# 10-701: Introduction to Machine Learning Lecture 7 – Logistic Regression

Pradeep Ravikumar

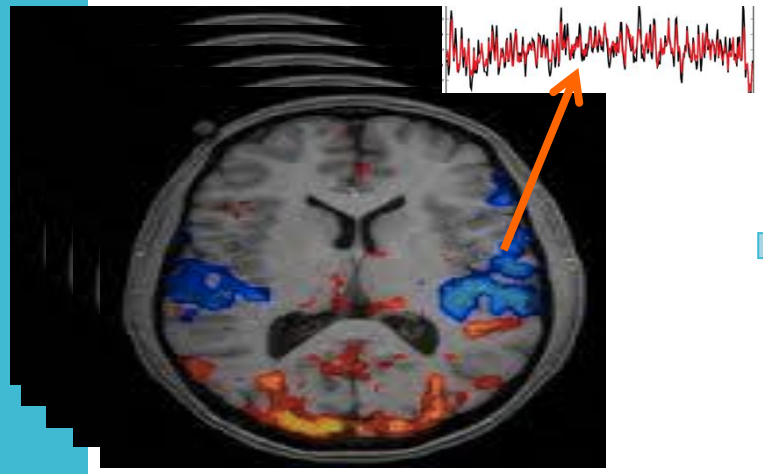
Spring 2026

# Front Matter

- Announcements:
  - Recitation this Friday
- Recommended Readings:
  - Murphy, Section 8.1 - 8.3

# Regression to Classification

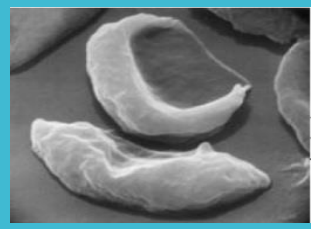
**Regression**

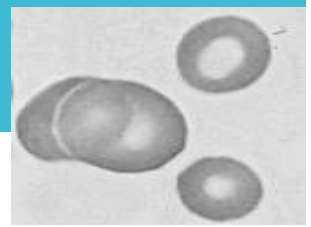


X = Brain Scan

Y = Age of a subject

**Classification**





X = Cell Image

Anemic cell
Healthy cell

Y = Diagnosis

Can we predict the "probability" of class label being Anemic or Healthy – a real number – using regression methods?

But output (probability) needs to be in [0,1]

# Classification

**Goal:** Construct a **predictor** $f : X \rightarrow Y$ to minimize a risk (performance measure) $R(f)$


Sports
Science
News

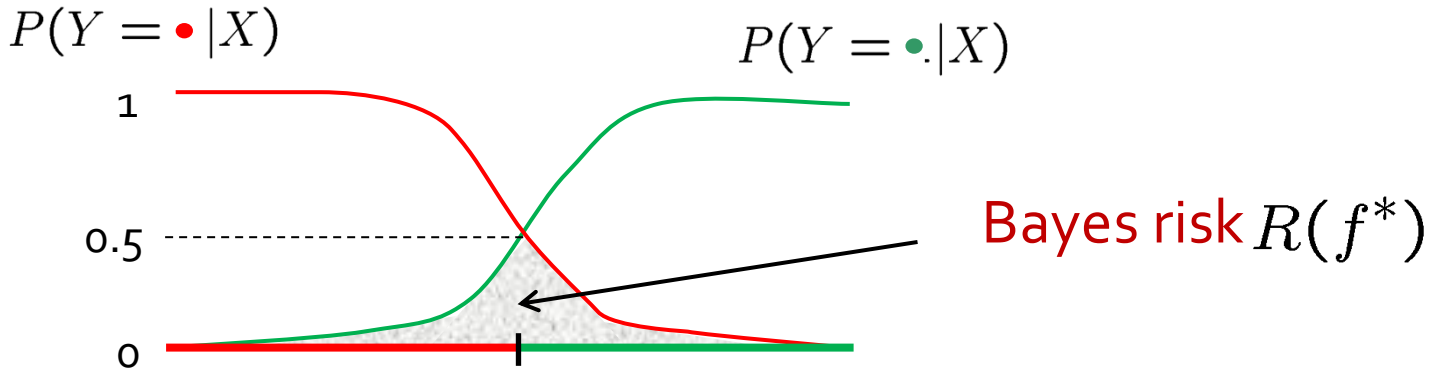
**Features, X** **Labels, Y**

$$R(f) = P(f(X) \neq Y)$$ **Probability of Error**

# Optimal Classification

Optimal predictor: (Bayes classifier)

$$f^* = \arg\min_f P(f(X) \neq Y)$$




$$f^*(x) = \arg\max_{Y=y} P(Y = y|X = x)$$

- Even the optimal classifier makes mistakes R(f*) > 0
- Optimal classifier depends on **unknown** distribution $P_{XY}$

# Optimal Classifier

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

**Bayes Rule:**

$$P(Y = y|X = x) = \frac{P(X = x|Y = y)P(Y = y)}{P(X = x)}$$

$$f^*(x) = \arg\max_{Y=y} P(Y = y|X = x)$$

$$= \arg\max_{Y=y} P(X = x|Y = y)P(Y = y)$$

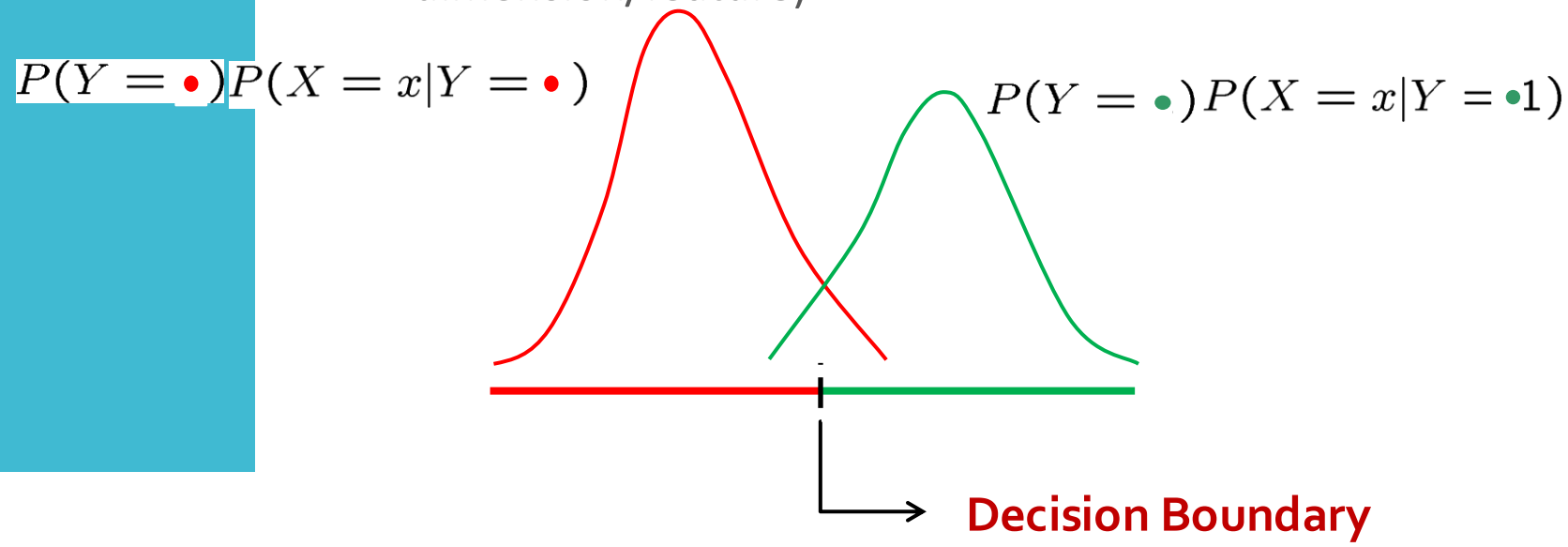
Class conditional density

Class prior

# Example Decision Boundaries

$$P(X = x|Y = y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x-\mu_y)^2}{2\sigma_y^2}\right)$$

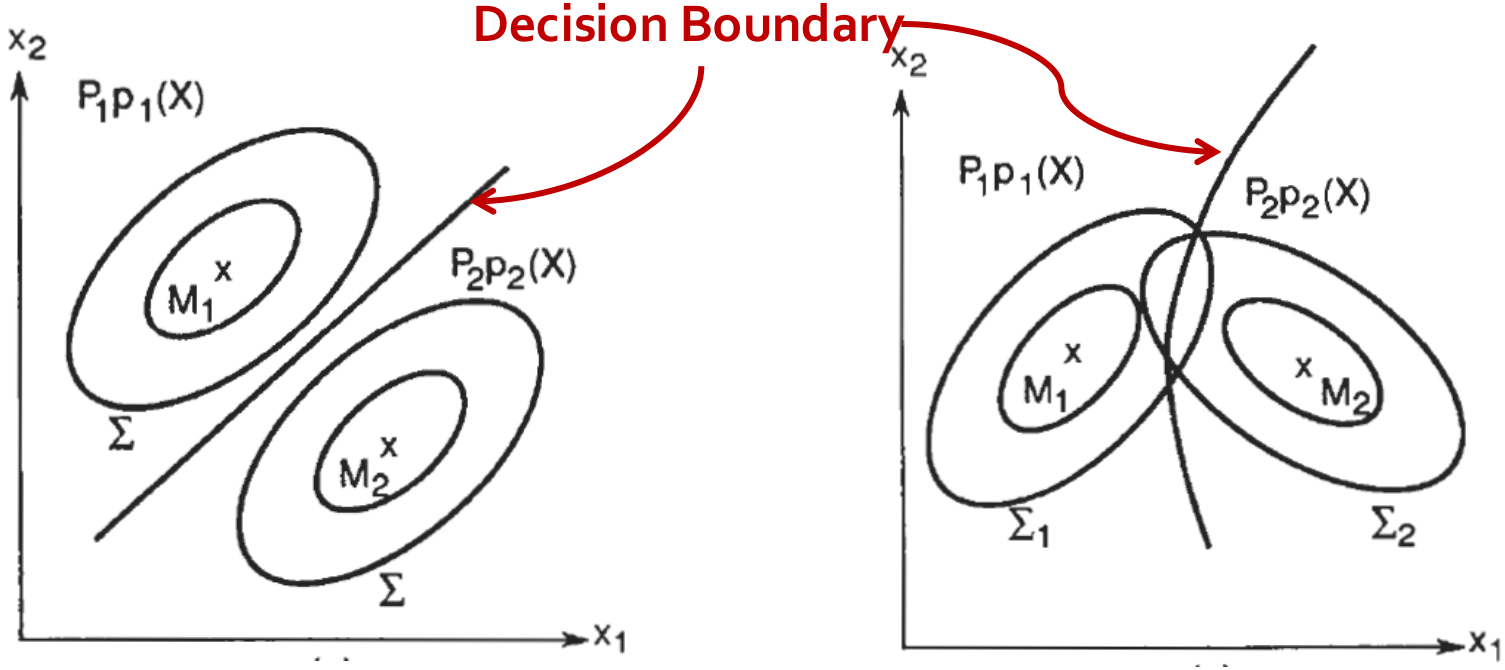
- Gaussian class conditional densities (1-dimension/feature)



# Example Decision Boundaries

- Gaussian class conditional densities (2-dimensions/features)

$$P(X = x|Y = y) = \frac{1}{\sqrt{2\pi|\Sigma_y|}} \exp\left(-\frac{(x-\mu_y)\Sigma_y^{-1}(x-\mu_y)'}{2}\right)$$



# Learning the Optimal Classifier

**Optimal classifier:**

$$f^*(x) = \arg\max_{Y=y} P(Y = y|X = x)$$

$$= \arg\max_{Y=y} \underbrace{P(X = x|Y = y)}_{\text{Class conditional density}} \underbrace{P(Y = y)}_{\text{Class prior}}$$

Need to know Prior P(Y = y) for all y

Likelihood P(X=x|Y = y) for all x,y

# Two truths and a lie

**Which statement is the LIE?**

- A. Bayes classifier: $f^*(x) = \text{argmax}_y P(Y = y \mid X = x)$
- B. Even the optimal classifier can make mistakes (Bayes risk $R(f^*) > 0$)
- C. The Bayes classifier does NOT depend on the (unknown) distribution of (X,Y)

Discuss with a neighbor for 20 seconds.

# Generative vs. Discriminative Classifiers

Generative classifiers

- Assume some functional form for P(X,Y) (or P(X|Y) and P(Y))
- Estimate parameters of P(X|Y), P(Y) directly from training data
- Use Bayes rule to calculate P(Y|X)

Discriminative classifiers

- Assume some functional form for P(Y|X) or for the decision boundary
- Estimate parameters of P(Y|X) directly from training data

# Probabilistic Classification

- Classification:
  - (Unknown) Target function, $c^*: \mathcal{X} \rightarrow \mathcal{Y}$
  - Classifier, $h : \mathcal{X} \rightarrow \mathcal{Y}$
  - Goal: find a classifier, $h$, that best approximates $c^*$
- Probabilistic Classification:
  - (Unknown) Target *distribution*, $y \sim P^*(Y|\boldsymbol{x})$
  - Distribution, $P(Y|\boldsymbol{x})$
  - Goal: find a distribution, $P$, that best approximates $P^*$

# Building a Probabilistic Classifier

- Define a decision rule
  - Given a test data point $\boldsymbol{x}'$, predict its label $\hat{y}$ using the *posterior distribution* $P(Y = y|X = \boldsymbol{x}')$
  - Common choice: $\hat{y} = \underset{y}{\text{argmax}}\, P(Y = y|X = \boldsymbol{x}')$
- Model the posterior distribution
  - Option 1 ("Discriminative Model") (today)
    - Model $P(Y|X)$ directly as some function of $X$
  - Option 2 ("Generative Model")
    - Use Bayes' rule:

$$P(Y|X) = \frac{P(X|Y)\ P(Y)}{P(X)} \propto P(X|Y)\ P(Y)$$

# Modelling the Posterior

- Suppose we have binary labels $y \in \{0,1\}$ and $D$-dimensional inputs $\boldsymbol{x} = [1, x_1, \ldots, x_D]^T \in \mathbb{R}^{D+1}$
- **Assume**

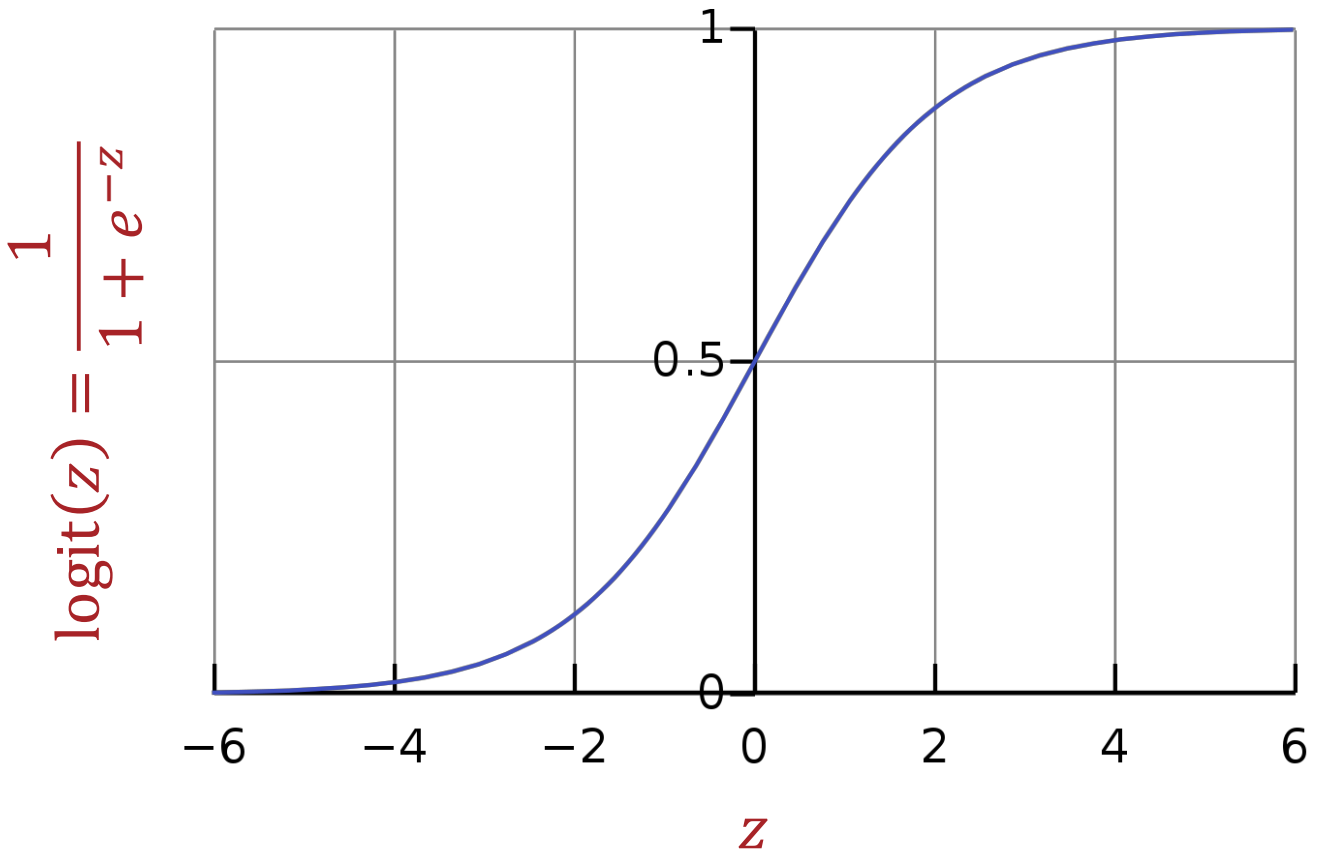
$$P(Y = 1|\boldsymbol{x}) = \text{logit}(\boldsymbol{w}^T\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^T\boldsymbol{x})}$$

$$= \frac{\exp(\boldsymbol{w}^T\boldsymbol{x})}{\exp(\boldsymbol{w}^T\boldsymbol{x}) + 1}$$

- This implies two useful facts:

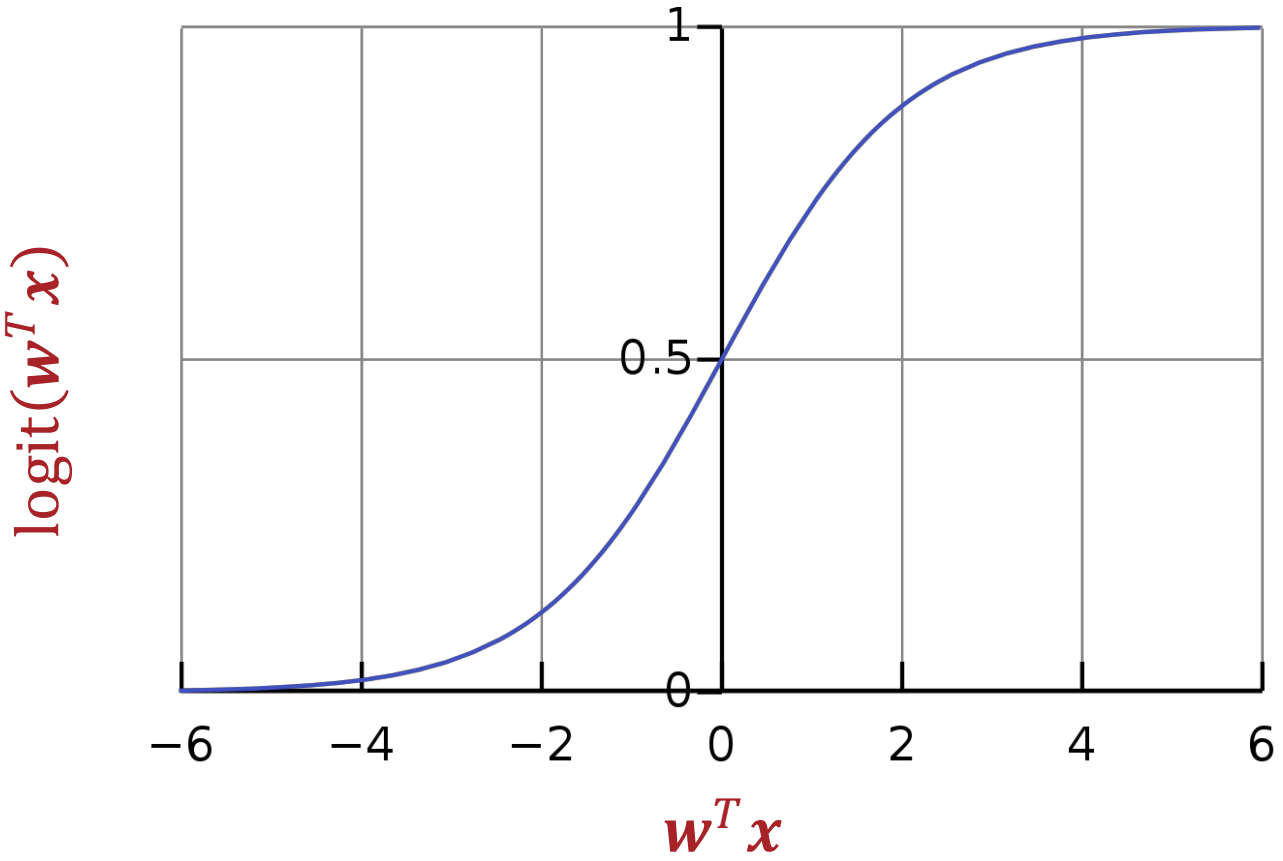
1. $P(Y = 0|\boldsymbol{x}) = 1 - P(Y = 1|\boldsymbol{x}) = \frac{1}{\exp(\boldsymbol{w}^T\boldsymbol{x}) + 1}$
2. $\frac{P(Y = 1|\boldsymbol{x})}{P(Y = 0|\boldsymbol{x})} = \exp(\boldsymbol{w}^T\boldsymbol{x}) \rightarrow \log\frac{P(Y = 1|\boldsymbol{x})}{P(Y = 0|\boldsymbol{x})} = \boldsymbol{w}^T\boldsymbol{x}$

# Logistic Function




Source: https://en.wikipedia.org/wiki/Logistic_function#/media/File:Logistic-curve.svg

# Why use the Logistic Function?




- Differentiable everywhere
- $\text{logit}: \mathbb{R} \rightarrow [0, 1]$
- The decision boundary is linear in $\boldsymbol{x}$!

Source: https://en.wikipedia.org/wiki/Logistic_function#/media/File:Logistic-curve.svg

Mythbuster

# Mythbusters

**MYTH**

Because the sigmoid is nonlinear, logistic regression has a nonlinear decision boundary.

# Mythbuster

## Mythbusters

**MYTH**

Because the sigmoid is nonlinear, logistic regression has a nonlinear decision boundary.

**BUSTED**

With threshold 1/2, the boundary is $w^T x = 0$ — linear in x. (The sigmoid only squashes scores into probabilities.)

# Logistic Regression Decision Boundary

$$\hat{y} = \begin{cases} 1 \text{ if } P(Y = 1|\boldsymbol{x}) \geq \frac{1}{2} \\ 0 \text{ otherwise} \end{cases}$$

$$P(Y = 1|\boldsymbol{x}) = \text{logit}(\boldsymbol{w}^T\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^T\boldsymbol{x})} \geq \frac{1}{2}$$

$$2 \geq 1 + \exp(-\boldsymbol{w}^T\boldsymbol{x})$$

$$1 \geq \exp(-\boldsymbol{w}^T\boldsymbol{x})$$

$$\log(1) \geq -\boldsymbol{w}^T\boldsymbol{x}$$

$$0 \leq \boldsymbol{w}^T\boldsymbol{x}$$

# Logistic Regression Decision Boundary

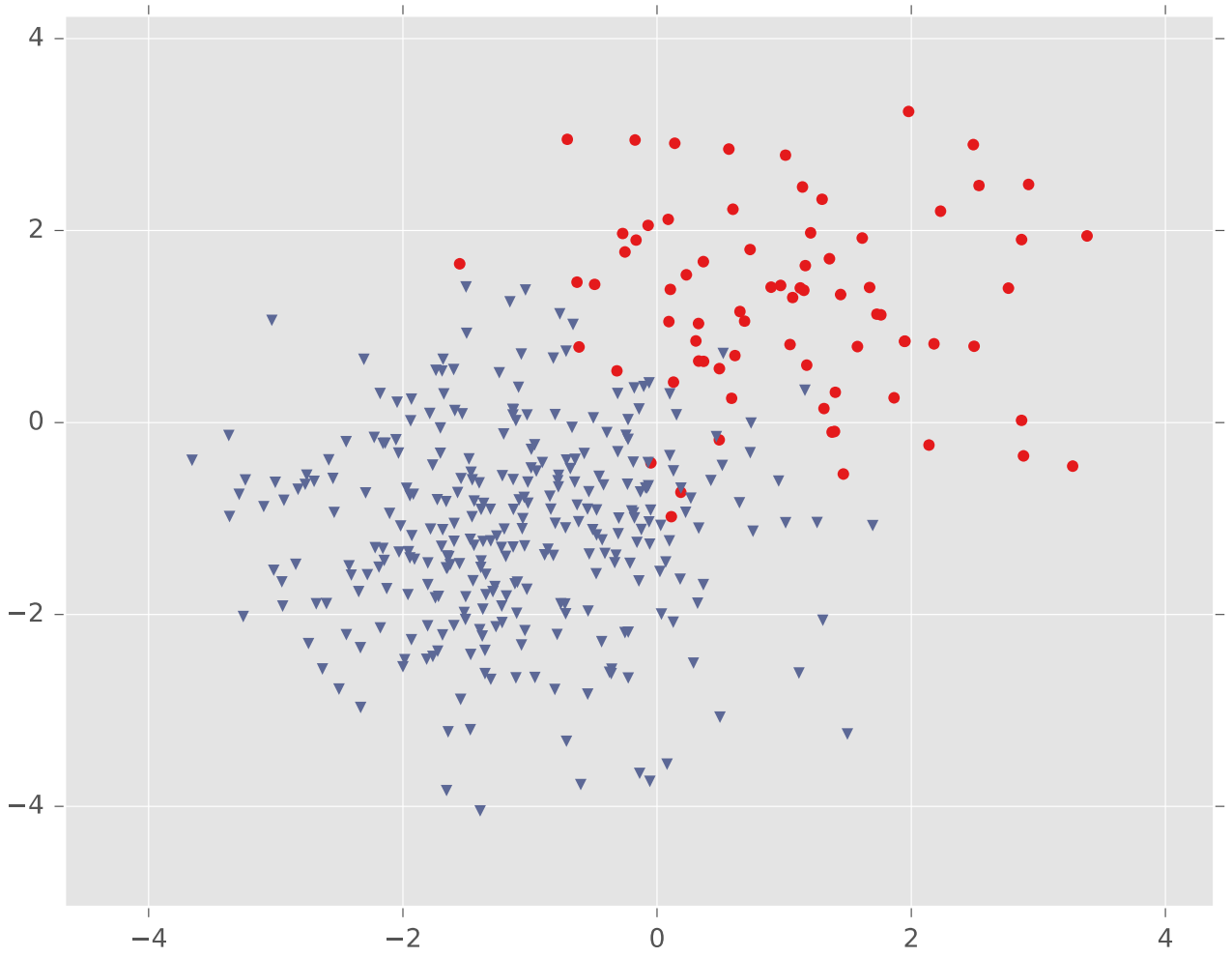



Figure courtesy of Matt Gormley

# Logistic Regression Decision Boundary

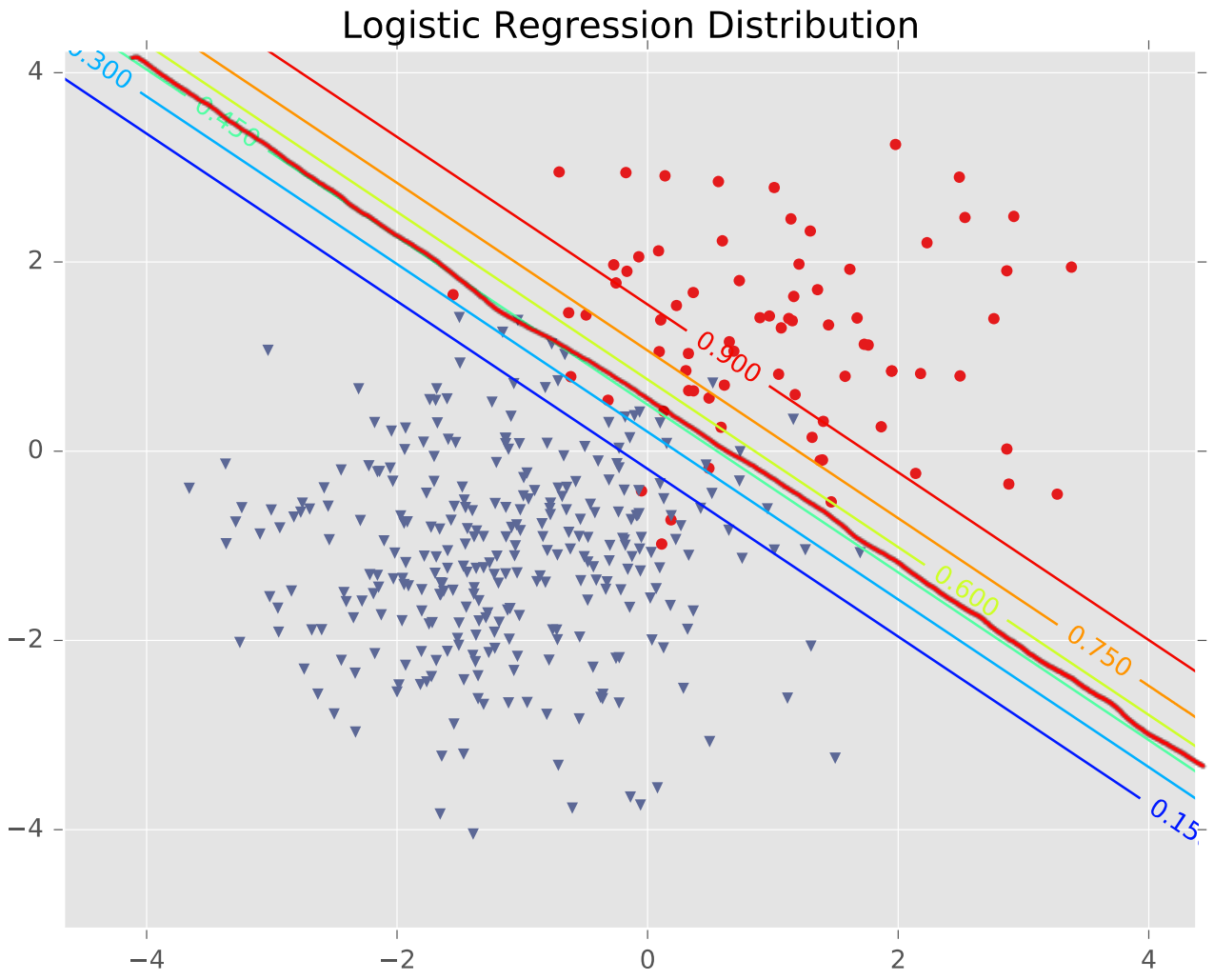



Figure courtesy of Matt Gormley

# Logistic Regression Decision Boundary

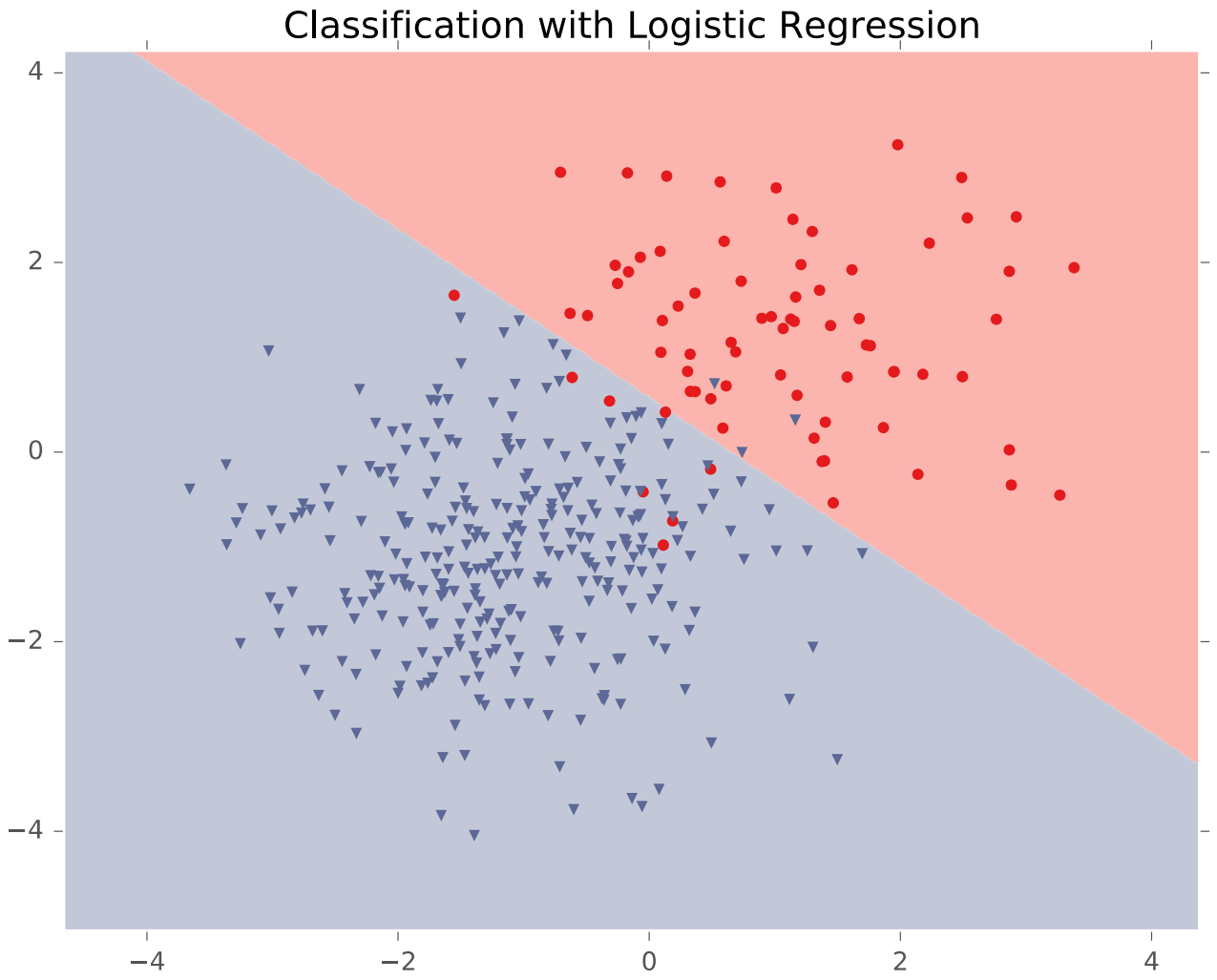



Figure courtesy of Matt Gormley

Mythbuster

# Mythbusters

**MYTH**

"Logistic regression" predicts a real-valued target like linear regression.

# Mythbuster

## Mythbusters

**MYTH**

"Logistic regression" predicts a real-valued target like linear regression.

**BUSTED**

It models $P(Y = 1 \mid X)$ using a logistic (sigmoid) of $w^T x$, producing a probability in [0,1]; we classify by thresholding.

# Logistic Regression

Not really regression

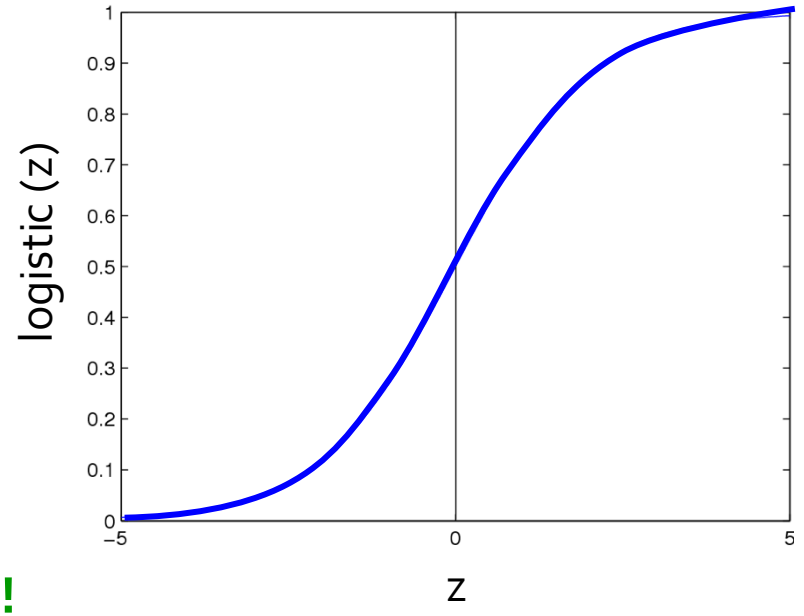
Assumes the following functional form for P(Y|X):

$$P(Y=0|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

Logistic function applied to a linear function of the data

**Logistic function (or Sigmoid):**

$$\frac{1}{1+exp(-z)}$$




**Features can be discrete or continuous!**

# Logistic Regression is a Linear Classifier!

Assumes the following functional form for P(Y|X):

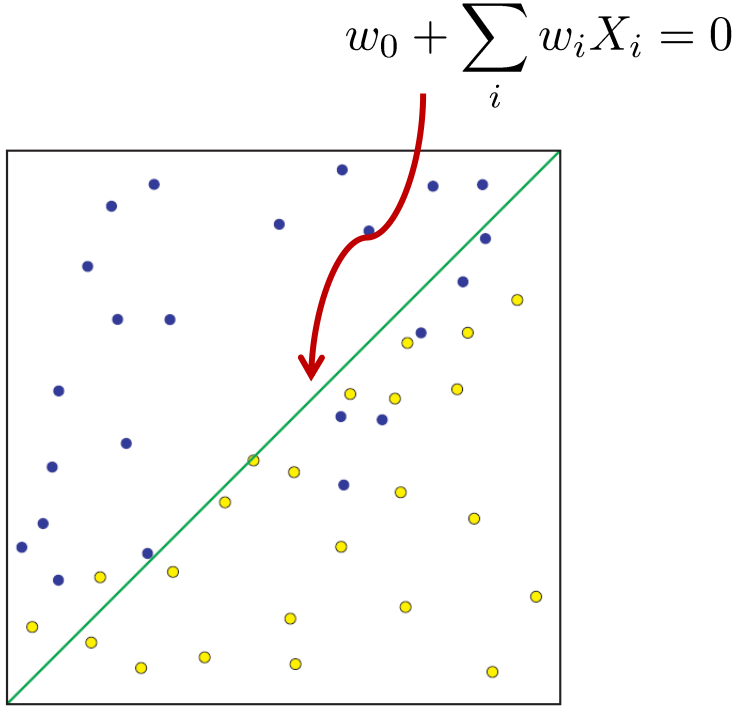
$$P(Y=0|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

Decision boundary: Note - Labels are 0,1

$$P(Y=0|X) \underset{1}{\overset{0}{\gtrless}} P(Y=1|X)$$

$$w_0 + \sum_i w_i X_i \underset{0}{\overset{1}{\gtrless}} 0$$

(Linear Decision Boundary)



# General Recipe for Machine Learning

- Define a model and model parameters
- Write down an objective function
- Optimize the objective w.r.t. the model parameters

# Recipe for Logistic Regression

- Define a model and model parameters
  - Assume independent, identically distributed (iid) data
  - Assume $P(Y = 1|X) = \text{logit}(\boldsymbol{w}^T \boldsymbol{x})$
  - Parameters: $\boldsymbol{w} = [w_0, w_1, \ldots, w_D]$
- Write down an objective function
  - ~~Maximize the *conditional* log-likelihood~~
  - Minimize the negative conditional log-likelihood
- Optimize the objective w.r.t. the model parameters
  - ???

# Setting the Parameters via Minimum Negative Conditional (log-)Likelihood Estimation (MCLE)

Find $\boldsymbol{w}$ that minimizes

$$\ell_{\mathcal{D}}(\boldsymbol{w}) = -\log P\left(y^{(1)}, \ldots, y^{(N)} \middle| \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}, \boldsymbol{w}\right) = -\log \prod_{n=1}^{N} P\left(y^{(n)} \middle| \boldsymbol{x}^{(n)}, \boldsymbol{w}\right)$$

$$= -\log \prod_{n=1}^{N} P\left(Y = 1 \middle| \boldsymbol{x}^{(n)}, \boldsymbol{w}\right)^{y^{(n)}} \left(P\left(Y = 0 \middle| \boldsymbol{x}^{(n)}, \boldsymbol{w}\right)\right)^{1-y^{(n)}}$$

$$= -\sum_{i=1}^{N} y^{(n)} \log P\left(Y = 1 \middle| \boldsymbol{x}^{(n)}, \boldsymbol{w}\right) + \left(1 - y^{(n)}\right) \log P\left(Y = 0 \middle| \boldsymbol{x}^{(n)}, \boldsymbol{w}\right)$$

$$= -\sum_{i=1}^{N} y^{(n)} \log \frac{P\left(Y = 1 \middle| \boldsymbol{x}^{(n)}, \boldsymbol{w}\right)}{P\left(Y = 0 \middle| \boldsymbol{x}^{(n)}, \boldsymbol{w}\right)} + \log P\left(Y = 0 \middle| \boldsymbol{x}^{(n)}, \boldsymbol{w}\right)$$

$$= -\sum_{i=1}^{N} y^{(n)} \boldsymbol{w}^T \boldsymbol{x}^{(n)} - \log\left(1 + \exp\left(\boldsymbol{w}^T \boldsymbol{x}^{(n)}\right)\right)$$

# Minimizing the Negative Conditional (log-)Likelihood

$$\ell_{\mathcal{D}}(\boldsymbol{w}) = -\sum_{n=1}^{N} y^{(n)} \boldsymbol{w}^T \boldsymbol{x}^{(n)} - \log\left(1 + \exp\left(\boldsymbol{w}^T \boldsymbol{x}^{(n)}\right)\right)$$

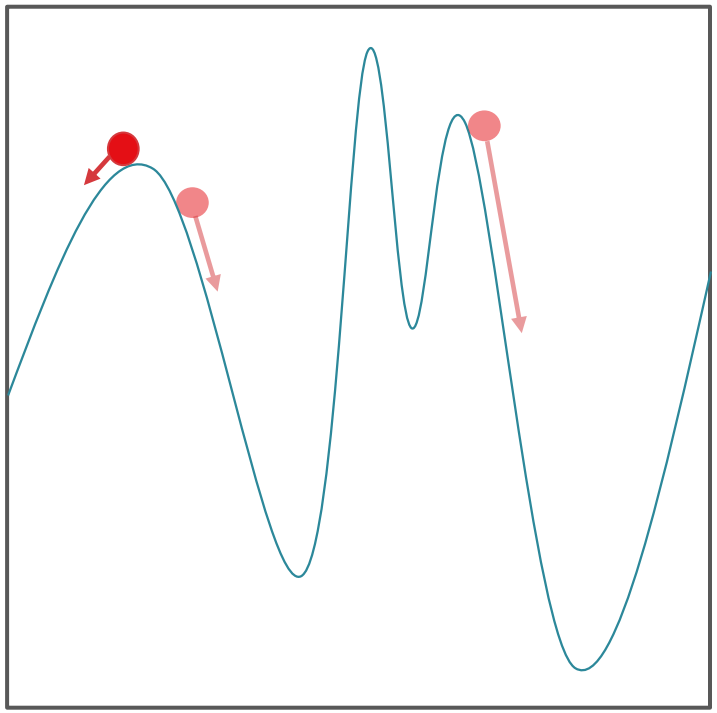
$$\nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}(\boldsymbol{w}) = -\sum_{n=1}^{N} y^{(n)} \nabla_{\boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{x}^{(n)} - \nabla_{\boldsymbol{w}} \log\left(1 + \exp\left(\boldsymbol{w}^T \boldsymbol{x}^{(n)}\right)\right)$$

$$= -\sum_{n=1}^{N} y^{(n)} \boldsymbol{x}^{(n)} - \frac{\exp\left(\boldsymbol{w}^T \boldsymbol{x}^{(n)}\right)}{1 + \exp\left(\boldsymbol{w}^T \boldsymbol{x}^{(n)}\right)} \boldsymbol{x}^{(n)}$$

$$= \sum_{n=1}^{N} \boldsymbol{x}^{(n)} \left(P\left(Y = 1 \middle| \boldsymbol{x}^{(n)}, \boldsymbol{w}\right) - y^{(n)}\right)$$

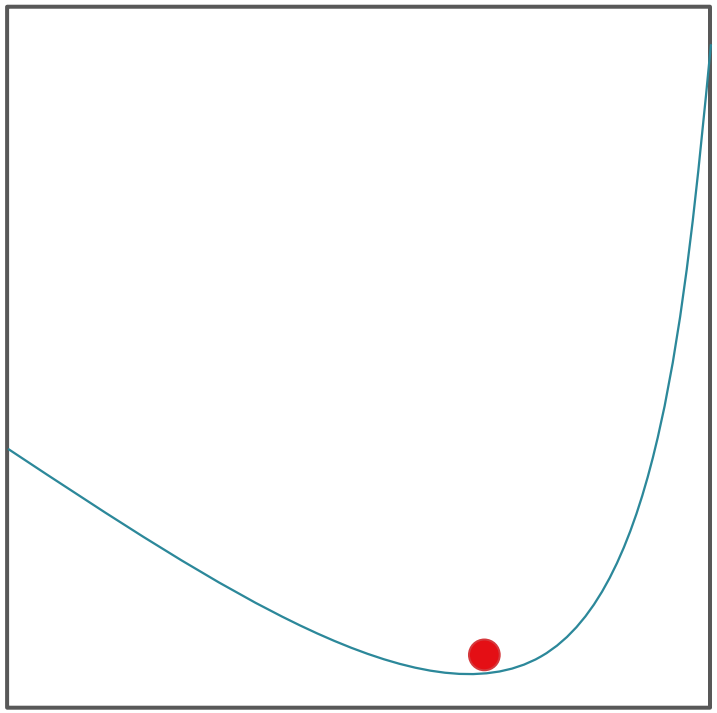
# Recall: Gradient Descent

- An iterative method for minimizing functions
- Requires the gradient to exist everywhere



# Recall: Gradient Descent

- An iterative method for minimizing functions
- Requires the gradient to exist everywhere



- Good news: the negative conditional log-likelihood, like the squared error, is also convex!

# Gradient Descent

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}, \eta^{(0)}$

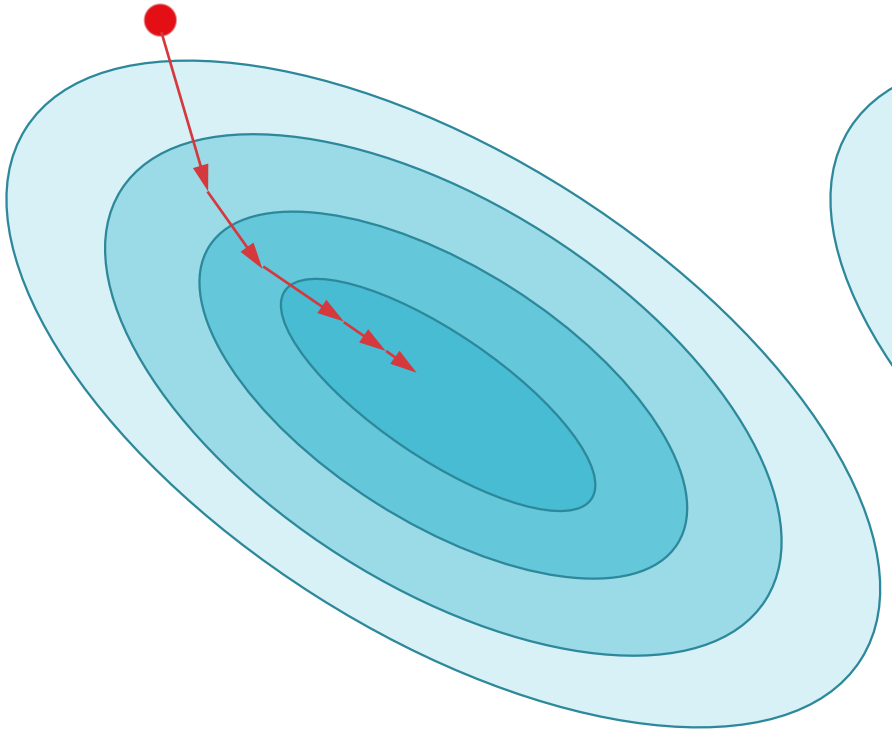
1. Initialize $\boldsymbol{w}^{(0)}$ to all zeros and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. Compute the gradient:

$$O(ND)\left\{\quad \nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right) = \sum_{n=1}^{N} \boldsymbol{x}^{(n)}\left(P\left(Y = 1 \middle| \boldsymbol{x}^{(n)}, \boldsymbol{w}^{(t)}\right) - y^{(n)}\right)\right.$$

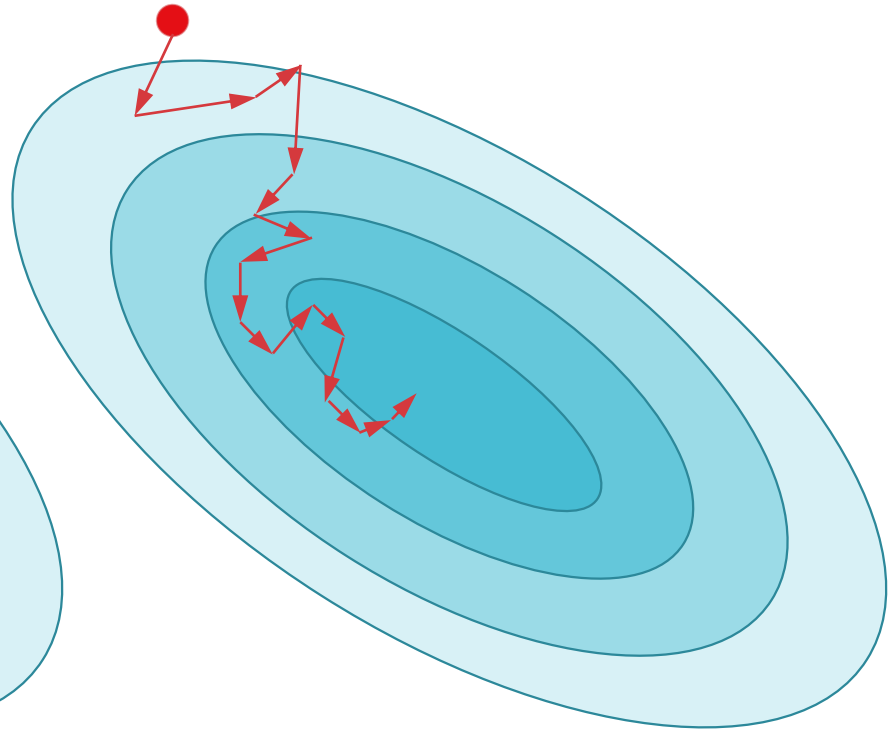
   b. Update $\boldsymbol{w}$: $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta^{(0)} \nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)$
   c. Increment $t$: $t \leftarrow t + 1$

- Output: $\boldsymbol{w}^{(t)}$

# Stochastic Gradient Descent vs. Gradient Descent



Gradient Descent



Stochastic Gradient Descent

# Stochastic Gradient Descent

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}, \eta_{SGD}^{(0)}$

1. Initialize $\boldsymbol{w}^{(0)}$ to all zeros and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. Randomly sample a data point from $\mathcal{D}$, $\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)$
   b. Compute the pointwise gradient:

$$\nabla_{\boldsymbol{w}} \ell^{(n)}\left(\boldsymbol{w}^{(t)}\right) = \boldsymbol{x}^{(n)}\left(P\left(Y = 1 \middle| \boldsymbol{x}^{(n)}, \boldsymbol{w}^{(t)}\right) - y^{(n)}\right)$$

   c. Update $\boldsymbol{w}$: $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta_{SGD}^{(0)} \nabla_{\boldsymbol{w}} \ell^{(n)}\left(\boldsymbol{w}^{(t)}\right)$
   d. Increment $t$: $t \leftarrow t + 1$

- Output: $\boldsymbol{w}^{(t)}$

# Stochastic Gradient Descent

- If the data point is sampled uniformly at random, then the expected value of the pointwise gradient is proportional to the full gradient:

$$E\left[\nabla_{\boldsymbol{w}}\ell_{\boldsymbol{x}^{(n)},y^{(n)}}\left(\boldsymbol{w}^{(t)}\right)\right] = \frac{1}{N}\sum_{n=1}^{N}\nabla_{\boldsymbol{w}}\ell^{(n)}\left(\boldsymbol{w}^{(t)}\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}^{(n)}\left(P\left(Y=1\middle|\boldsymbol{x}^{(n)},\boldsymbol{w}^{(t)}\right)-y^{(n)}\right)$$

$$= \frac{1}{N}\nabla_{\boldsymbol{w}}\ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)$$

- In practice, the data set is randomly shuffled then looped through so that each data point is used equally often

# Mini-batch Stochastic Gradient Descent

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}, \eta_{MB}^{(0)}, B$

1. Initialize $\boldsymbol{w}^{(0)}$ to all zeros and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. Randomly sample $B$ data points from $\mathcal{D}$:

$$\mathcal{D}_{batch}\left\{\left(\boldsymbol{x}^{(b)}, y^{(b)}\right)\right\}_{b=1}^{B}$$

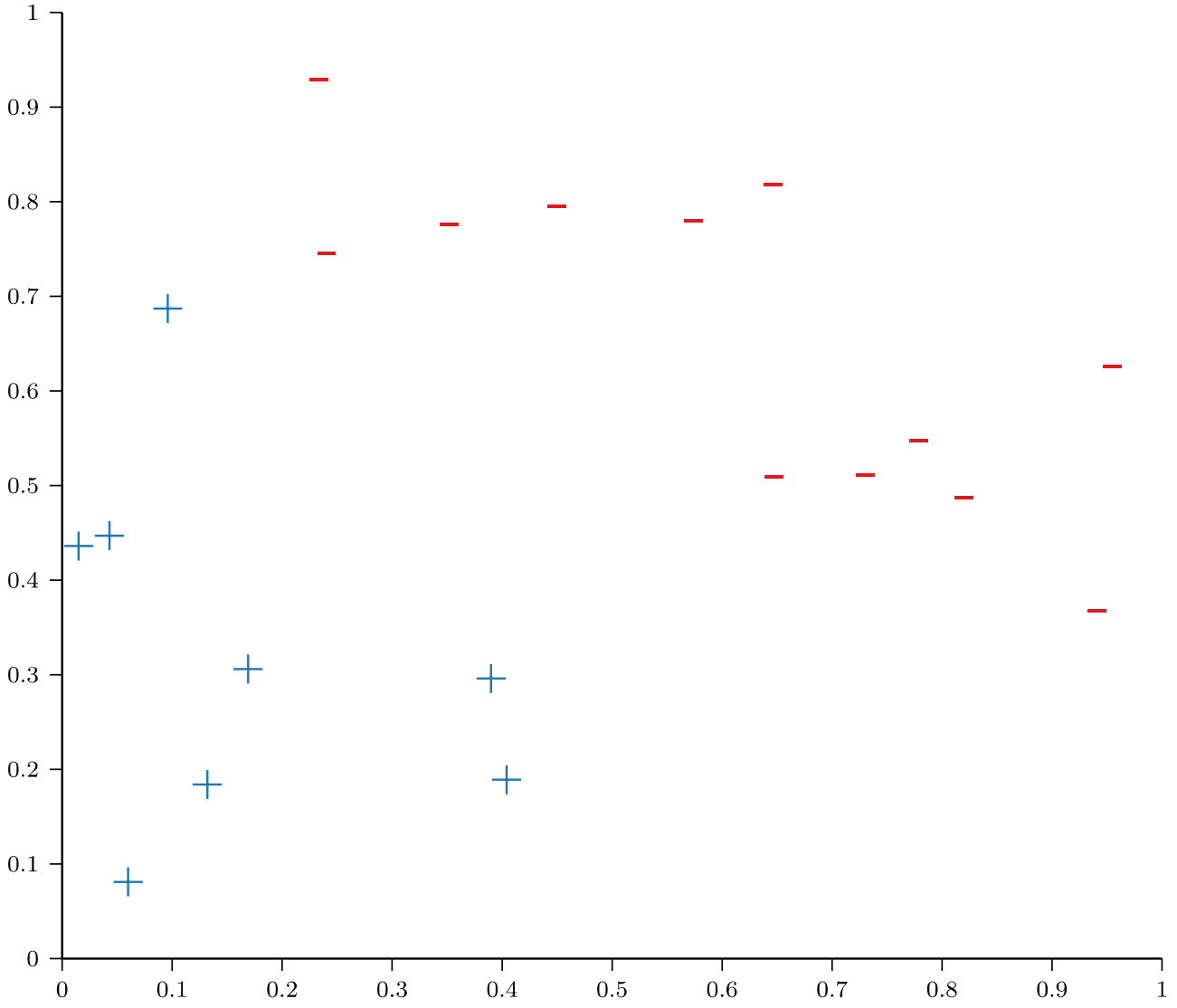
   b. Compute the gradient w.r.t. the sampled *batch*:

$$\nabla_{\boldsymbol{w}} \ell_{\mathcal{D}_{batch}}\left(\boldsymbol{w}^{(t)}\right) = \sum_{b=1}^{B} \boldsymbol{x}^{(b)}\left(P\left(Y = 1 \middle| \boldsymbol{x}^{(b)}, \boldsymbol{w}\right) - y^{(b)}\right)$$

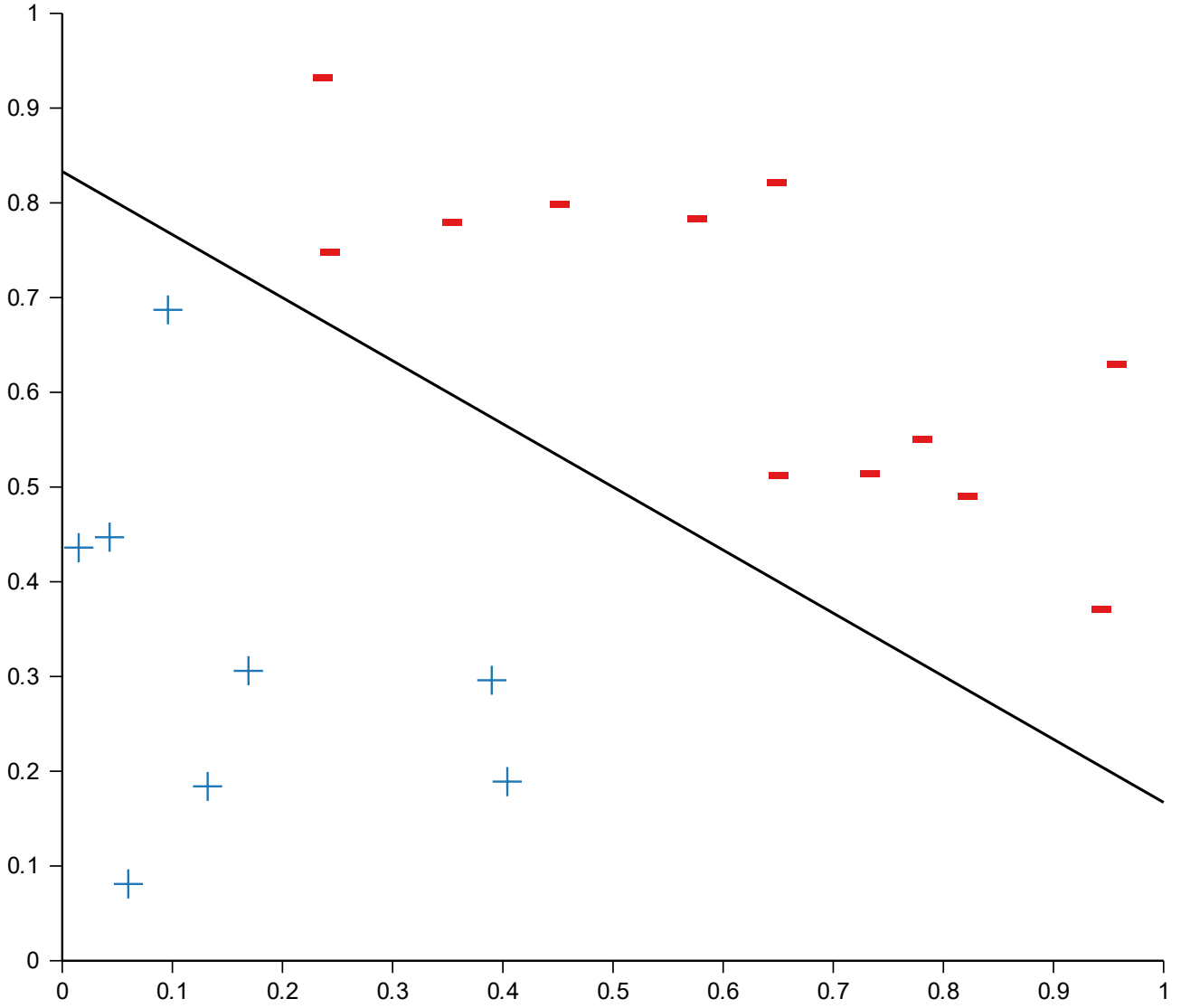
   c. Update $\boldsymbol{w}$: $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta_{MB}^{(0)} \nabla_{\boldsymbol{w}} \ell_{\mathcal{D}_{batch}}\left(\boldsymbol{w}^{(t)}\right)$
   d. Increment $t$: $t \leftarrow t + 1$

- Output: $\boldsymbol{w}^{(t)}$

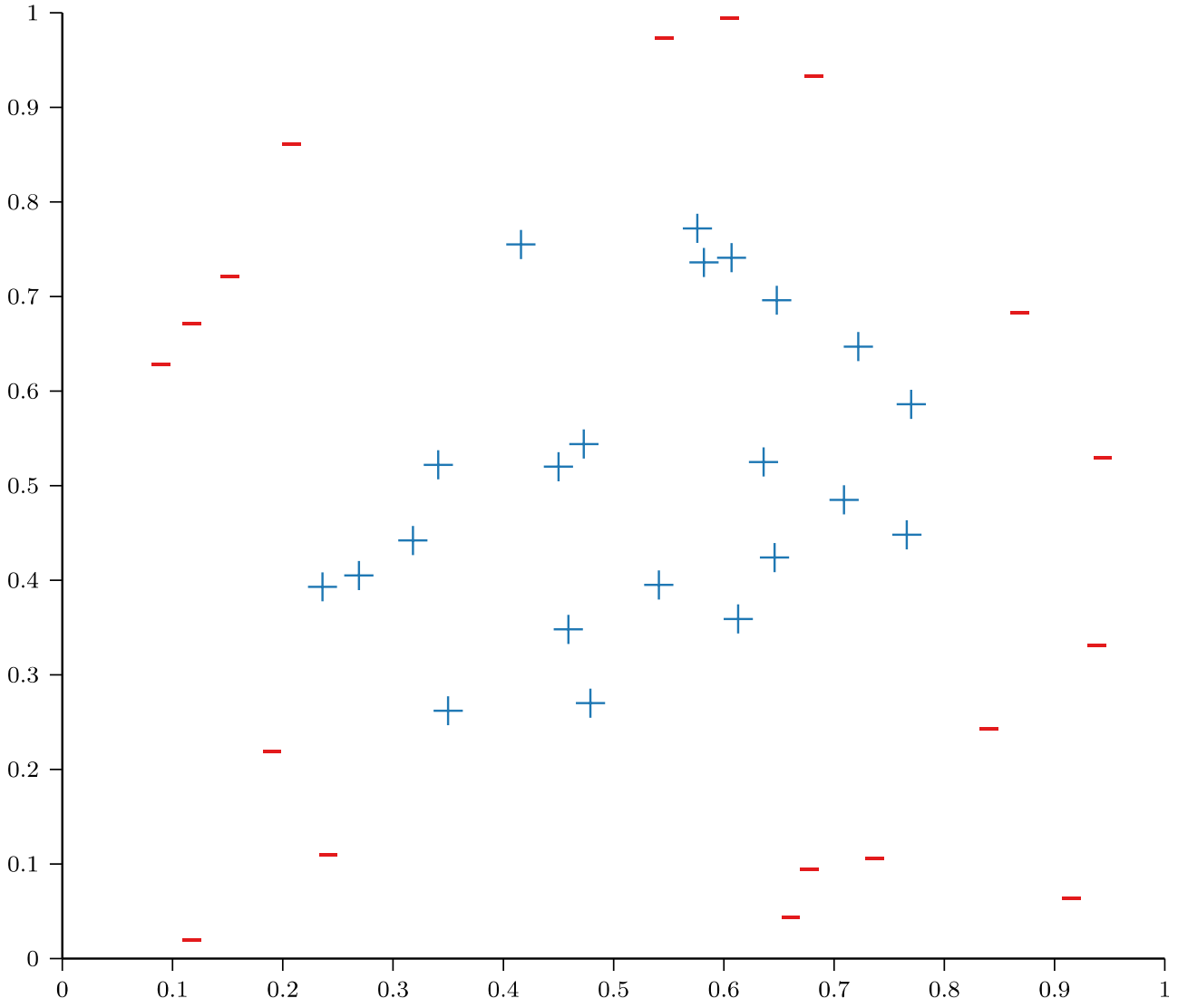
# Linear Models



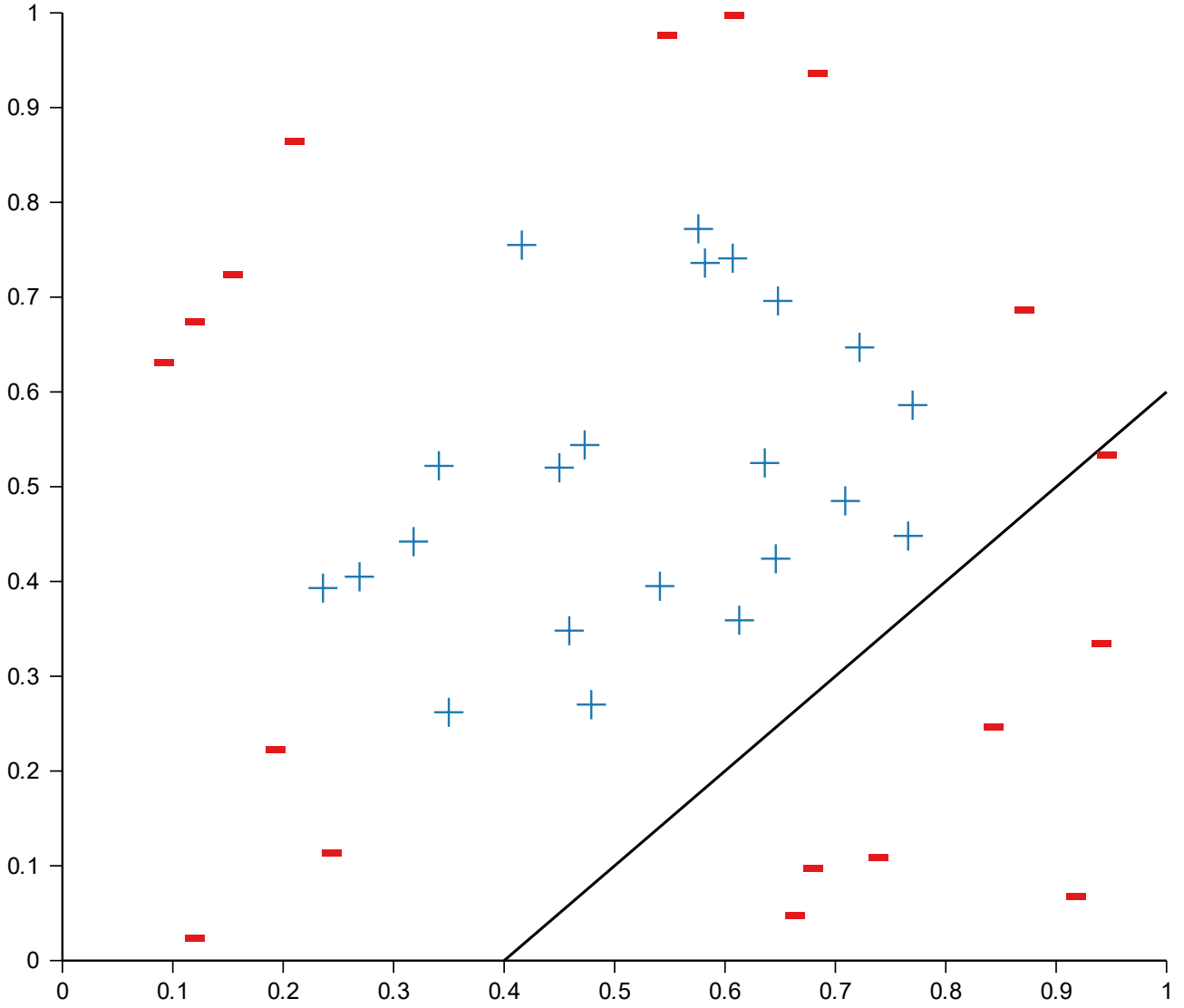
# Linear Models



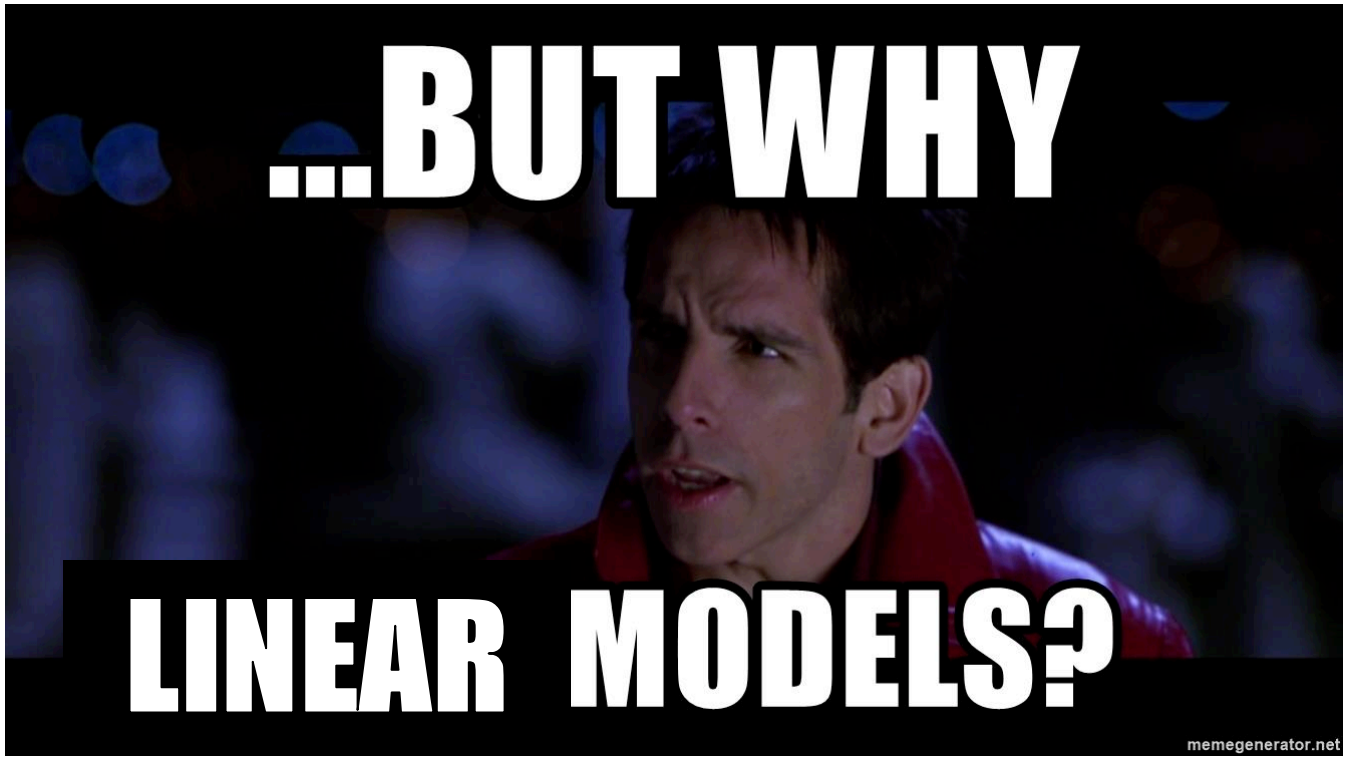
# Linear Models?



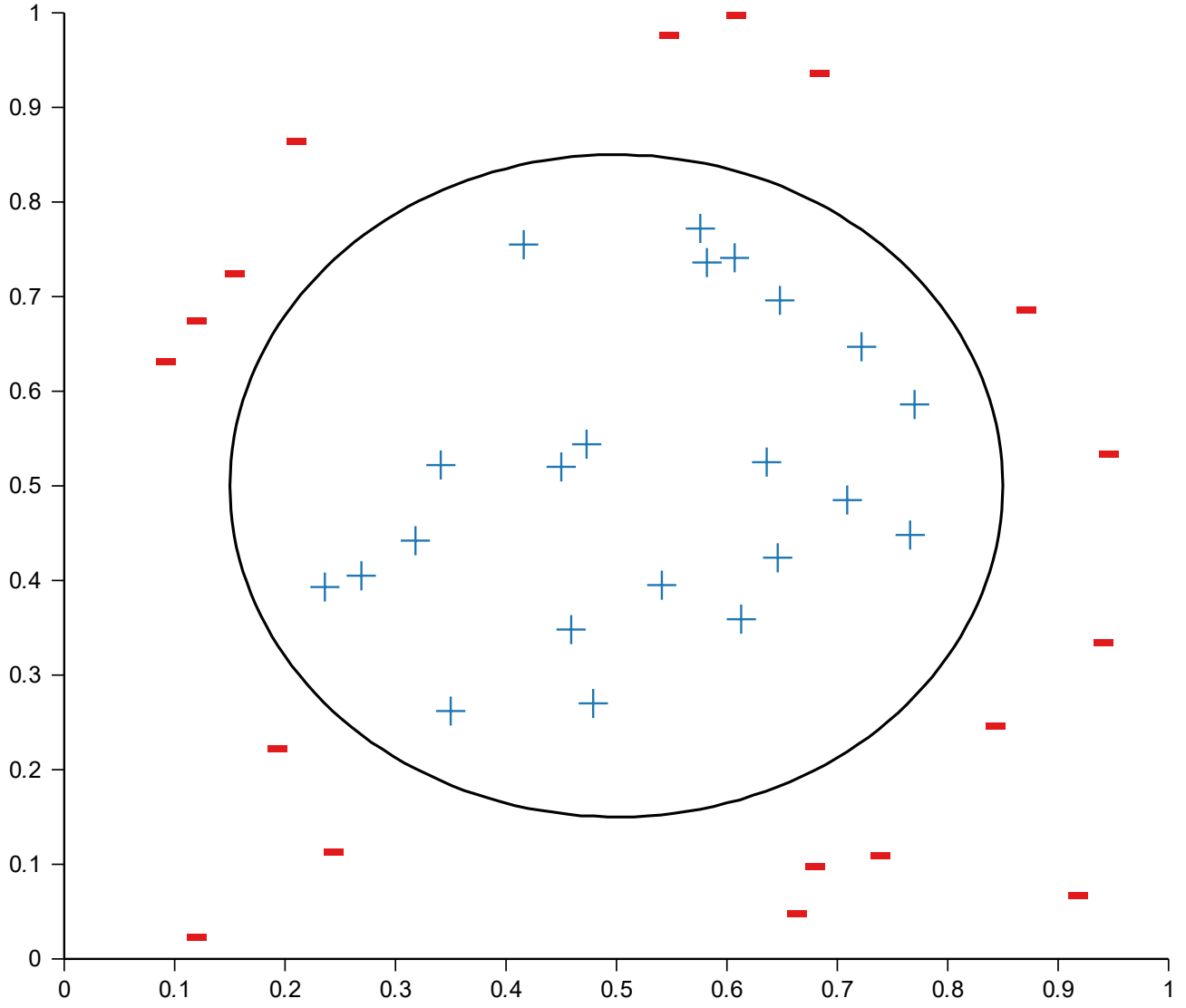
# Linear Models?



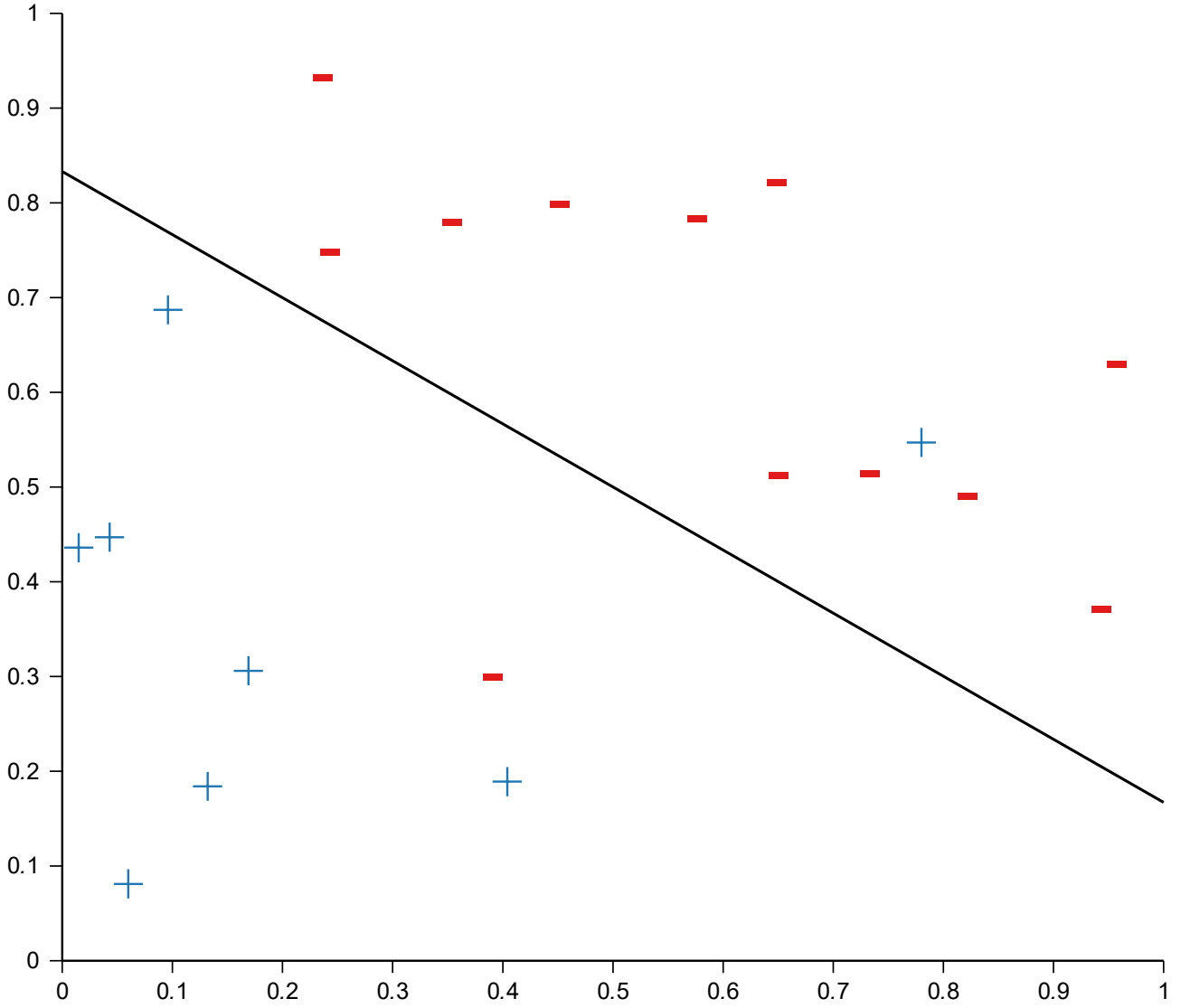
# Linear Models?



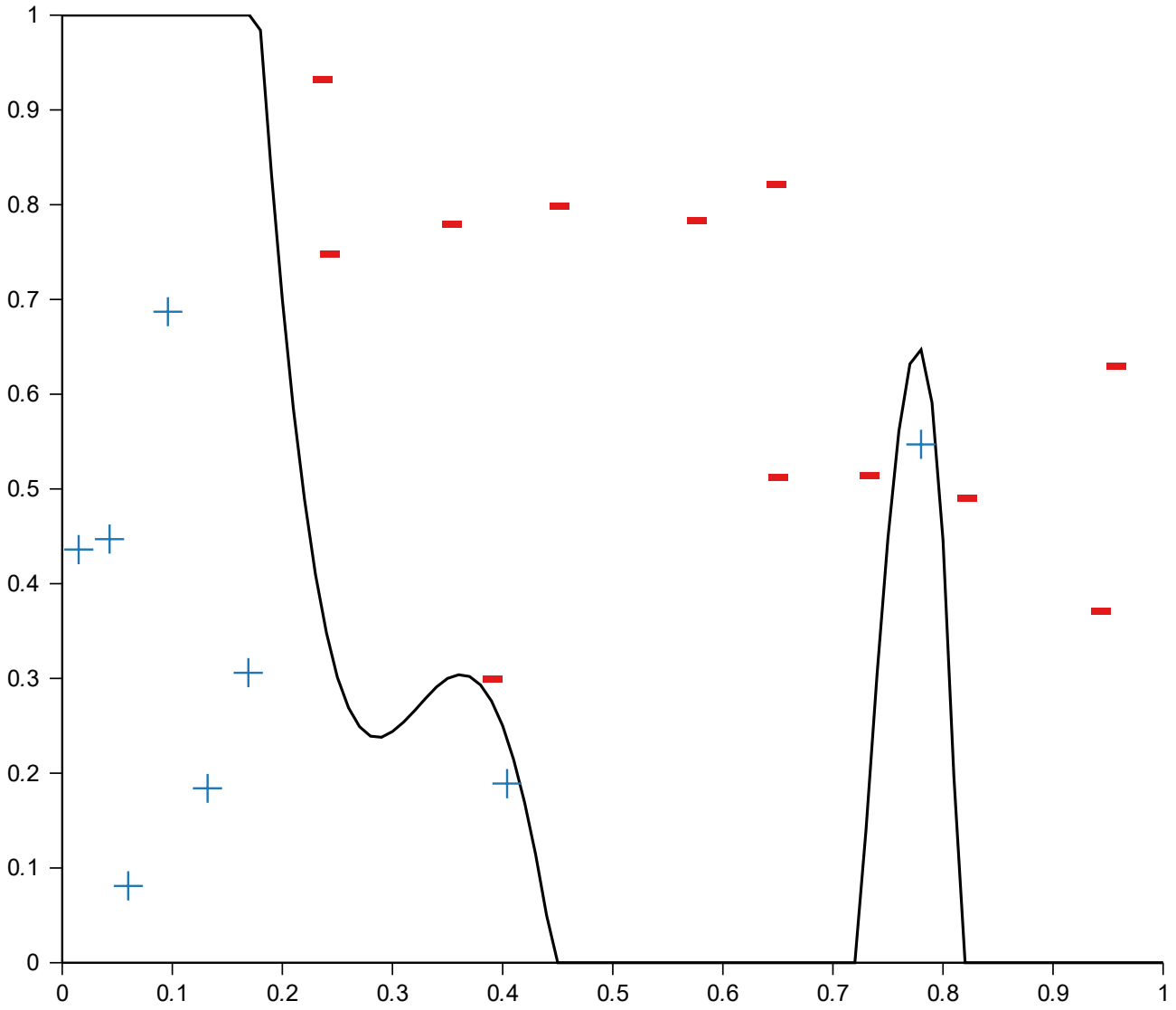
# Nonlinear Models



# Linear Models



# Nonlinear Models?



# Feature Transforms

- Given $D$-dimensional inputs $\boldsymbol{x} = [x_1, \dots, x_D]$, first compute some transformation of our input, e.g.,

$$\phi([x_1, x_2]) = [z_1 = (x_1 - 0.5)^2, z_2 = (x_2 - 0.5)^2]$$

# General $Q^{th}$-order Transforms

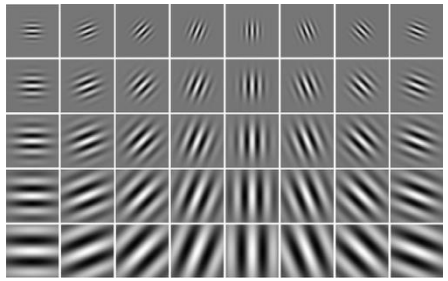
- $\phi_{2,2}([x_1, x_2]) = [x_1, x_2, x_1^2, x_1x_2, x_2^2]$
- $\phi_{2,3}([x_1, x_2]) = [x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3]$
- $\phi_{2,4}([x_1, x_2]) =$
  $[x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3, x_1^4, x_1^3x_2, x_1^2x_2^2, x_1x_2^3, x_2^4]$
- $\phi_{2,Q}$ maps a 2-D input to a $O(Q^2)$-D output
- Scales even worse for higher-dimensional inputs…

# Deterministic Transforms

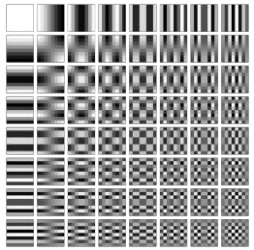
- log-transforms: x -> log(a + x)
- power/root transforms: x -> x^a
- polynomial transforms: x -> (x, x^2, x^3,…)

# Harmonic Transforms

- Gabor Filters:



- Discrete Cosine Transforms:



- …

# Can also have Linear Transforms

$$X \mapsto \{w_j^T X\}_{j=1}^k$$

- Principal Component Analysis
- Independent Component Analysis

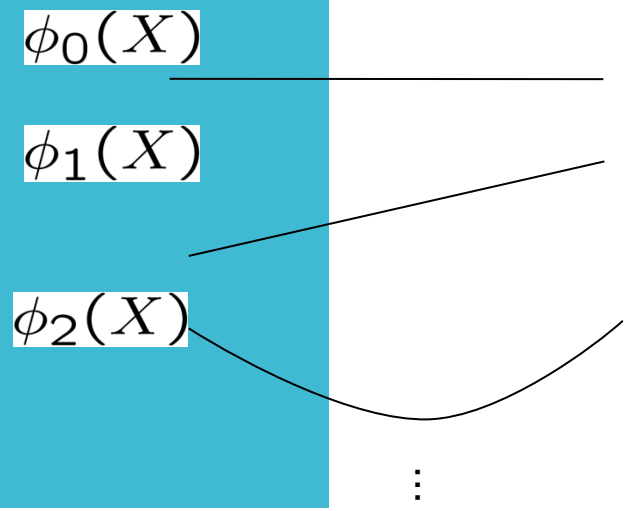
# Regression with basis functions

$$f(X) = \sum_{j=0}^{m} \beta_j \phi_j(X)$$

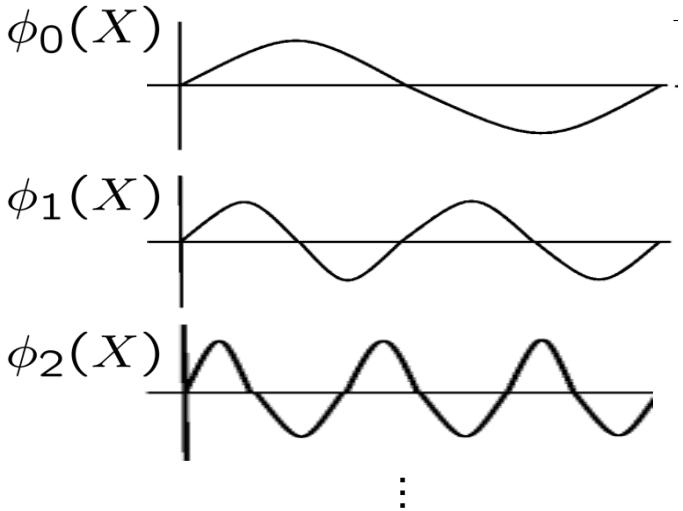
Basis coefficients

Basis functions (Linear combinations yield meaningful spaces)

Polynomial Basis

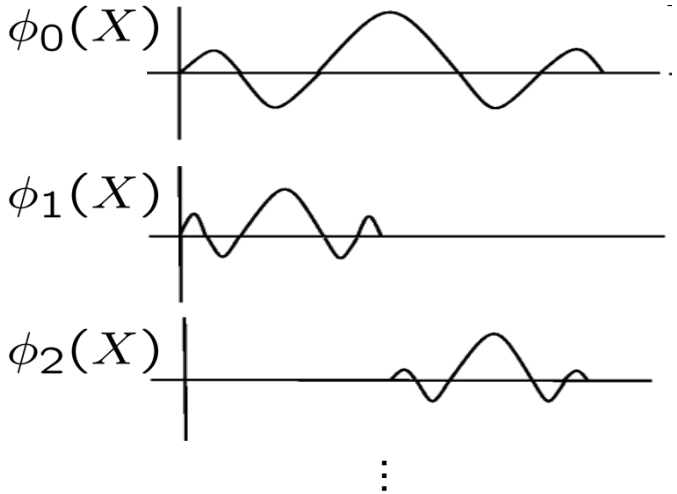



Fourier Basis




Good representation for periodic functions

Wavelet Basis



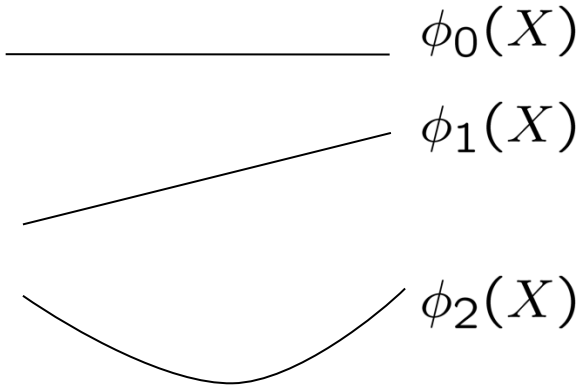

Good representation for local functions

# Regression with nonlinear features

$$f(X) = \sum_{j=0}^{m} \beta_j X^j = \sum_{j=0}^{m} \beta_j \phi_j(X)$$

Weight of each feature

Nonlinear features




In general, use any nonlinear features

e.g. $e^X$, log X, 1/X, sin(X), …

$$\widehat{\beta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}$$

or

$$(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

$$\mathbf{A} = \begin{bmatrix} \phi_0(X_1) & \phi_1(X_1) & \dots & \phi_m(X_1) \\ \vdots & & \ddots & \vdots \\ \phi_0(X_n) & \phi_1(X_n) & \dots & \phi_m(X_n) \end{bmatrix}$$

$$\widehat{f}_n(X) = \mathbf{X}\widehat{\beta}$$

$$\mathbf{X} = [\phi_0(X) \;\; \phi_1(X) \;\; \dots \;\; \phi_m(X)]$$

# Polynomial Regression

Univariate (1-dim) case:

degree m

$$f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_m X^m = \mathbf{X}\beta$$

where $\mathbf{X} = [1 \;\; X \;\; X^2 \ldots X^m] \quad \beta = [\beta_1 \ldots \beta_m]^T$

$$\widehat{\beta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y} \quad \text{or} \quad (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y} \qquad \widehat{f}_n(X) = \mathbf{X}\widehat{\beta}$$

where $\mathbf{A} = \begin{bmatrix} 1 & X_1 & X_1^2 & \ldots & X_1^m \\ \vdots & & & \ddots & \vdots \\ 1 & X_n & X_n^2 & \ldots & X_n^m \end{bmatrix}$

Multivariate (p-dim) case:

$$f(X) = \beta_0 + \beta_1 X^{(1)} + \beta_2 X^{(2)} + \cdots + \beta_p X^{(p)} + \sum_{i=1}^{p}\sum_{j=1}^{p} \beta_{ij} X^{(i)} X^{(j)} + \sum_{i=1}^{p}\sum_{j=1}^{p}\sum_{k=1}^{p} X^{(i)} X^{(j)} X^{(k)} + \ldots \text{terms up to degree m}$$

# Mythbusters

## Mythbusters

**MYTH**

If the data isn’t linearly separable, just keep increasing the polynomial degree.

# Mythbusters

## Mythbusters

**MYTH**

If the data isn’t linearly separable, just keep increasing the polynomial degree.

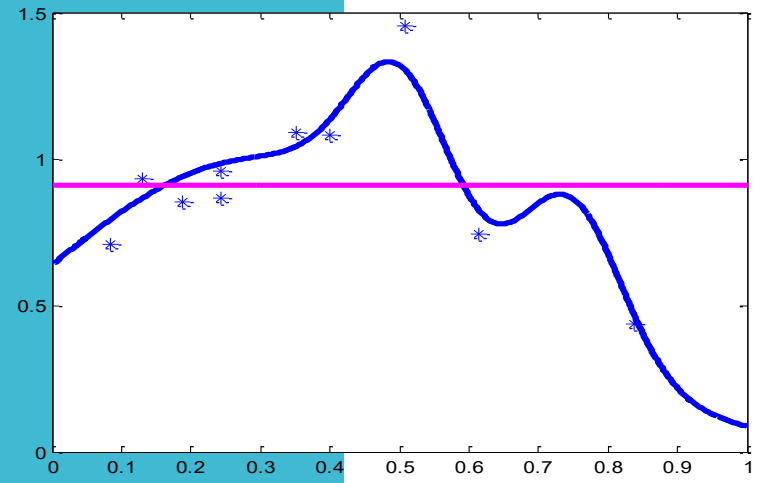
**BUSTED**

Higher-order feature transforms can overfit (bias–variance tradeoff) and can be very expensive to compute (e.g., $\Phi^{10}(x)$ costs $O(D^{10})$).

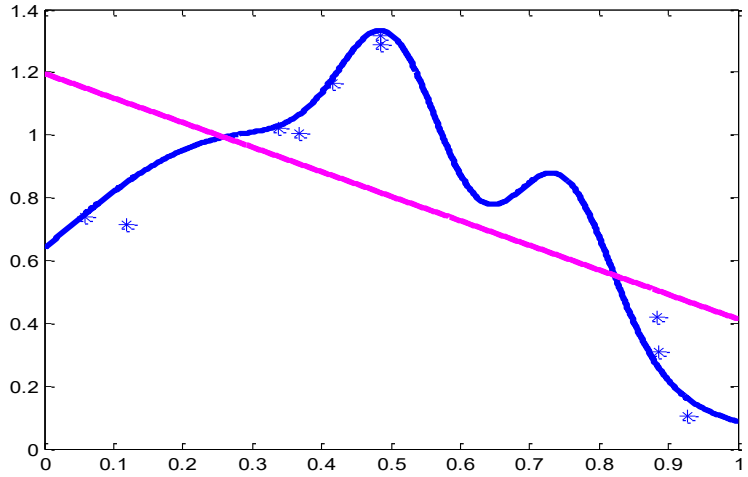
# Polynomial Regression

Polynomial of order k, equivalently of degree up to k-1

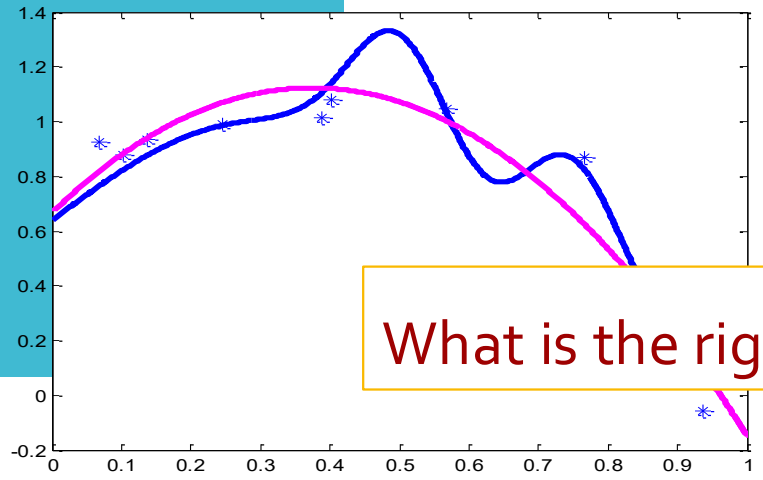
k=1



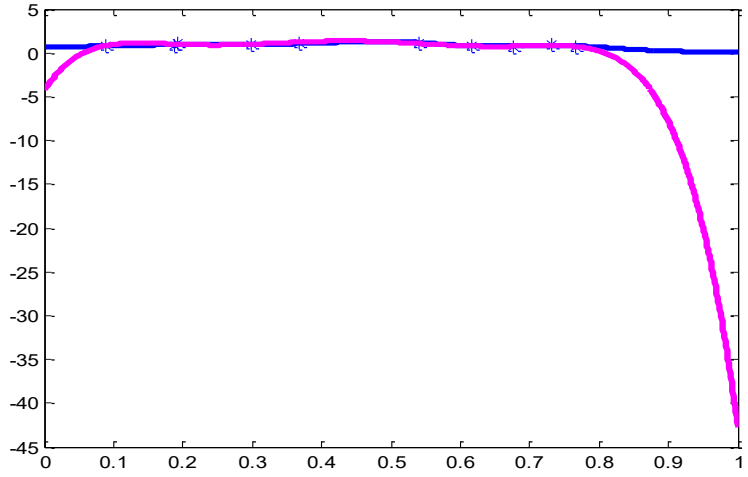

k=2




k=3

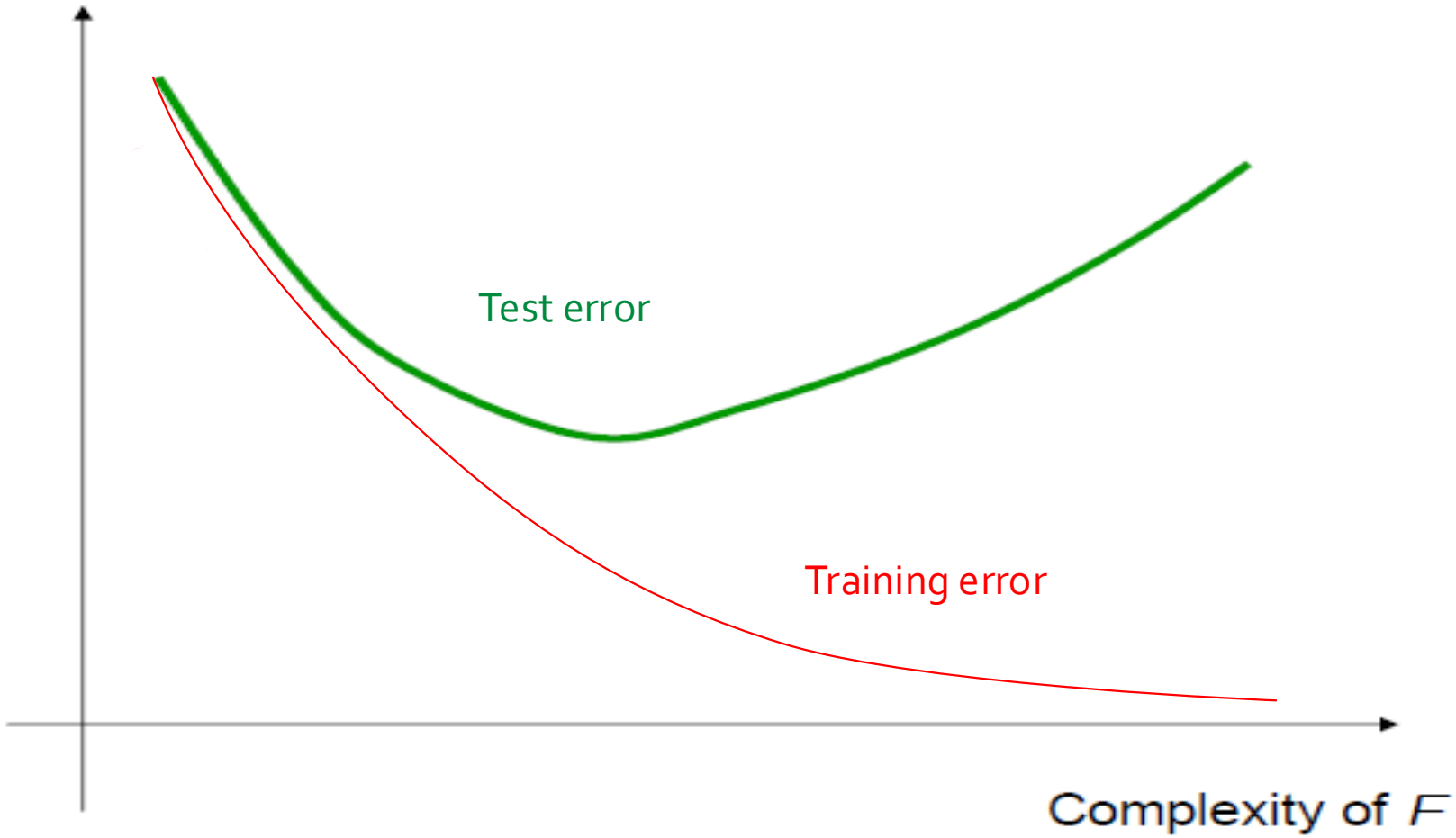



k=7




What is the right order?

# Effect of Model Complexity



# Efficiency

- Depending on the transformation $\Phi$ and the dimensionality of the original input space $\mathcal{X}$, computing $\Phi(\boldsymbol{x})$ can be prohibitively computationally expensive
  - Computing $\Phi_2(\boldsymbol{x}) = [x_1, x_2, \ldots, x_D, x_1^2, x_1 x_2, \ldots, x_D^2]$ for $x \in \mathbb{R}^D$ requires $D + \binom{D}{2} + D = \frac{D^2+3D}{2} = O(D^2)$ time
  - Computing $\Phi_{10}(\boldsymbol{x})$ requires $O(D^{10})$ time
- Tradeoff:
  - High-dimensional transformations can result in good hypotheses (as long as they don't overfit) but…
  - High-dimensional transformations are expensive