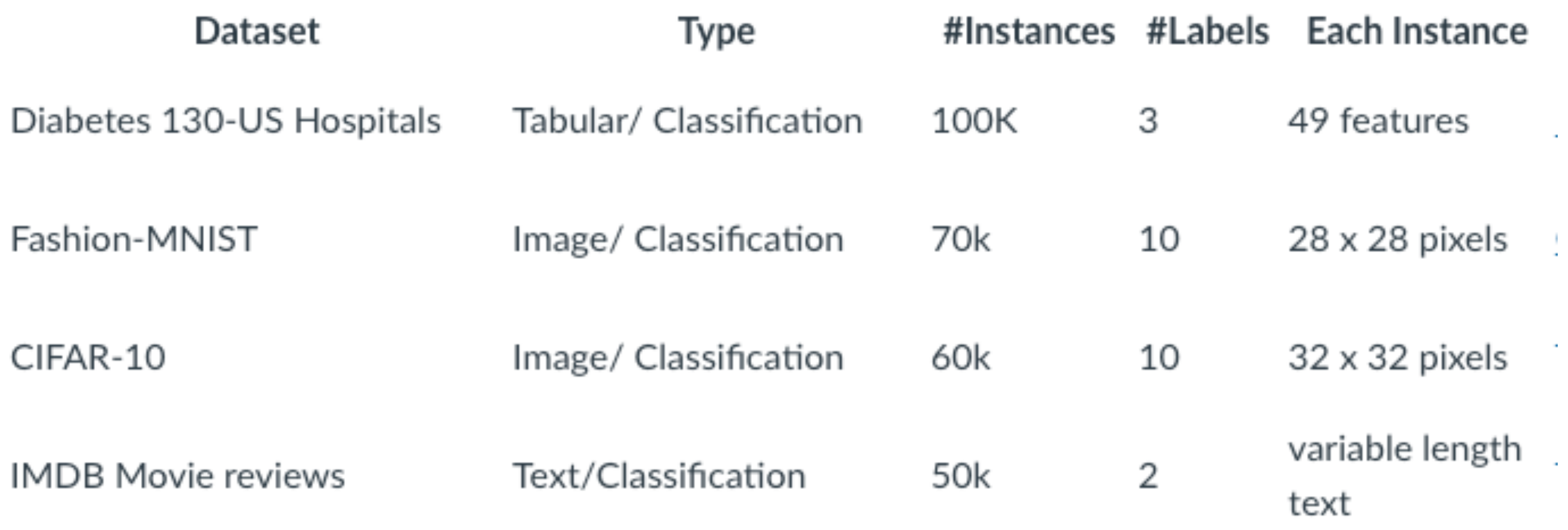
Main goals of the project:

- for you to learn about applying ML models to real datasets
- to gain insight and understanding about strengths/weaknesses of the models we’ve seen (and will later see) in class

The goal is not to build the world’s most accurate classifier

- Insights and understanding more important

## Class Projects: Data Sets



| Dataset | Type | #Instances | #Labels | Each Instance |
|---|---|---|---|---|
| Diabetes 130-US Hospitals | Tabular/ Classification | 100K | 3 | 49 features |
| Fashion-MNIST | Image/ Classification | 70k | 10 | 28 x 28 pixels |
| CIFAR-10 | Image/ Classification | 60k | 10 | 32 x 32 pixels |
| IMDB Movie reviews | Text/Classification | 50k | 2 | variable length text |

## Class Projects: Classification Algorithms

**At least 4 classifiers**

3 of which are: kNN, logistic, and feedforward neural network

4th: your choice, e.g., random forests, more complex neural network, etc

For the image and text classification datasets you could try
fairly complex models, e.g.,
-> convolutional neural networks for images
-> recurrent neural networks for text

Midterm & Projects

Logistic Classifiers

Binary Cross-Entropy Loss

Gradient for Logistic Classifiers

## Using Probabilities for Classification

Say we want to predict a binary label y from a feature vector **x**

How can we model the uncertainty in our prediction?

That is, we are interested in the *conditional probabilities*

$$p(y = 1 \mid \boldsymbol{x})$$

$$p(y = 0 \mid \boldsymbol{x}) = 1 - p(y = 1 \mid \boldsymbol{x})$$

- Sufficient to model $p(y = 1 \mid x)$
- Classifiers we have seen so far (kNN, nearest centroids) only produce labels, not probabilities
- We need a new model!

# Logistic Classifier Intuition

Key idea:

for every feature vector **x**, produce a numerical *score* $f(\boldsymbol{x} \mid \theta)$

- Here $\theta$ represents some parameters of our model
- High values for $f(\boldsymbol{x} \mid \theta)$ indicate y=1 is more likely
- Low values for $f(\boldsymbol{x} \mid \theta)$ indicate y=0 is more likely
- $f(\boldsymbol{x} \mid \theta)$ can take arbitrary real values

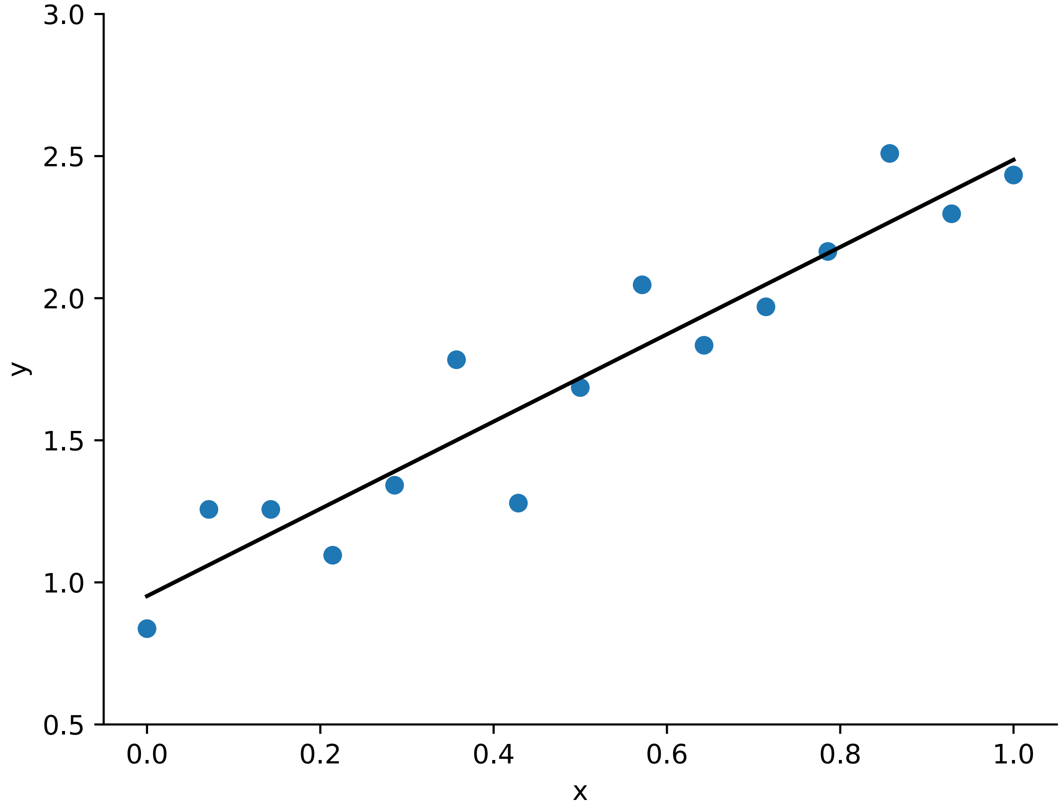
To model $p(y = 1 \mid \boldsymbol{x})$, we then need to turn our score into a probability

- Recall probabilities are between 0 and 1
- So we will need to somehow transform our score into this range

## Reminders on Linear Regression

Recall that in *regression* problems we are predicting a real-valued output

- *Linear regression* models fit a line (or hyperplane) to our data



Equation of a line:

$$f(x\,|\theta) = \theta_0 + \theta_1 x$$

Fit by minimizing the MSE

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i|\theta))^2$$

(e.g. via gradient descent)

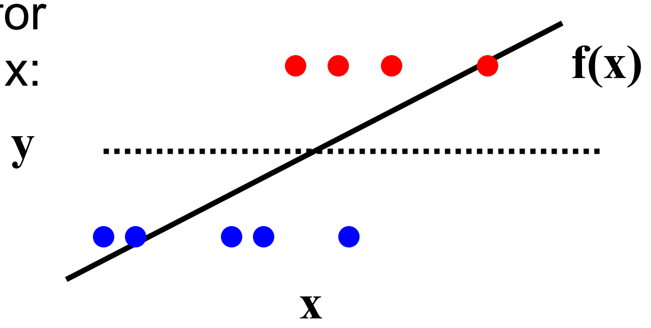
## Linear Classifiers

Recall that in *regression* problems we are predicting a real-valued output

- *Linear regression* models fit a line (or hyperplane) to our data
- What if we just did linear regression on binary labels $y \in \{0, 1\}$?

Visualization with a single feature x: $f(x \mid \theta) = \theta_0 + \theta_1 x$

Visualizing for one feature x:



## Linear Classifiers

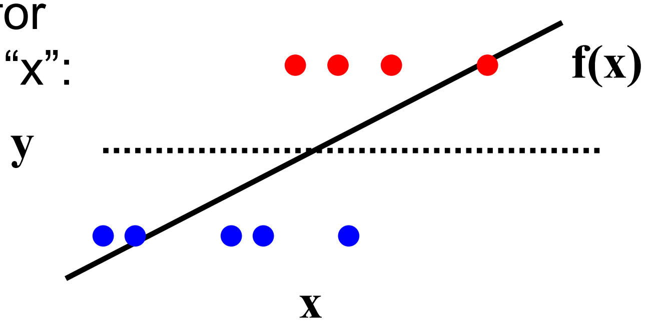
What if we just did linear regression on binary labels $y \in \{0, 1\}$?

Visualization with a single feature x: $f(x \mid \theta) = \theta_0 + \theta_1 x$

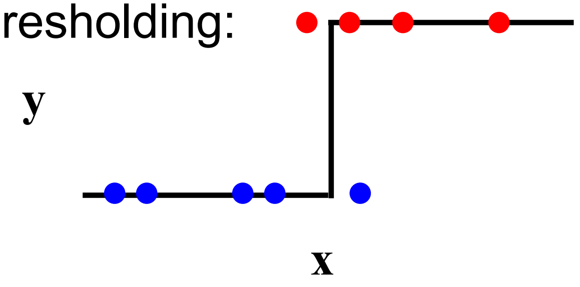
We can turn the real-valued outputs of *f* into 0/1 classifications by *thresholding*

$$\hat{y} = T(x \mid \theta) = \begin{cases} 1 & \text{If } f(x \mid \theta) > 0 \\ 0 & \text{If } f(x \mid \theta) < 0 \\ ? & \text{If } f(x \mid \theta) = 0 \text{ (we are on the decision boundary)} \end{cases}$$

Visualizing for one feature "x":

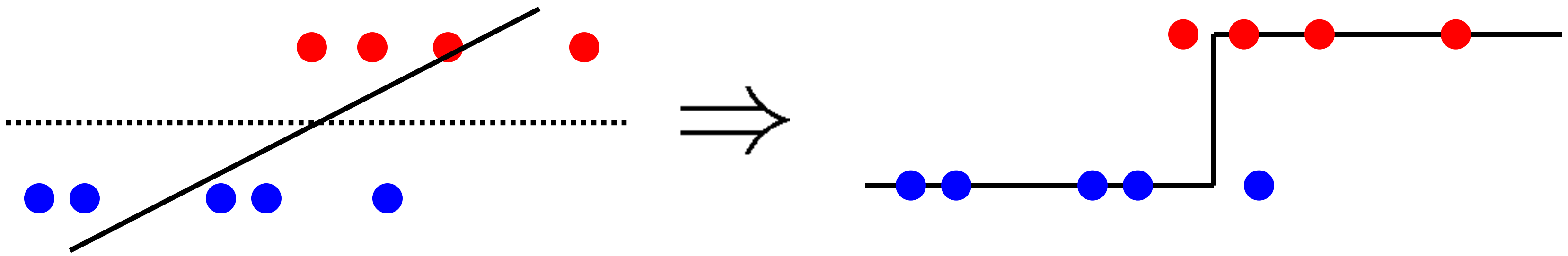


After thresholding:



# The Problem with our Linear Model

The type of boundary we are learning with our linear model



We could try fitting this model by minimizing MSE:

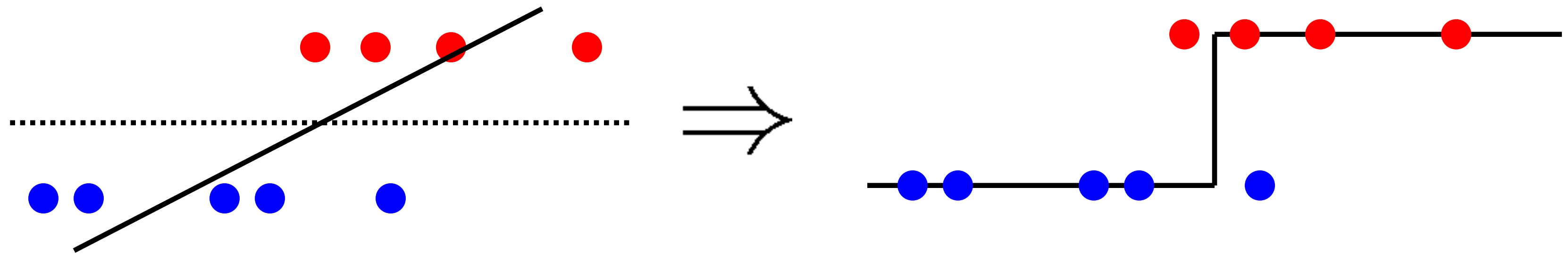
$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - T(\boldsymbol{x}_i|\theta))^2$$

Question: what is potentially wrong with this?

- Hint: how do we actually minimize MSE?

# The Problem with our Linear Model

The type of boundary we are learning with our linear model



We could try fitting this model by minimizing MSE:

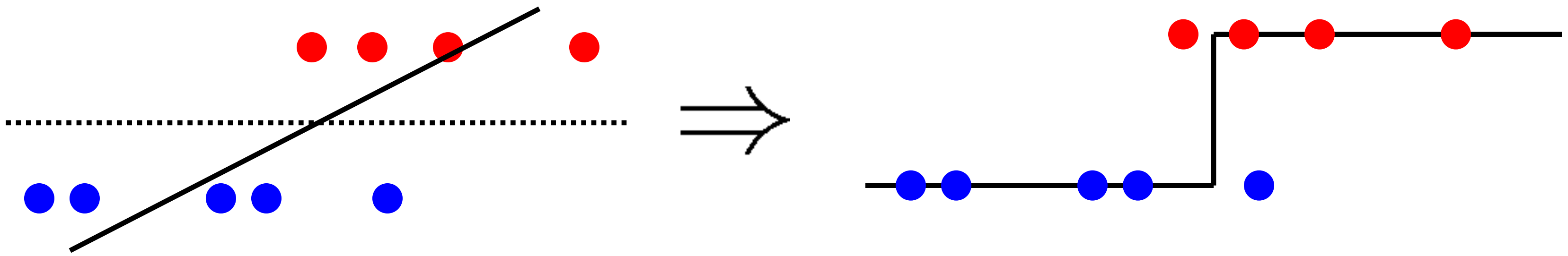
$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - T(\boldsymbol{x}_i|\theta))^2$$

Question: what is potentially wrong with this?

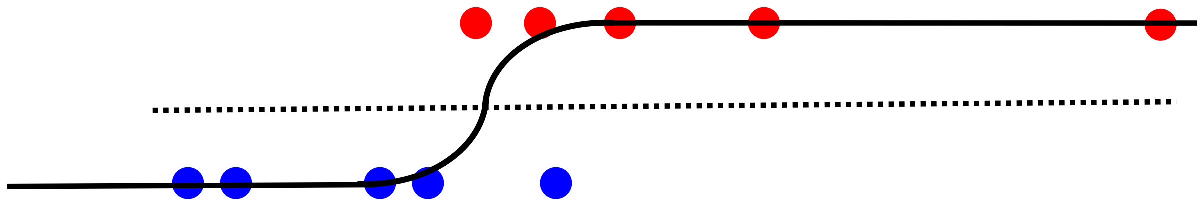
- Threshold function T has derivative zero! Can't learn with gradient descent.

# Soft Thresholding

The type of boundary we are learning with our linear model



Alternative: create a “soft threshold” that is differentiable



- Called a *sigmoid* or *logistic* curve
- Has the benefit of lettings us interpret model outputs as probabilities
- Used in many modern ML models (e.g. neural networks)

# Notation for Logistic Model

Assume binary classification (2 classes, y=0 or y=1)

Model output $f(\mathbf{x} \mid \theta)$ will lie between 0 and 1

- Will model $\mathrm{f}(\mathrm{x} \mid \theta) \approx p(y = 1 \mid x)$

We can turn our probabilities into class label predictions as follows:

- if $f(\mathbf{x}; \boldsymbol{\theta}) > 0.5$ predict class label 1
- if $f(\mathbf{x}; \boldsymbol{\theta}) < 0.5$ predict class label 0
- if $f(\mathbf{x}; \boldsymbol{\theta}) = 0.5$ we are on the decision boundary (toss a coin)

# Mathematical Definition of Logistic Classifier

Terminology note: this model is also referred to as "logistic regression"

Let $z(\mathbf{x};\boldsymbol{\theta}) = \theta_0 + \sum_{j=1}^{d} \theta_j x_j$ $\longleftarrow$ This is the same model as in linear regression

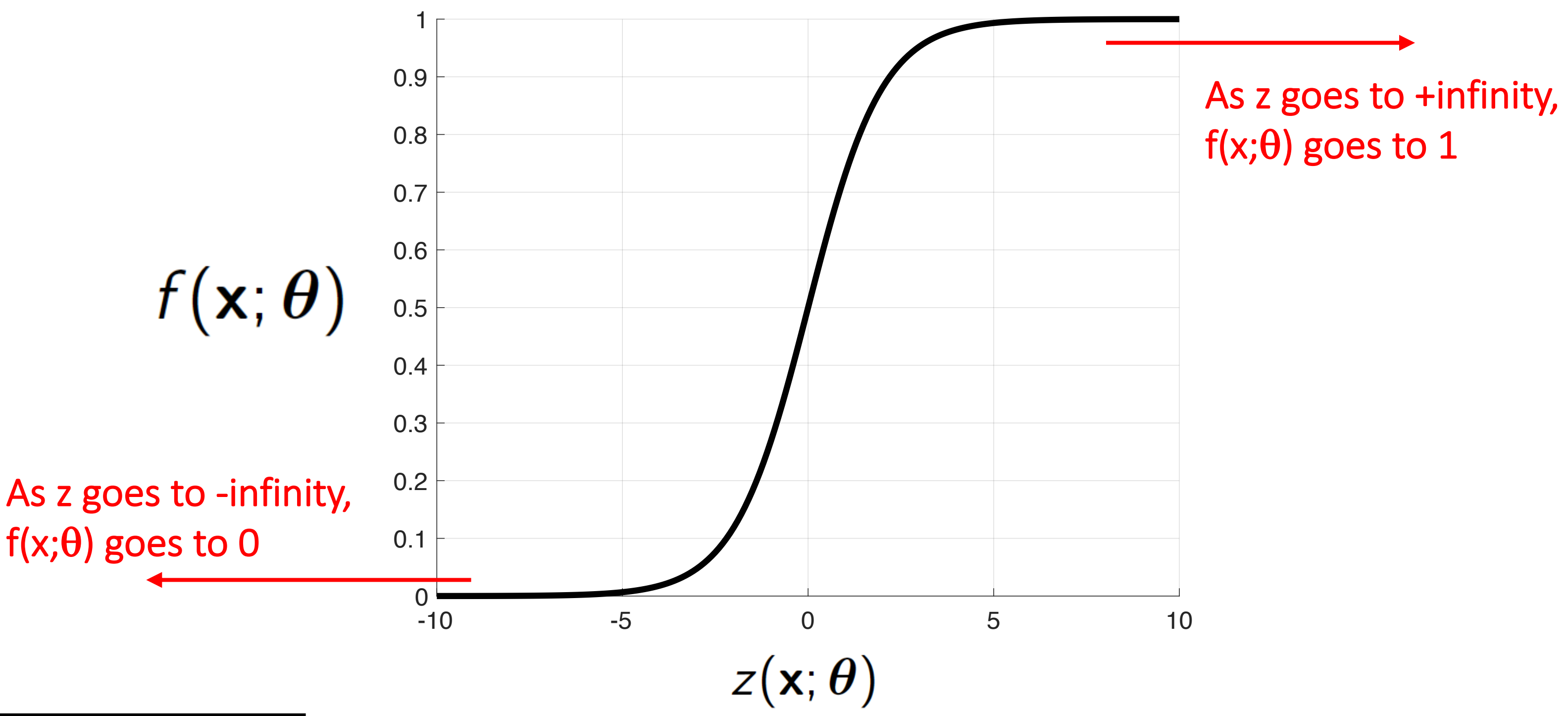
and $f(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{1+e^{-z(\mathbf{x};\boldsymbol{\theta})}}$ $\longleftarrow$ This is our new function (or model) f

How does function $f$ behave?

$$\text{As } z(\mathbf{x};\boldsymbol{\theta}) \to \infty \quad \Rightarrow \quad e^{-z(\mathbf{x};\boldsymbol{\theta})} \to 0 \quad \Rightarrow \quad f(\mathbf{x};\boldsymbol{\theta}) \to \frac{1}{1+0} = 1$$

$$\text{As } z(\mathbf{x};\boldsymbol{\theta}) \to -\infty \quad \Rightarrow \quad e^{-z(\mathbf{x};\boldsymbol{\theta})} \to \infty \quad \Rightarrow \quad f(\mathbf{x};\boldsymbol{\theta}) \to \frac{1}{1+\infty} = 0$$

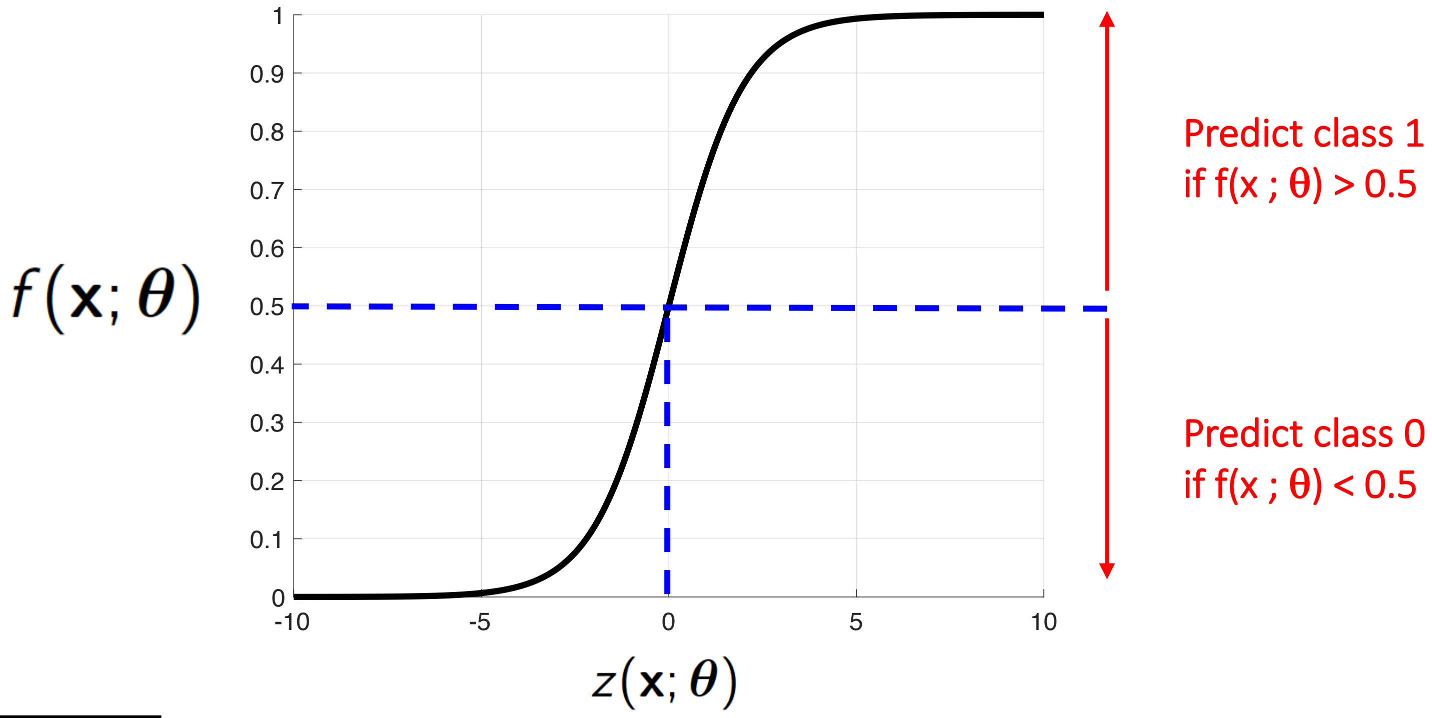
# Visualization of Logistic Classifier



$$z(\mathbf{x};\boldsymbol{\theta}) = \theta_0 + \sum_{j=1}^{d} \theta_j x_j$$

$$f(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{1 + e^{-z(\mathbf{x};\boldsymbol{\theta})}}$$

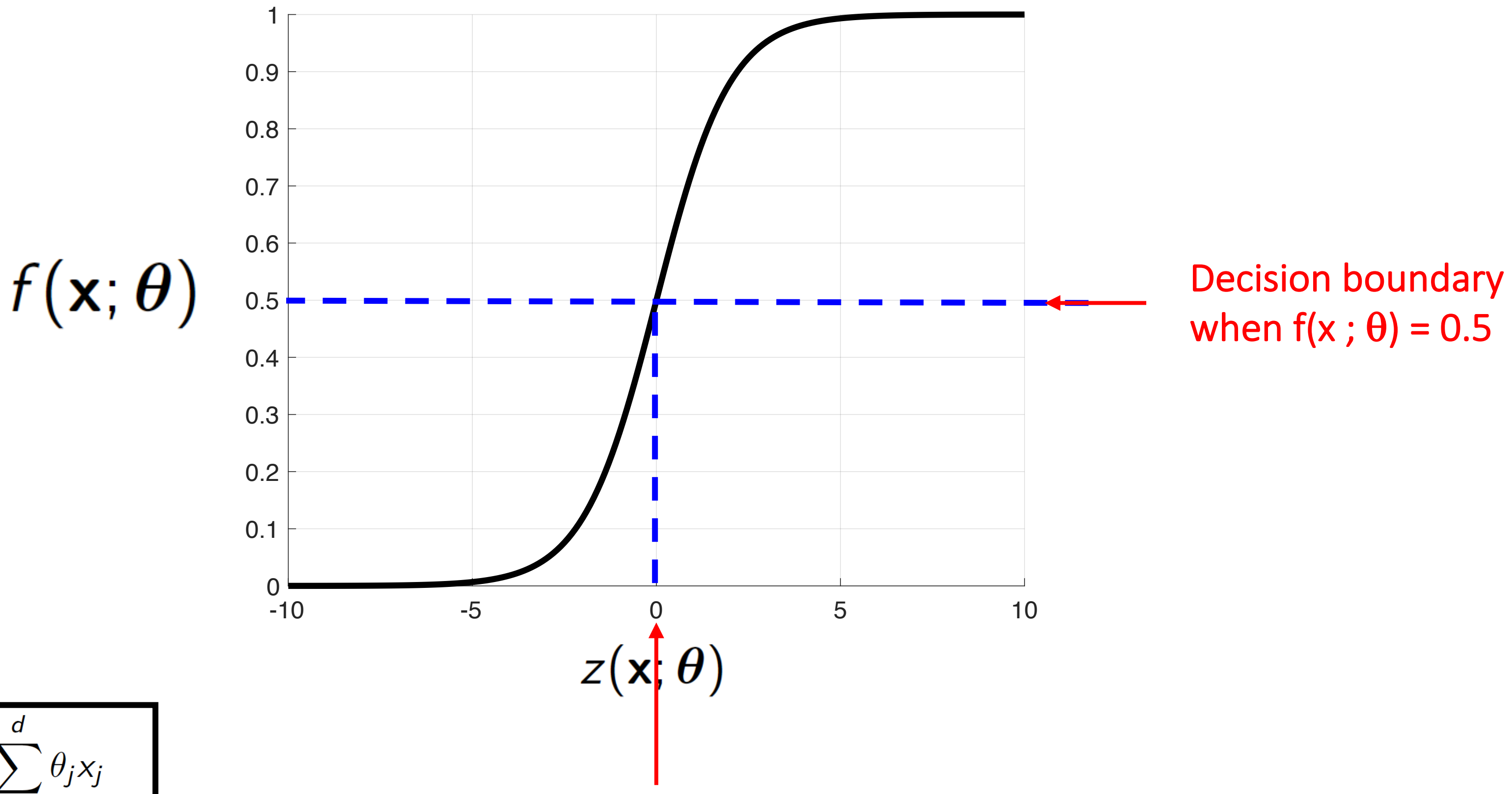
# Predictions with the Logistic Classifier



$$z(\mathbf{x};\boldsymbol{\theta}) = \theta_0 + \sum_{j=1}^{d} \theta_j x_j$$

$$f(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{1 + e^{-z(\mathbf{x};\boldsymbol{\theta})}}$$

# Predictions with the Logistic Classifier



$$z(\mathbf{x};\boldsymbol{\theta}) = \theta_0 + \sum_{j=1}^{d} \theta_j x_j$$

$$f(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{1 + e^{-z(\mathbf{x};\boldsymbol{\theta})}}$$

Equivalently:
Decision boundary
when z(x ; θ) = 0.0

## Decision Boundary for the Logistic Model

$f(\mathbf{x}; \boldsymbol{\theta}) = 0.5$ defines the **decision boundary**

How is the decision boundary defined in terms of $z$ and $\mathbf{x}$?

$$f(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{1 + e^{-z(\mathbf{x}; \boldsymbol{\theta})}} = 0.5$$

$$\Rightarrow 1 = 0.5(1 + e^{-z(\mathbf{x}; \boldsymbol{\theta})})$$

$$\Rightarrow e^{-z(\mathbf{x}; \boldsymbol{\theta})} = 1$$

$$\Rightarrow z(\mathbf{x}; \boldsymbol{\theta}) = 0$$

# Decision Boundary for the Logistic Model

So: the decision boundary occurs when the weighted sum $z(\mathbf{x}; \boldsymbol{\theta}) = 0$

In terms of **x** this can be written as

$$\theta_0 + \sum_{j=1}^{d} \theta_j x_j = 0$$

Note that this defines a **linear equation** in the $x_j$'s,

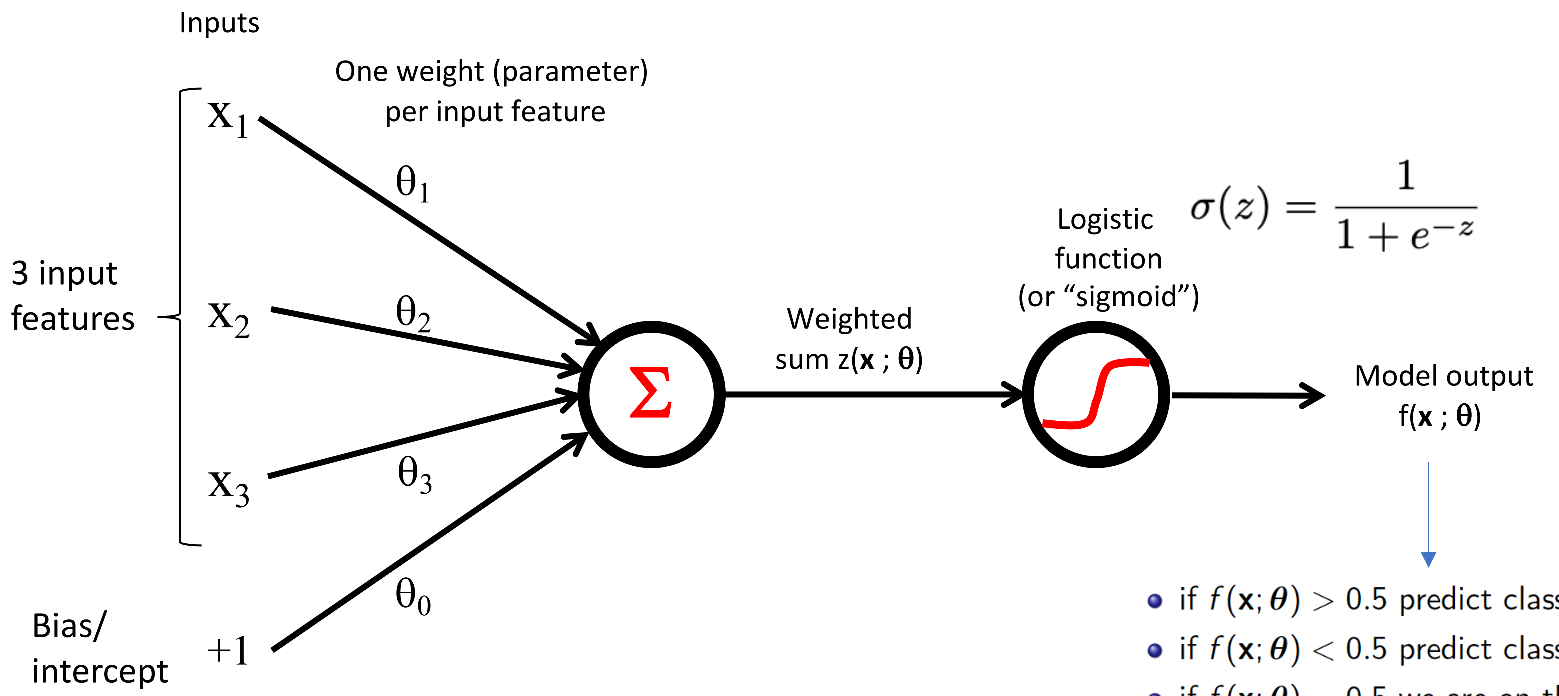
i.e., **the decision boundary of a logistic classifier is linear**
(lines, planes, hyperplanes)

For example with 2 features we have:

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$$

which can be expressed as $x_1 = a x_2 + b$.

# Block Diagram of a Logistic Classifier

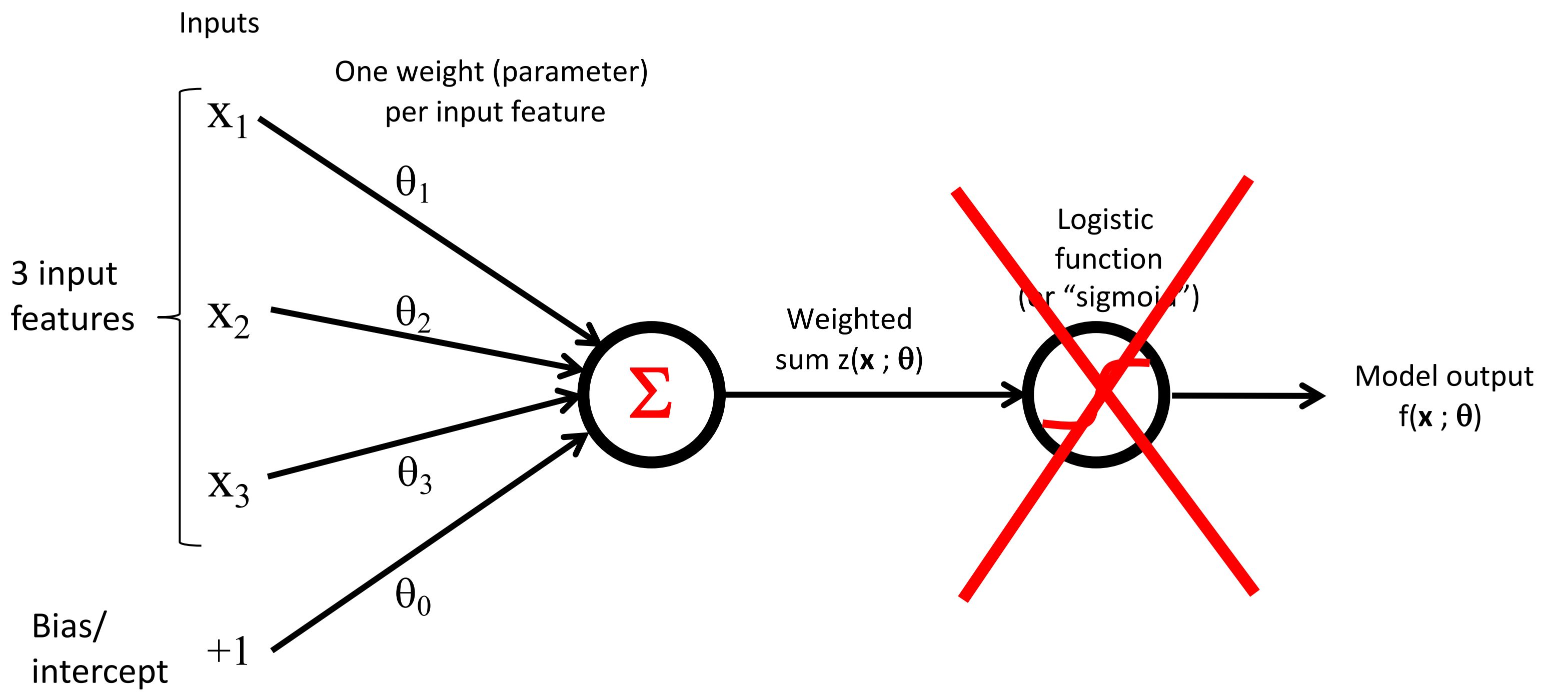


Inputs
One weight (parameter) per input feature
3 input features
$x_1$
$x_2$
$x_3$
$\theta_1$
$\theta_2$
$\theta_3$
$\theta_0$
Bias/ intercept
$+1$
$\Sigma$
Weighted sum z(**x** ; θ)
Logistic function (or "sigmoid")
Model output f(**x** ; θ)

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

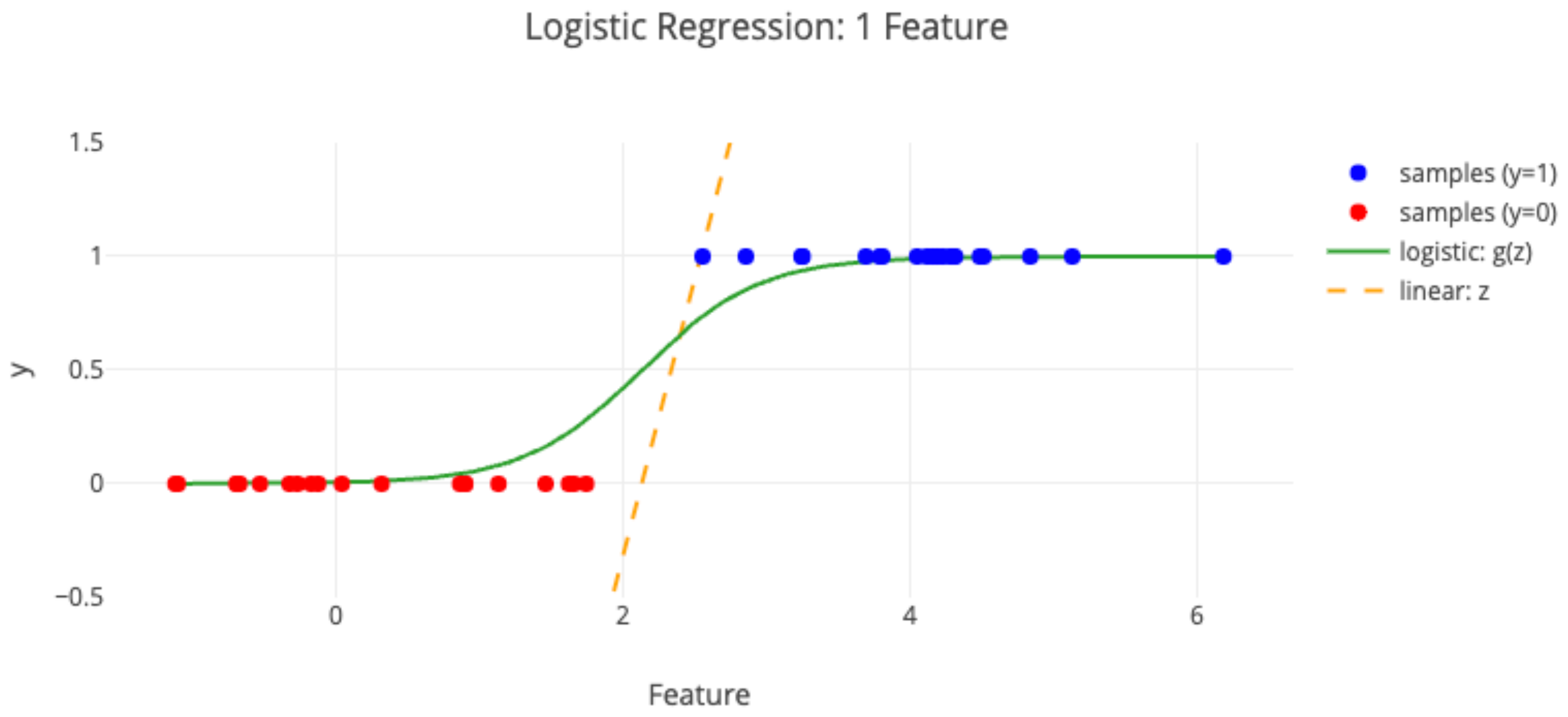
- if $f(\mathbf{x};\boldsymbol{\theta}) > 0.5$ predict class label 1
- if $f(\mathbf{x};\boldsymbol{\theta}) < 0.5$ predict class label 0
- if $f(\mathbf{x};\boldsymbol{\theta}) = 0.5$ we are on the decision boundary

# Block Diagram of the Model with No Sigmoid



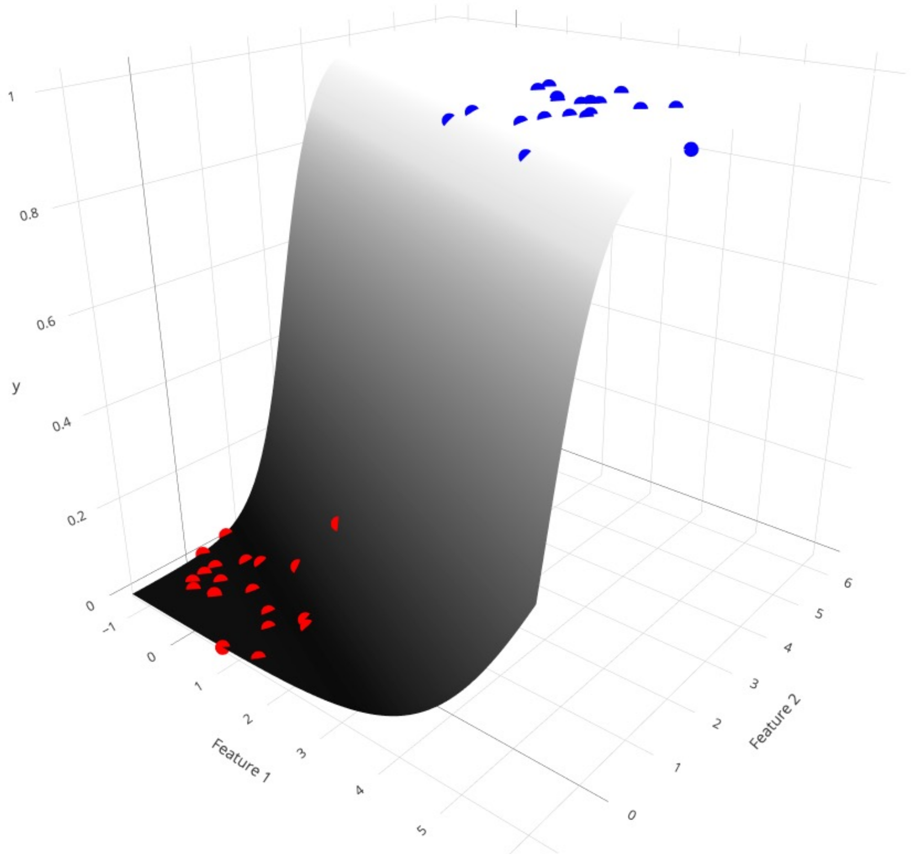
Inputs
One weight (parameter) per input feature
3 input features
Bias/ intercept
$x_1$
$x_2$
$x_3$
$+1$
$\theta_1$
$\theta_2$
$\theta_3$
$\theta_0$
$\Sigma$
Weighted sum z(**x** ; **θ**)
Logistic function (or "sigmoid")
Model output f(**x** ; **θ**)

# A Logistic Classifier with 1 Feature



From: https://florianhartl.com/logistic-regression-geometric-intuition.html

## A Logistic Classifier with 2 Features



From: https://florianhartl.com/logistic-regression-geometric-intuition.html

## Interpreting Logistic Outputs as Probabilities

We can treat the outputs $f(x\,;\,\theta)$ as probabilities

Specifically $f(\mathbf{x}\,;\,\theta)$ is modeling $P(y = 1 \mid \mathbf{x})$

i.e., the conditional probability that the class is 1 given features $\mathbf{x}$

This is a number between 0 and 1 that changes smoothly as x changes

It characterizes classifier confidence:

-> far away from the decision boundary it will be near 0 or 1

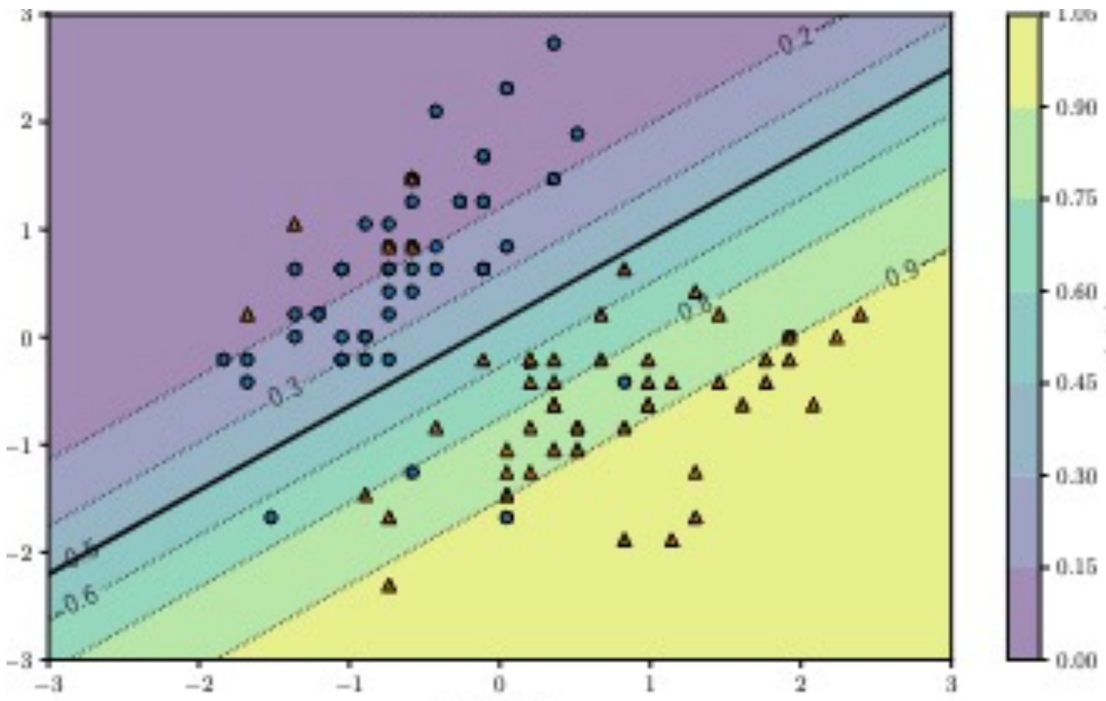
-> close to the decision boundary it will be closer to 0.5

# Class Probabilities from Logistic Model

Iris data in 2d with noise added

Contours of f(**x** ; θ) = P(class = 1 | **x**)

Feature 2



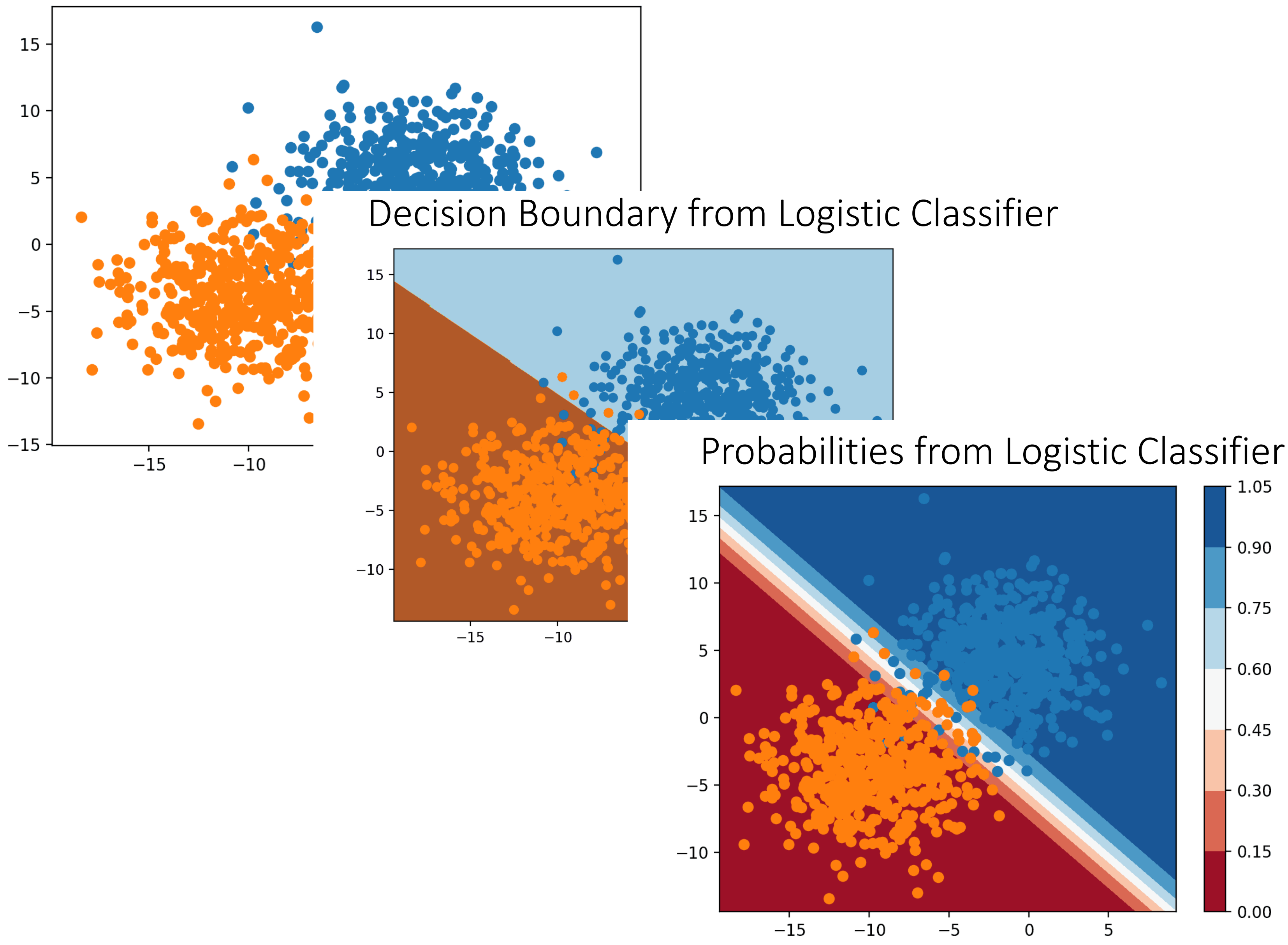
Feature 1

Color bar indicates probability values between 0 and 1

## Feature Data in 2 Dimensions

## Decision Boundary from Logistic Classifier

## Probabilities from Logistic Classifier



Example from https://machinelearningmastery.com/plot-a-decision-surface-for-machine-learning/

## Why are Class Probabilities Useful?

Allows classifier to indicate confidence in a “human-like” way

e.g., in autonomous driving

e.g., in medical image diagnosis

Allows classifier to perform ranking

e.g., in search: rank URLS that are most likely relevant to a user

e.g., in finance: rank loans that are most likely to be paid back

In general, class probabilities are widely used in machine learning not just by logistic models but also by neural networks and other models

However, getting calibrated probabilities is still a research problem

Getting machine learning systems to ”know what they don’t know”

## Questions?

Midterm & Projects

Logistic Classifiers

Binary Cross-Entropy Loss

Gradient for Logistic Classifiers

## How to Train Your ~~Dragon~~ Model

**What do we need to train a model?**

1. A definition of our function $f(\mathbf{x};\theta)$ <- this is the logistic model

2. A definition of a loss function

3. A definition of the gradients (everything is differentiable so its just a matter of working out what the derivatives are)

4. Apply our standard gradient descent algorithm: good to go!

## Loss Function for Logistic Models

We will use the **binary cross-entropy loss** (sometimes called log-loss)

- Don’t worry too much about the name: we’ll derive it in an intuitive way

For a binary classification problem:

- Have n training data points ($\mathbf{x}_i$, $y_i$)

- For each feature vector $\mathbf{x}_i$ our logistic classifier predicts a *probability* corresponding to the model’s estimate of p(y=1|x):

$$p_i = f(\boldsymbol{x_i} \mid \theta)$$

- When $y_i$ = 1, we want $p_i$ to be large (near one)
  - And when $y_i$ = 0, we want $p_i$ to be small (near zero)

- The predicted probability for the *correct class* for single datapoint $\mathbf{x}_i$ can then be written

$$p_i^{y_i}(1 - p_i)^{(1 - y_i)}$$

  - (this is a Bernoulli PMF, see probability review slides)

## Loss Function for Logistic Models

**Binary cross-entropy loss** (sometimes called log-loss)

- The predicted probability for the *correct class* for single datapoint $\mathbf{x}_i$ can then be written

$$p_i^{y_i}(1 - p_i)^{(1 - y_i)}$$

- Let's take the logarithm of this (for numerical stability reasons)

$$= y_i \log(p_i) + (1 - y_i)\log(1 - p_i)$$

- We then want to *maximize* the correct-class probability over our whole dataset; this is equivalent to *minimizing*

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

# Loss Function for Logistic Models

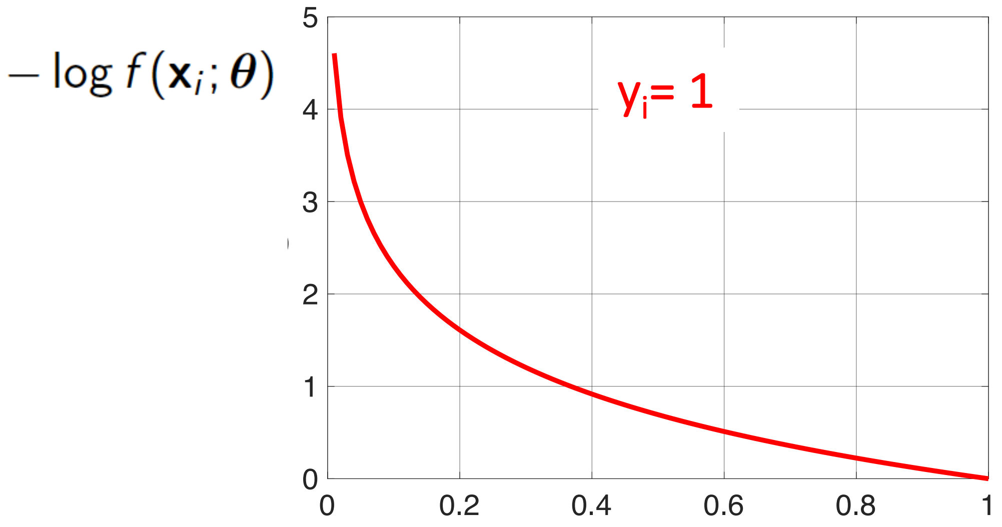
**Binary cross-entropy loss:** we seek to minimize

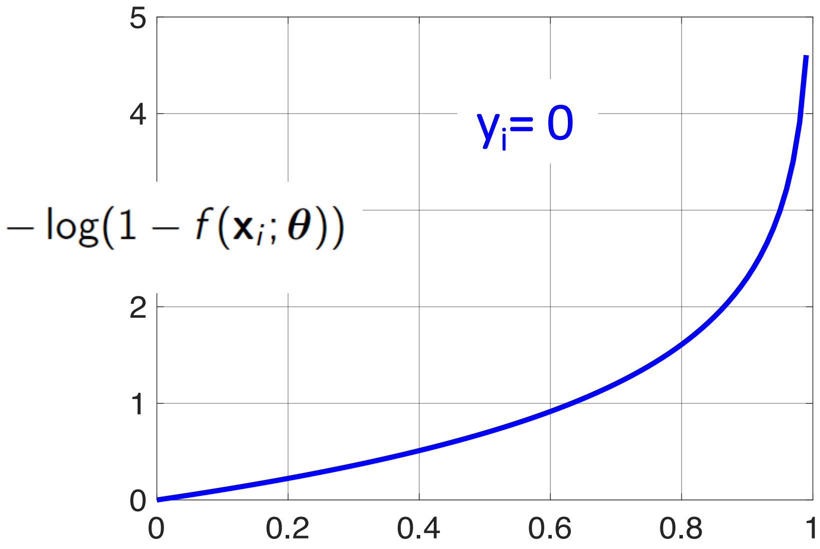
$$L_{BCE}(\theta) = \sum_{i=1}^{n} -y_i \log p_i - (1 - y_i) \log(1 - p_i)$$

$$= \sum_{i=1}^{n} -y_i \log\left(f(\boldsymbol{x}_i \mid \theta)\right) - (1 - y_i) \log(1 - f(\boldsymbol{x}_i \mid \theta))$$

- This is the loss function we will minimize in order to fit logistic classifiers
- Same loss function can be used for more complex models (e.g. neural networks)
- We will see an extension of this to multiple classes later

## Visualizing the Loss





Midterm & Projects

Logistic Classifiers

Binary Cross-Entropy Loss

Gradient for Logistic Classifiers

# How to Train Your ~~Dragon~~ Model

What do we need to train a model?

1. A definition of our function f(**x**;θ) <-  this is the logistic model

2. A definition of a loss function
- We will use the binary cross-entropy loss we just defined

**3. A definition of the gradients (everything is differentiable so its just a matter of working out what the derivatives are)**

4. Apply our standard gradient descent algorithm: good to go!

# Gradients for the Log-Loss with Logistic Model

Recall that the gradient vector for a loss function is a vector of partial derivatives, one per parameter

$$\nabla L(\boldsymbol{\theta}) = \left( \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_0}, \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_1}, \ldots, \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_d} \right)$$

For each parameter we need to work out the partial derivative of the cross-entropy loss when f( ) is the logistic model

We can do this using the chain rule for derivatives again (as with MSE)

## Gradients for the Logistic Model

$$\mathrm{L}_{BCE}(\theta) = \sum_{i=1}^{n} -y_i \log\left(f(\boldsymbol{x}_i \mid \theta)\right) - (1 - y_i) \log(1 - f(\boldsymbol{x}_i \mid \theta))$$

$$\frac{\partial}{\partial \theta_j} L_{\mathrm{BCE}}(\theta) = \sum_{i=1}^{n} -y_i \frac{\partial}{\partial \theta_j} \log f(\mathbf{x}_i \mid \theta) - (1 - y_i) \frac{\partial}{\partial \theta_j} \log(1 - f(\mathbf{x}_i \mid \theta))$$

Using the rule for differentiating logarithms, and the chain rule:

$$= \sum_{i=1}^{n} -\frac{y_i}{f(\mathbf{x}_i \mid \theta)} \frac{\partial}{\partial \theta_j} f(\mathbf{x}_i \mid \theta) + \frac{1 - y_i}{1 - f(\mathbf{x}_i \mid \theta)} \frac{\partial}{\partial \theta_j} f(\mathbf{x}_i \mid \theta)$$

We now just need to be able to calculate:

$$\frac{\partial}{\partial \theta_j} f(\mathbf{x}_i \mid \theta)$$

## Gradients for the Logistic Model

We now just need to be able to calculate:

$$\frac{\partial}{\partial \theta_j} f(\mathbf{x}_i \mid \theta) = \quad \frac{\partial}{\partial \theta_j} \sigma(z(\mathbf{x}_i \mid \theta))$$

Recall the definition of the sigmoid function:

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

Check for yourself using standard calculus that the (usual, ordinary) derivative is:

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

## Gradients for the Logistic Model

Then, by the chain rule:

$$\begin{aligned}\frac{\partial}{\partial\theta_j} f(\mathbf{x}_i \mid \theta) = & \quad \frac{\partial}{\partial\theta_j}\sigma(z(\mathbf{x}_i \mid \theta)) \\ = & \quad \sigma(z(\mathbf{x}_i \mid \theta))(1-\sigma(z(\mathbf{x}_i \mid \theta)))\frac{\partial}{\partial\theta_j} z(\mathbf{x}_i \mid \theta)\end{aligned}$$

Recall that $z(\, x_i \mid \theta \,)$ is a linear function:

$$z(\mathbf{x}_i \mid \theta) = \quad \theta_0 + \sum_{j=1}^{d} x_j\theta_j$$

$$\implies \quad \frac{\partial}{\partial\theta_j} z(\mathbf{x}_i \mid \theta) = x_{ij}$$

# Gradients for the Logistic Model

Plugging everything back in, and writing in terms of $p_i$

$$\begin{aligned}
\frac{\partial}{\partial\theta_j} L_{\mathrm{BCE}}(\theta) = \quad & \sum_{i=1}^{n} -y_i \frac{\partial}{\partial\theta_j} \log f(\mathbf{x}_i \mid \theta) - (1-y_i)\frac{\partial}{\partial\theta_j}\log(1 - f(\mathbf{x}_i \mid \theta)) \\
= \quad & \sum_{i=1}^{n} -\frac{y_i}{f(\mathbf{x}_i \mid \theta)} \frac{\partial}{\partial\theta_j} f(\mathbf{x}_i \mid \theta) + \frac{1-y_i}{1 - f(\mathbf{x}_i \mid \theta)} \frac{\partial}{\partial\theta_j} f(\mathbf{x}_i \mid \theta) \\
= \quad & \sum_{i=1}^{n} -\frac{y_i}{p_i}(p_i(1-p_i)x_{ij}) + \frac{1-y_i}{1-p_i}(p_i(1-p_i)x_{ij}) \\
= \quad & \sum_{i=1}^{n} -y_i(1-p_i)x_{ij} + (1-y_i)p_i x_{ij} \\
= \quad & \sum_{i=1}^{n} (p_i - y_i)x_{ij}
\end{aligned}$$

- After an ugly calculation, we get a relatively simple formula for the gradient

# Time-Complexity of Computing a Gradient Vector

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{n}\sum_{i=1}^{n}\left(f(\mathbf{x}_i;\boldsymbol{\theta}) - y_i\right)x_{ij}$$

Step 1: compute a prediction for each training example, which is O(nd), and subtract off each $y_i$. This is O(nd) in total. Produces a vector of length n

Step 2: for each parameter j, multiply each $x_{ij}$ elementwise by the result from Step 1 and compute a sum over the n datapoints. This is also O(nd)

So the total time-complexity is O(nd)

# Time-Complexity of Computing a Gradient Vector

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{n}\sum_{i=1}^{n}\left(f(\mathbf{x}_i;\boldsymbol{\theta}) - y_i\right)x_{ij}$$

Note that in practice its better not to do this in code with for-loops: ”vectorization” is typically much faster (e.g., using numpy), where we do as many operations as we can using vectors (e.g, use a vector of length n for all the n predictions)

The time-complexity with vectorization is still O(nd), but each of the nd operations is faster (optimized)

The total time-complexity for gradient descent is O(n d I ) where “I” is the total number of iterations until convergence (which we don’t of course know in advance)

## Summary

Logistic Classifiers (for binary classification problems) are models of the form

$$z(\mathbf{x};\boldsymbol{\theta}) = \theta_0 + \sum_{j=1}^{d} \theta_j x_j$$

$$f(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{1 + e^{-z(\mathbf{x};\boldsymbol{\theta})}}$$

- The model output $f(\boldsymbol{x} \mid \theta)$ can be thought of as the model's belief in $p(y = 1 \mid x)$

Learned by minimizing the *binary cross-entropy loss*:

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

- Use gradient descent to minimize; derived gradient on previous slides