10-701: Introduction to Machine Learning

Lecture 7 –Logistic Regression

Hoda Heidari

* Slides adopted from F24 offering of 10701 by Henry Chai.

Recall: Probabilistic Learning

- Previously:
  - (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^*
- Now:
  - (Unknown) Target *distribution*, $y \sim P^*(Y|\boldsymbol{x})$
  - Distribution, $P(Y|\boldsymbol{x})$
  - Goal: find a distribution, P, that best approximates P^*

Recipe for Naïve Bayes

- Define a model space and model parameters
  - Make the Naïve Bayes assumption

$$P(X|Y) = \prod_{d=1}^{D} P(X_d|Y)$$

  - Parameters: $\pi = P(Y = 1)$, $\theta_{d,y} = P(X_d = 1|Y = y)$
- Write down an objective function
  - Maximize the log-likelihood
- Optimize the objective w.r.t. the model parameters
  - Solve in *closed form*: take partial derivatives, set to 0 and solve

Bernoulli Naïve Bayes

- Binary label
  - $Y \sim \text{Bernoulli}(\pi)$
  - $\hat{\pi} = {N_{Y=1}}/{N}$
    - N = # of data points
    - $N_{Y=1}$ = # of data points with label 1
- Binary features
  - $X_d | Y = y \sim \text{Bernoulli}\left(\theta_{d,y}\right)$
  - $\hat{\theta}_{d,y} = {N_{Y=y,\, X_d=1}}/{N_{Y=y}}$
    - $N_{Y=y}$ = # of data points with label y
    - $N_{Y=y,\, X_d=1}$ = # of data points with label y and feature $X_d = 1$

Bernoulli Naïve Bayes: Making Predictions

- Given a test data point $\boldsymbol{x}' = [x_1', \dots, x_D']^T$

Bernoulli Naïve Bayes: Making Predictions

- Given a test data point $\boldsymbol{x}' = [x'_1, \dots, x'_D]^T$

$$P(Y = 1|\boldsymbol{x}') \propto P(Y = 1)P(\boldsymbol{x}'|Y = 1)$$

$$= \hat{\pi} \prod_{d=1}^{D} \hat{\theta}_{d,1}^{x'_d} \left(1 - \hat{\theta}_{d,1}\right)^{1-x'_d}$$

$$P(Y = 0|\boldsymbol{x}') \propto (1 - \hat{\pi}) \prod_{d=1}^{D} \hat{\theta}_{d,0}^{x'_d} \left(1 - \hat{\theta}_{d,0}\right)^{1-x'_d}$$

$$\hat{y} = \begin{cases} 1 \text{ if } \hat{\pi} \prod_{d=1}^{D} \hat{\theta}_{d,1}^{x'_d} \left(1 - \hat{\theta}_{d,1}\right)^{1-x'_d} > \\ \quad (1 - \hat{\pi}) \prod_{d=1}^{D} \hat{\theta}_{d,0}^{x'_d} \left(1 - \hat{\theta}_{d,0}\right)^{1-x'_d} \\ 0 \text{ otherwise} \end{cases}$$

What if some Word-Label pair never appears in our training data?

x_1 ("hat")	x_2 ("cat")	x_3 ("dog")	x_4 ("fish")	x_5 ("mom")	x_6 ("dad")	y (Dr. Seuss)
1	1	0	0	0	0	1
0	0	1	0	0	0	0
0	0	0	1	0	0	1
0	0	0	0	1	0	0

The Cat in the Hat gets a Dog (by ???)

- If some $\hat{\theta}_{d,y} = 0$ and that word appears in our test data $\boldsymbol{x}'$, then $P(Y = y|\boldsymbol{x}') = 0$ even if all the other features in $\boldsymbol{x}'$ point to the label being y!
- The model has been overfit to the training data...
- We can address this with a prior over the parameters!

Setting the Parameters via MAP

- Binary label
  - $Y \sim \text{Bernoulli}(\pi)$
  - $\hat{\pi} = {}^{N_{Y=1}}/_{N}$
    - N = # of data points
    - $N_{Y=1}$ = # of data points with label 1
- Binary features
  - $X_d | Y = y \sim \text{Bernoulli}\left(\theta_{d,y}\right)$ and $\theta_{d,y} \sim \text{Beta}(\alpha, \beta)$
  - $\hat{\theta}_{d,y} = {}^{N_{Y=y,\, X_d=1}+(\alpha-1)}/_{N_{Y=y}+(\alpha-1)+(\beta-1)}$
    - $N_{Y=y}$ = # of data points with label y
    - $N_{Y=y,\, X_d=1}$ = # of data points with label y and feature $X_d = 1$
    - α and β are “pseudocounts” of imagined data points that help avoid zero-probability predictions.
    - Common choice: $\alpha = \beta = 2$

What can we do when this is a bad/incorrect assumption, e.g., when our features are words in a sentence?

- ***Assume*** features are conditionally independent given the label:

$$P(X|Y) = \prod_{d=1}^{D} P(X_d|Y)$$

- Pros:
  - Significantly reduces computational complexity
  - Also reduces model complexity, combats overfitting
- Cons:
  - Is a strong, often illogical assumption
    - We'll see a relaxed version of this much later when we discuss Bayesian networks

Key Takeaways

- Text data
  - Bag-of-words feature representation
- Naïve Bayes
  - Conditional independence assumption
    - Pros and cons
  - Different Naïve Bayes models based on type of features
  - MLE vs. MAP for Bernoulli Naïve Bayes

Recall: 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: $\hat{y} = \underset{y}{\operatorname{argmax}}\, P(Y = y|X = \boldsymbol{x}')$
- Model the posterior distribution
  - Option 1 - Model $P(Y|X)$ directly as some function of X (today!)
  - Option 2 - Use Bayes' rule (Monday):

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

Modelling the Posterior

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

- This implies two useful facts:

Modelling the Posterior

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

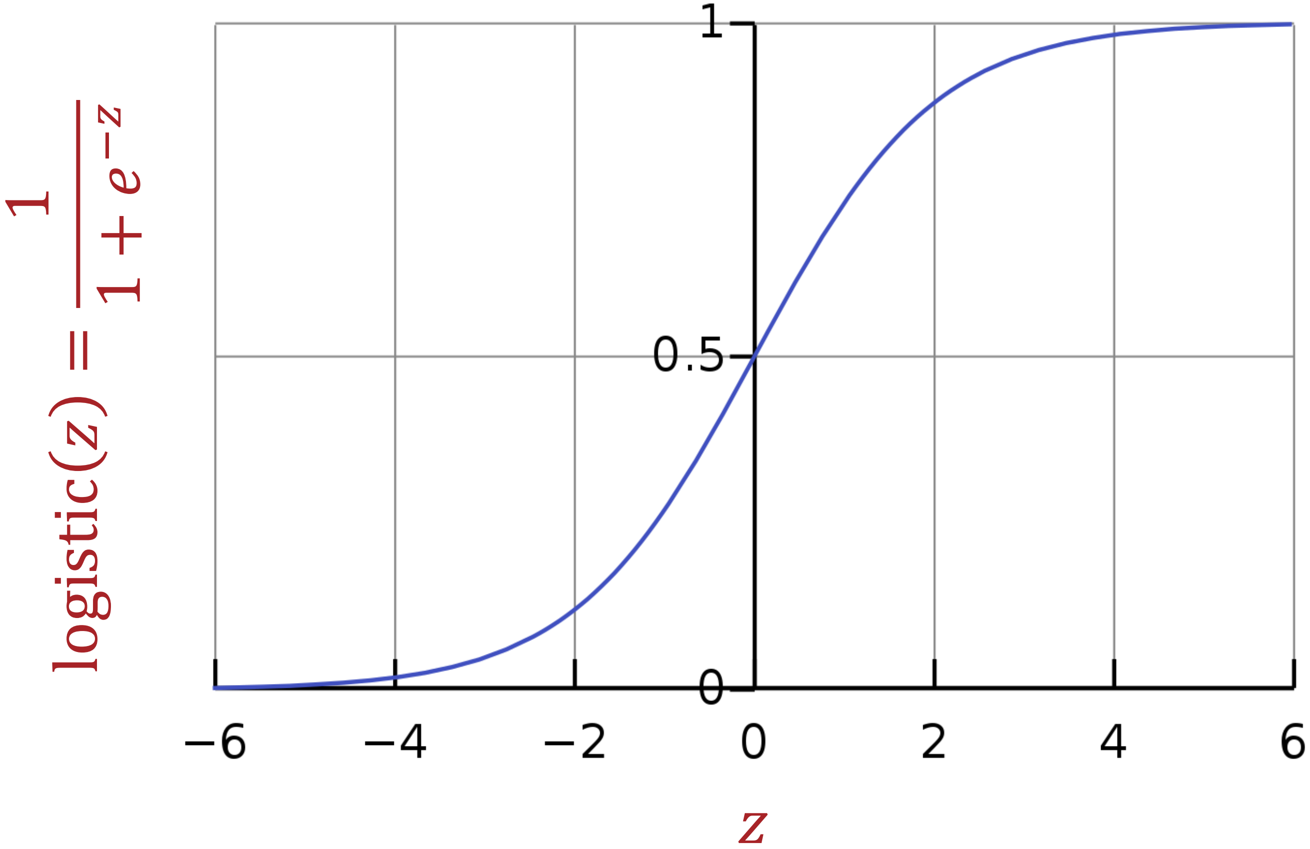
$$P(Y = 1|\boldsymbol{x}) = \text{logistic}(\boldsymbol{w}^T\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^T\boldsymbol{x})}$$

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

- This implies two useful facts:

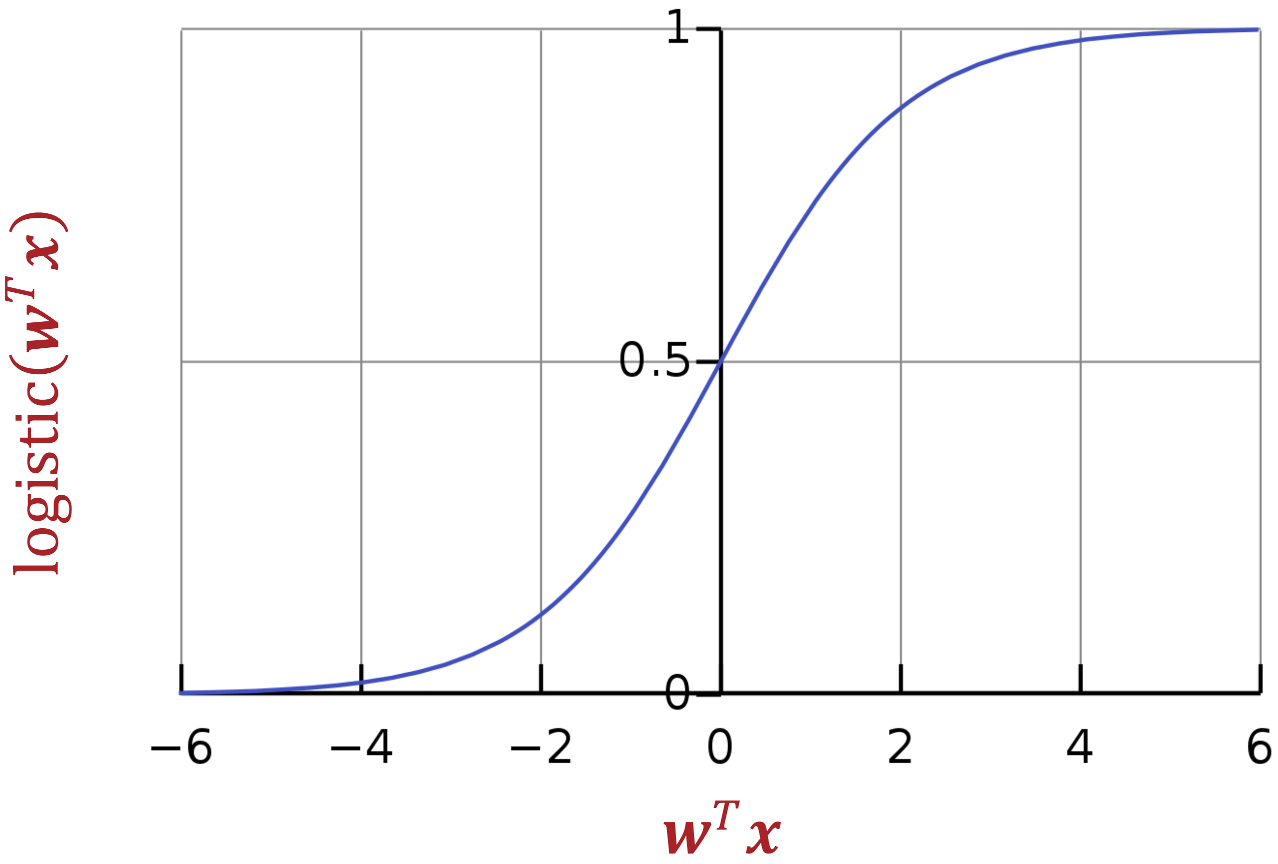
1. $P(Y = 0|\boldsymbol{x}) = 1 - P(Y = 1|\boldsymbol{x}) = \dfrac{1}{\exp(\boldsymbol{w}^T\boldsymbol{x}) + 1}$
2. $\dfrac{P(Y = 1|\boldsymbol{x})}{P(Y = 0|\boldsymbol{x})} = \exp(\boldsymbol{w}^T\boldsymbol{x}) \rightarrow \log\dfrac{P(Y = 1|\boldsymbol{x})}{P(Y = 0|\boldsymbol{x})} = \boldsymbol{w}^T\boldsymbol{x}$

Logistic Function



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

Why use the Logistic Function?



- Differentiable everywhere
- logistic: $\mathbb{R} \rightarrow [0, 1]$

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

Logistic Regression Decision Boundary

The decision boundary is linear in $\boldsymbol{x}$!

Logistic Regression Decision Boundary

The decision boundary is linear in $\boldsymbol{x}$!

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

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

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

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

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

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

Logistic Regression Decision Boundary

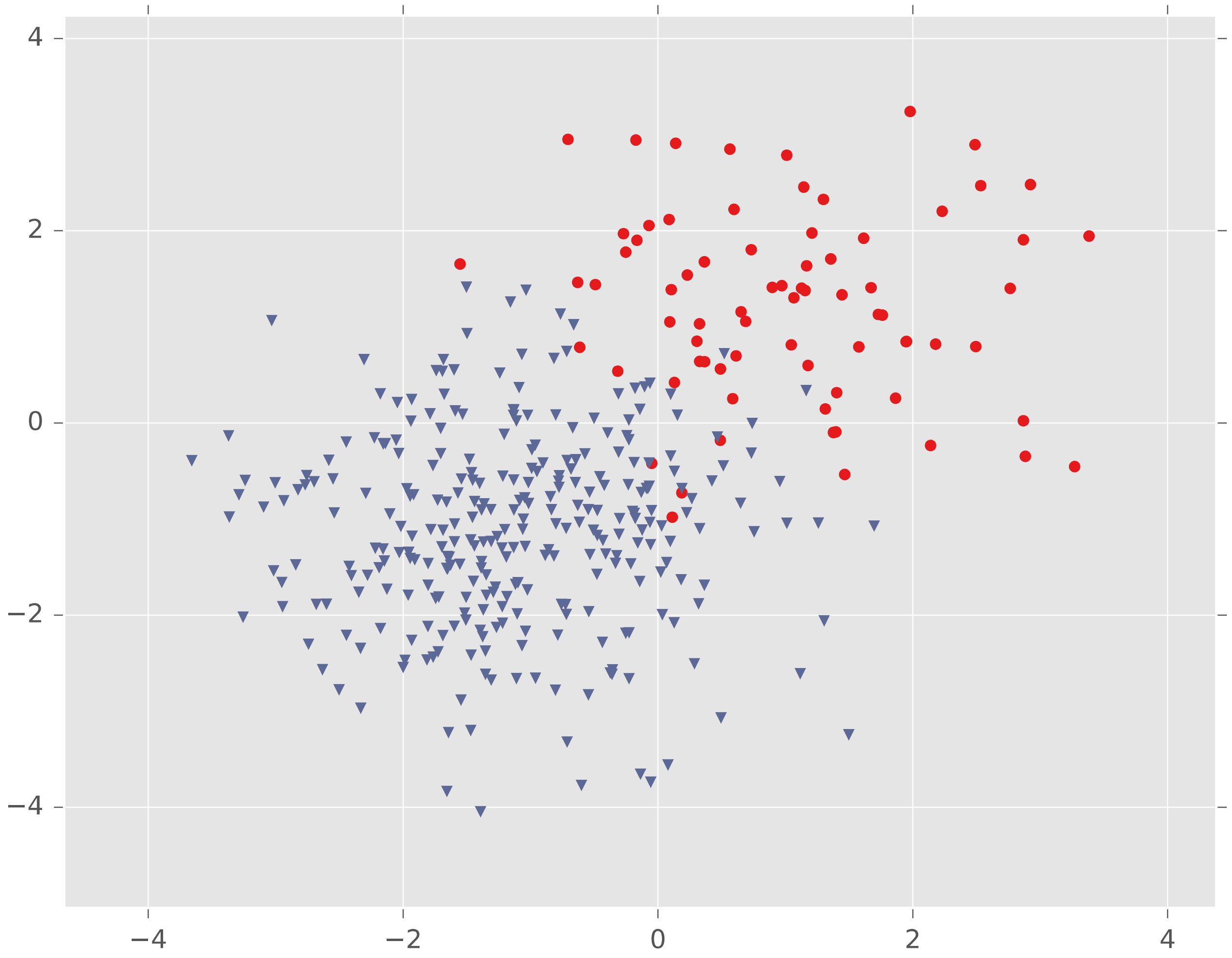


Figure courtesy of Matt Gormley

Logistic Regression Decision Boundary

Figure courtesy of Matt Gormley

Logistic Regression Decision Boundary

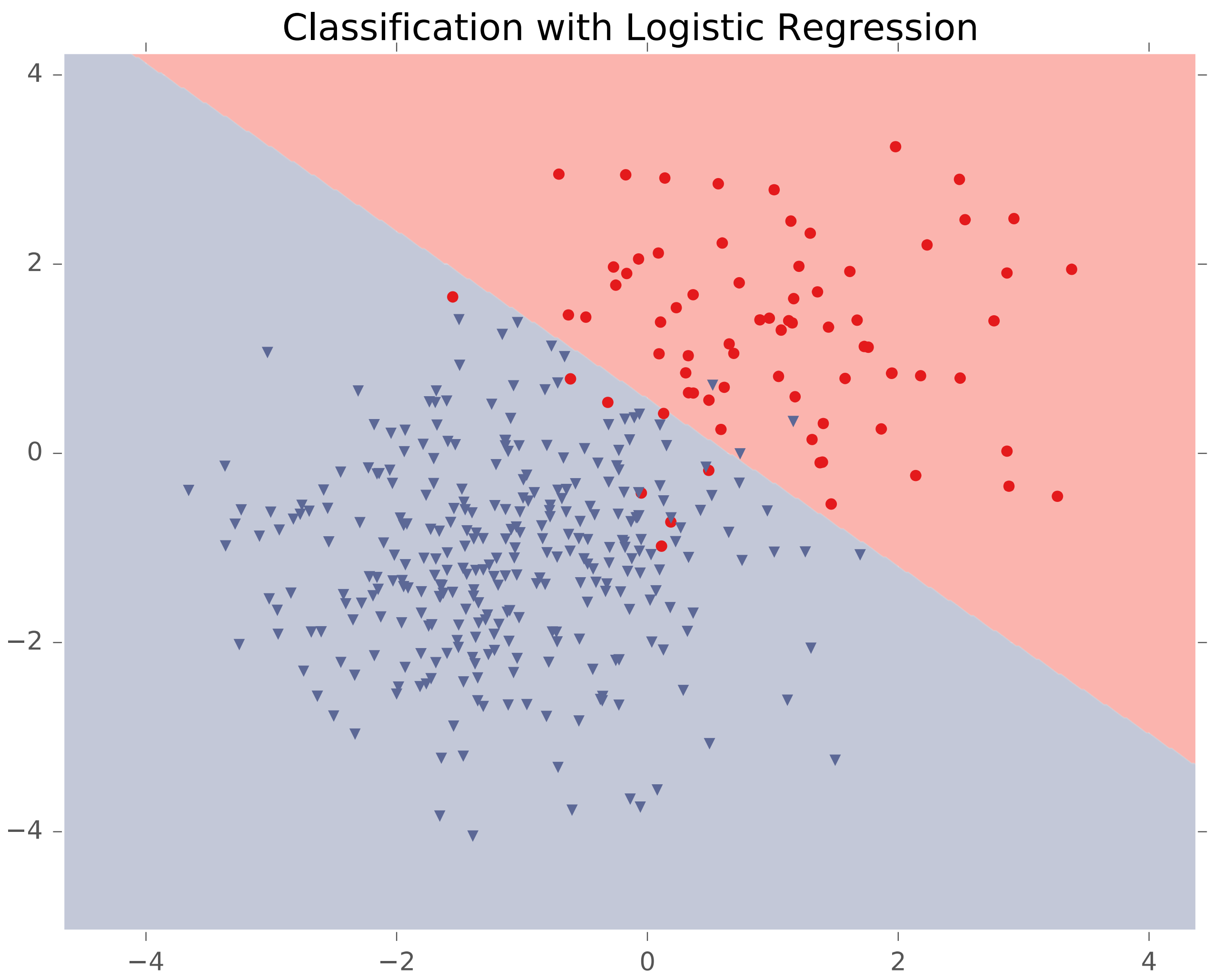


Figure courtesy of Matt Gormley

General Recipe for Machine Learning

- Define a model space and model parameters

- Write down an objective function

- Optimize the objective w.r.t. the model parameters

Recipe for Logistic Regression

- Define a model space and model parameters
  - Assume independent, identically distributed (iid) data
  - Assume $P(Y = 1|X) = \text{logistic}(\boldsymbol{w}^T\boldsymbol{x})$
  - Parameters: $\boldsymbol{w} = [w_0, w_1, \dots, w_D]$
- 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

$$\ell_{\mathcal{D}}(\boldsymbol{w}) =$$

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

Find $\boldsymbol{w}$ that minimizes

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

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

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

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

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

Minimizing the Negative Conditional (log-)Likelihood

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

Minimizing the Negative Conditional (log-)Likelihood

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

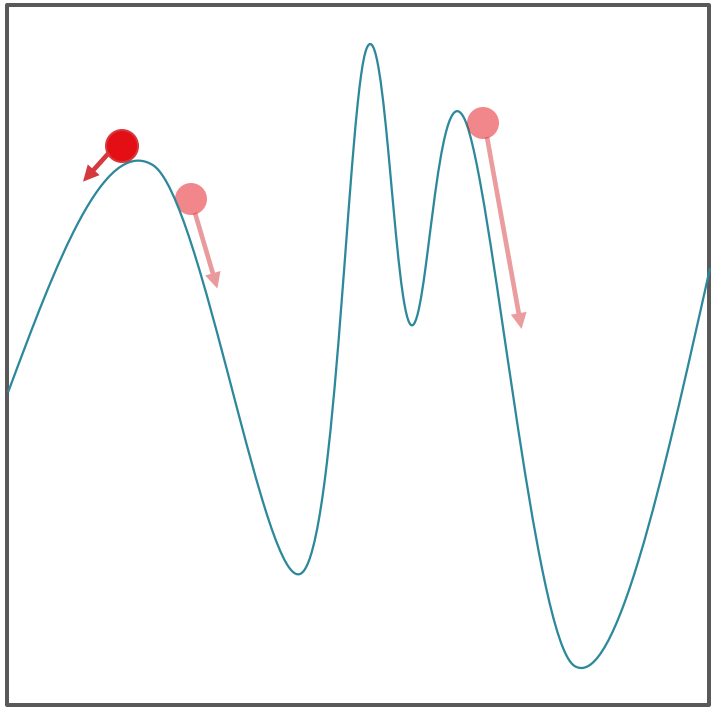
$$\nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}(\boldsymbol{w}) = -\sum_{n=1}^{N} y^{(n)} \nabla_{\boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{x}^{(n)} - \nabla_{\boldsymbol{w}} \log\left(1 + \exp\left(\boldsymbol{w}^T \boldsymbol{x}^{(n)}\right)\right)$$

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

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

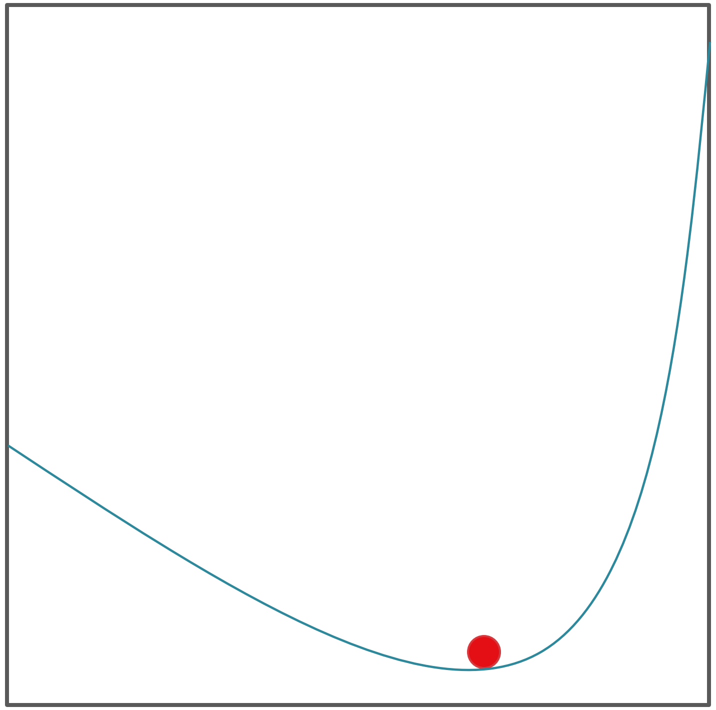
Recall: Gradient Descent

- 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:

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

   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

- 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:

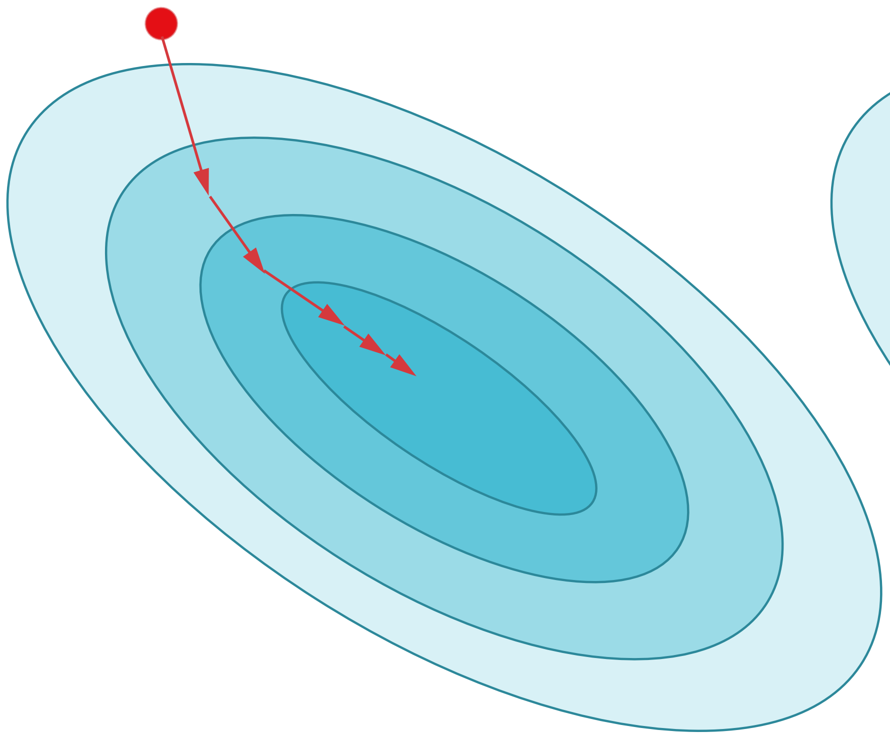
$$E\left[\nabla_{\boldsymbol{w}} \ell_{\boldsymbol{x}^{(n)}, y^{(n)}}\left(\boldsymbol{w}^{(t)}\right)\right] = \frac{1}{N} \sum_{n=1}^{N} \nabla_{\boldsymbol{w}} \ell^{(n)}\left(\boldsymbol{w}^{(t)}\right)$$

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

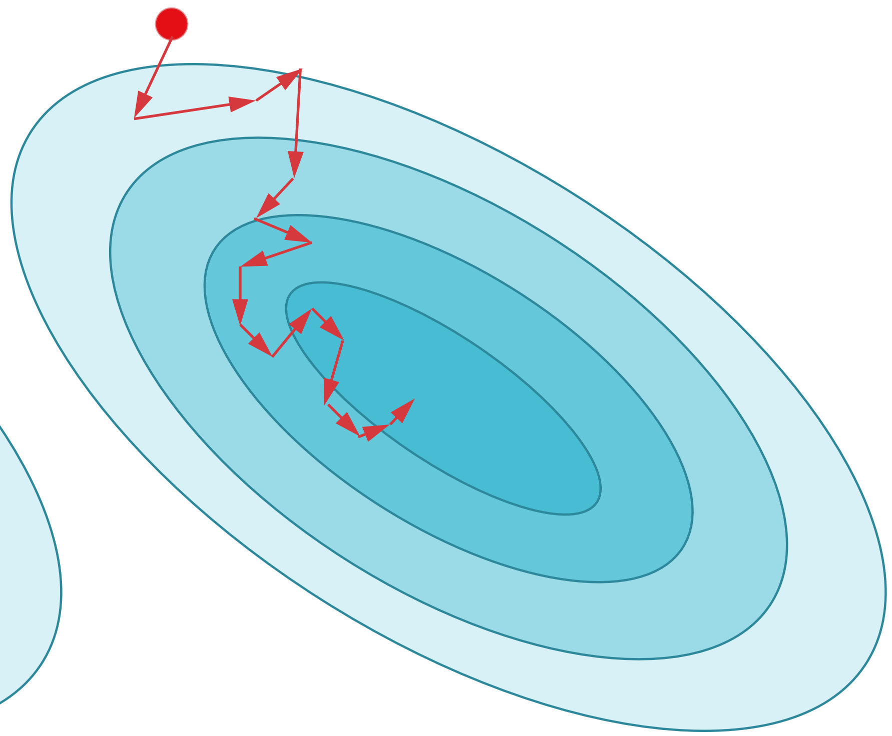
$$= \frac{1}{N} \nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)$$

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

Stochastic Gradient Descent vs. Gradient Descent



Gradient Descent



Stochastic Gradient Descent

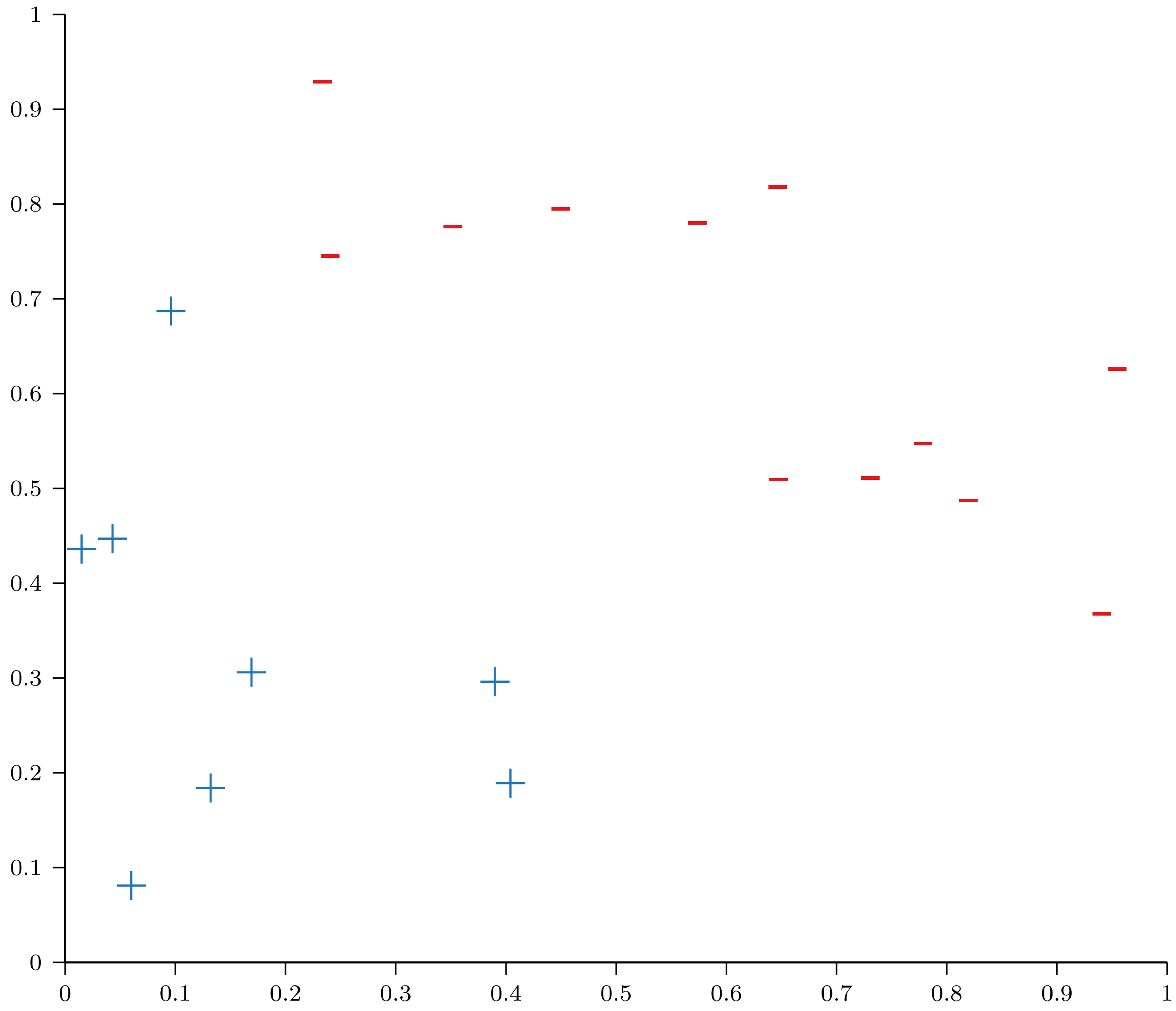
Mini-batch Stochastic Gradient Descent

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

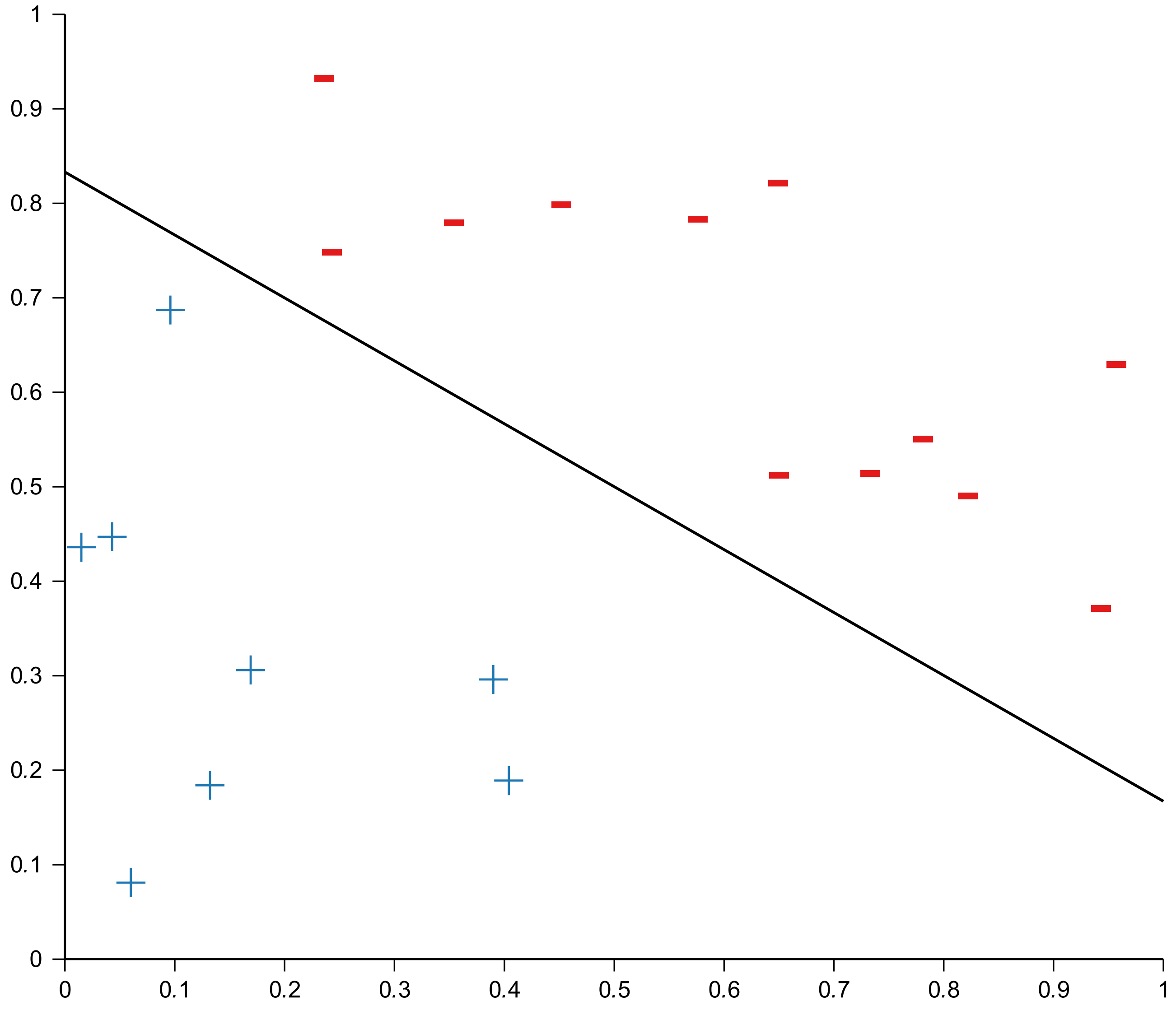
1. Initialize $\boldsymbol{w}^{(0)}$ to all zeros and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. Randomly sample B data points from $\mathcal{D}$:
   $$\mathcal{D}_{batch}\left\{\left(\boldsymbol{x}^{(b)}, y^{(b)}\right)\right\}_{b=1}^{B}$$
   b. Compute the gradient w.r.t. the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

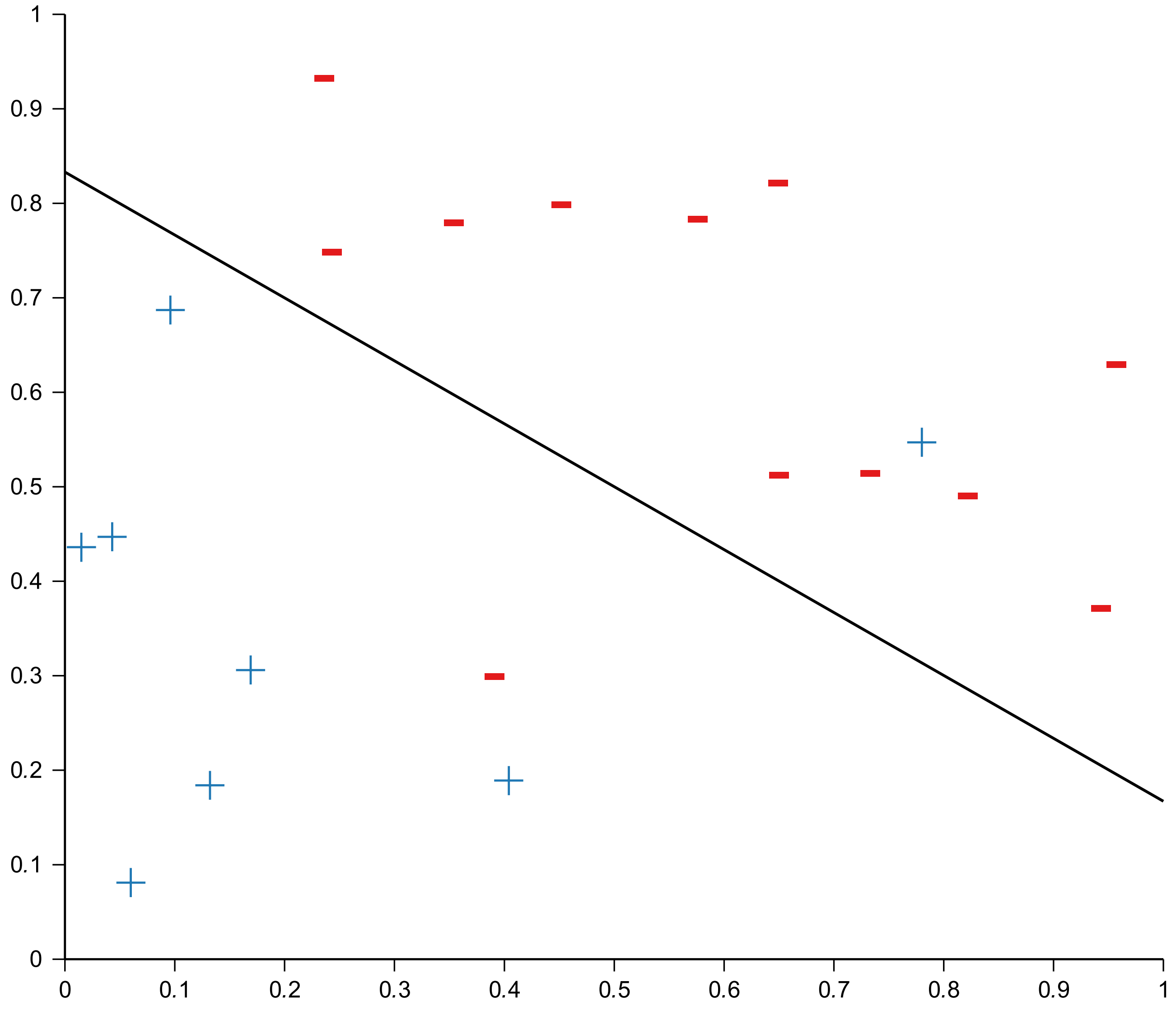
Linear Models



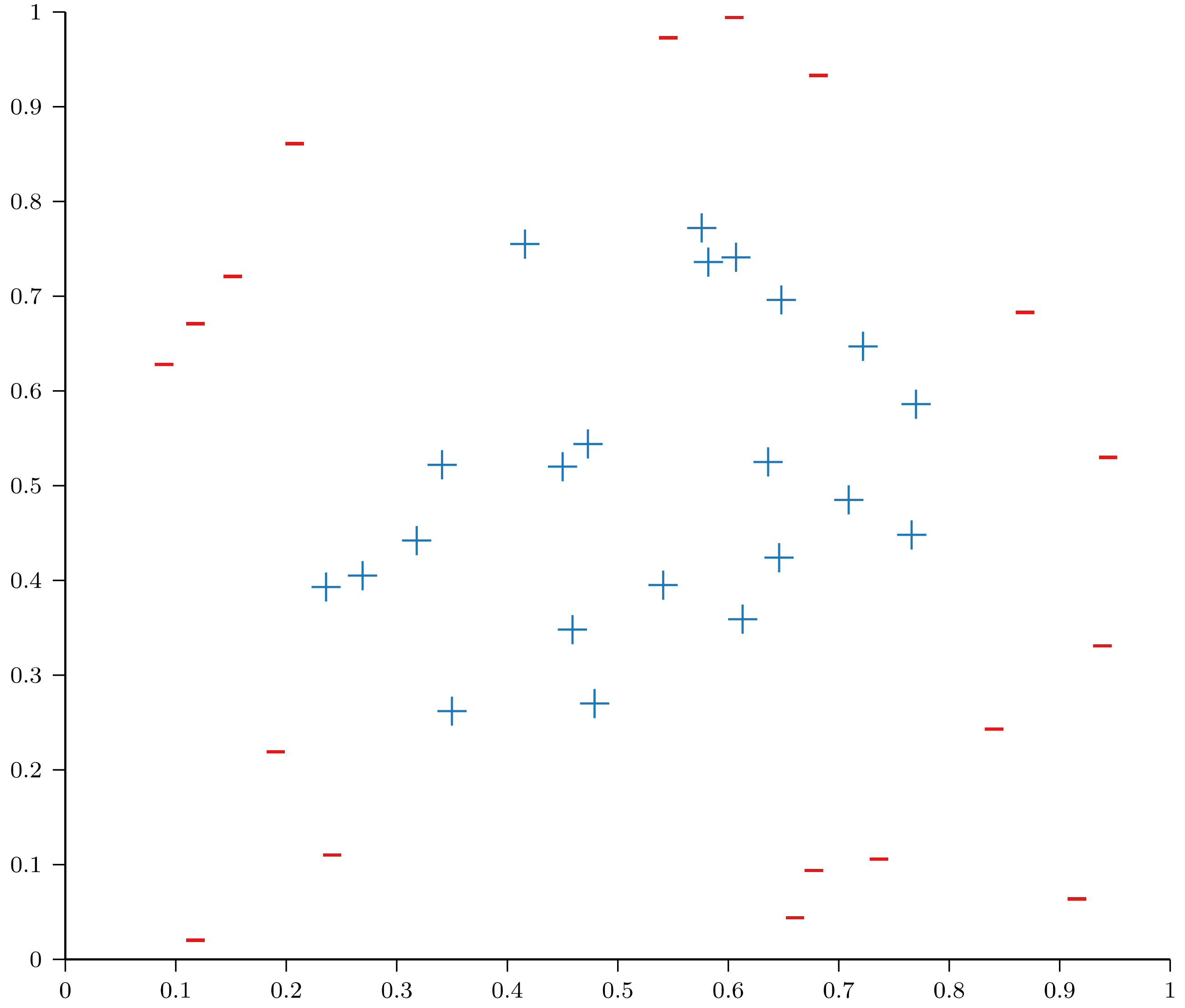
Linear Models



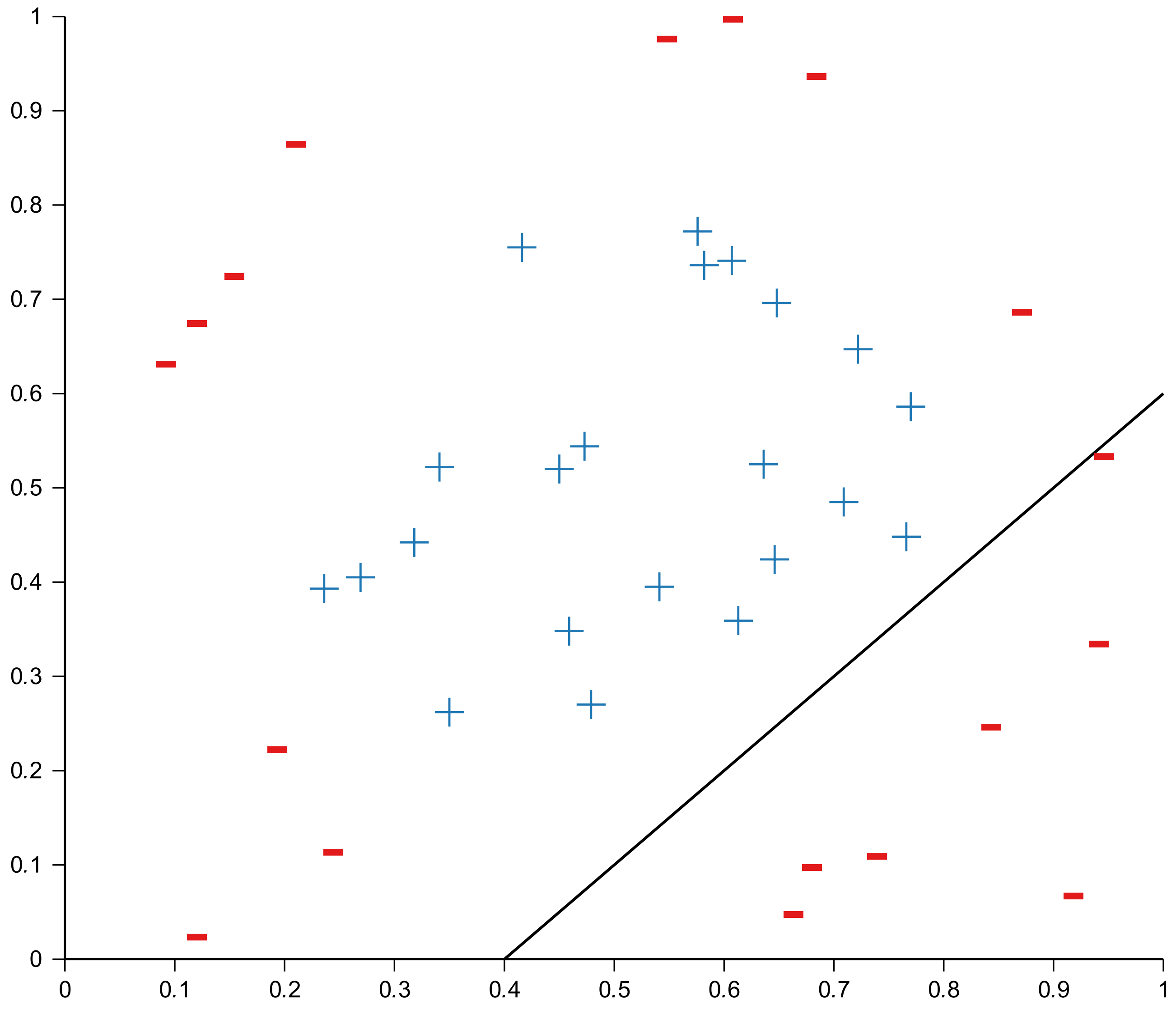
Linear Models



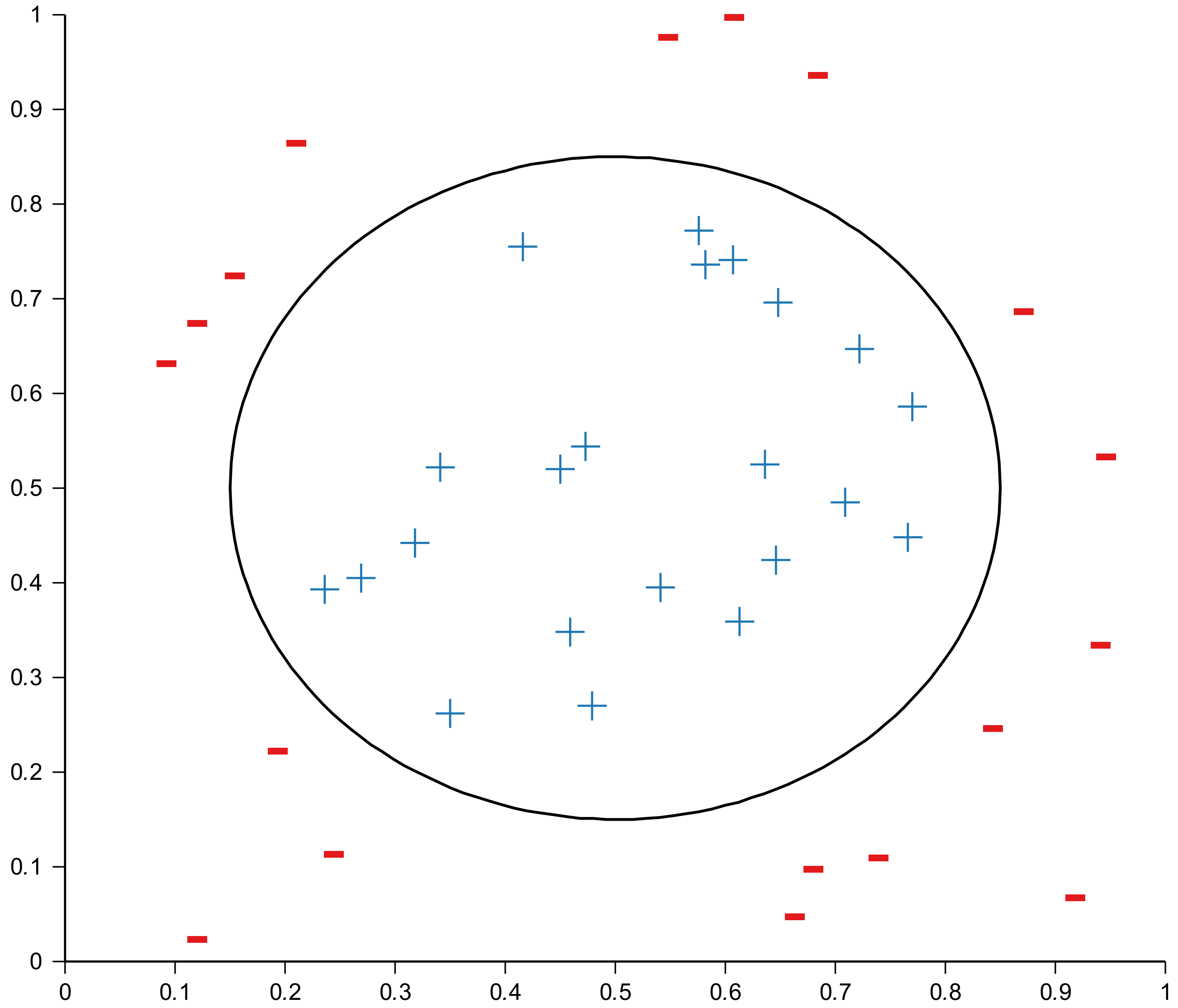
Linear Models?



Linear Models?



Nonlinear Models

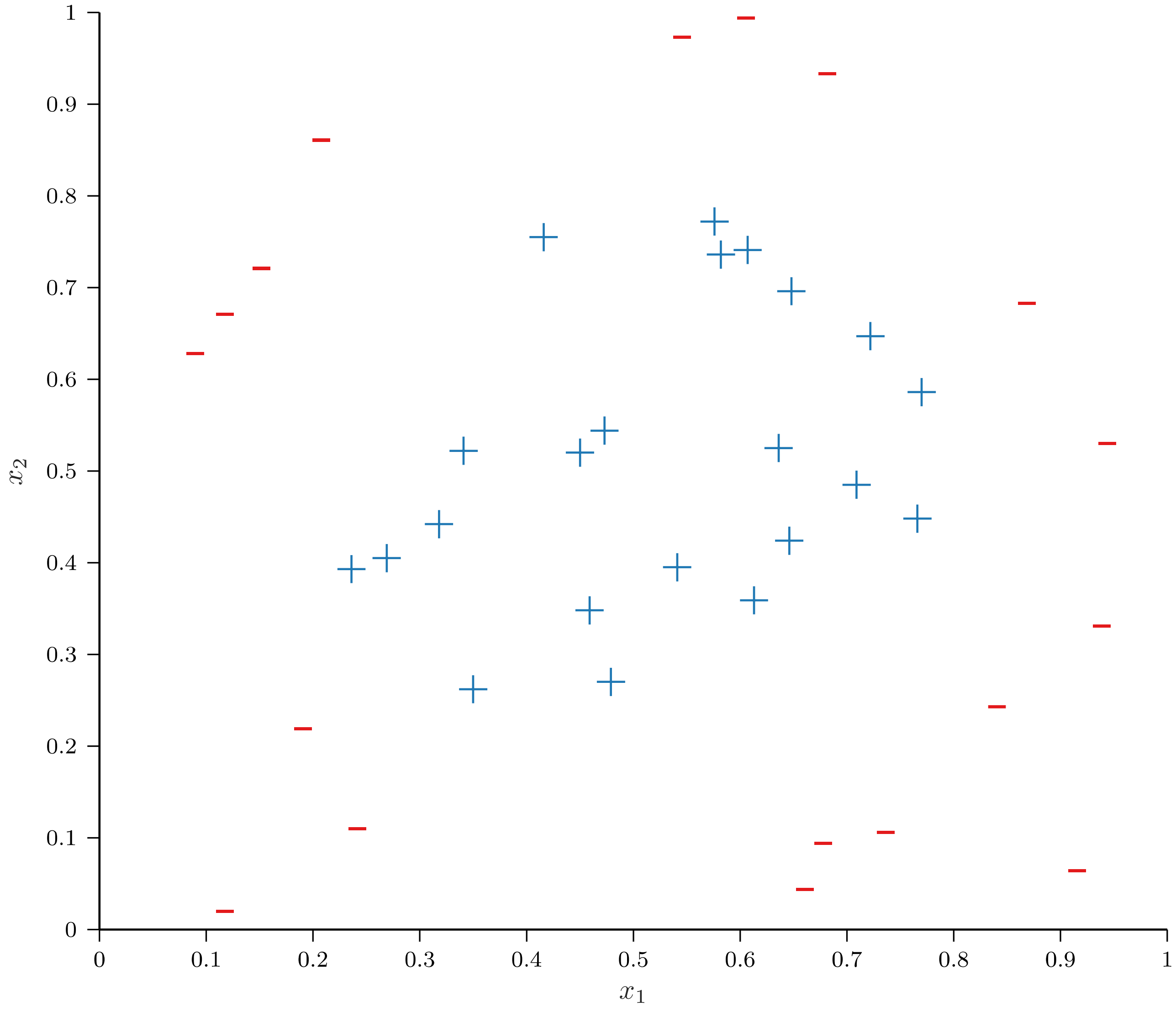


Feature Transforms

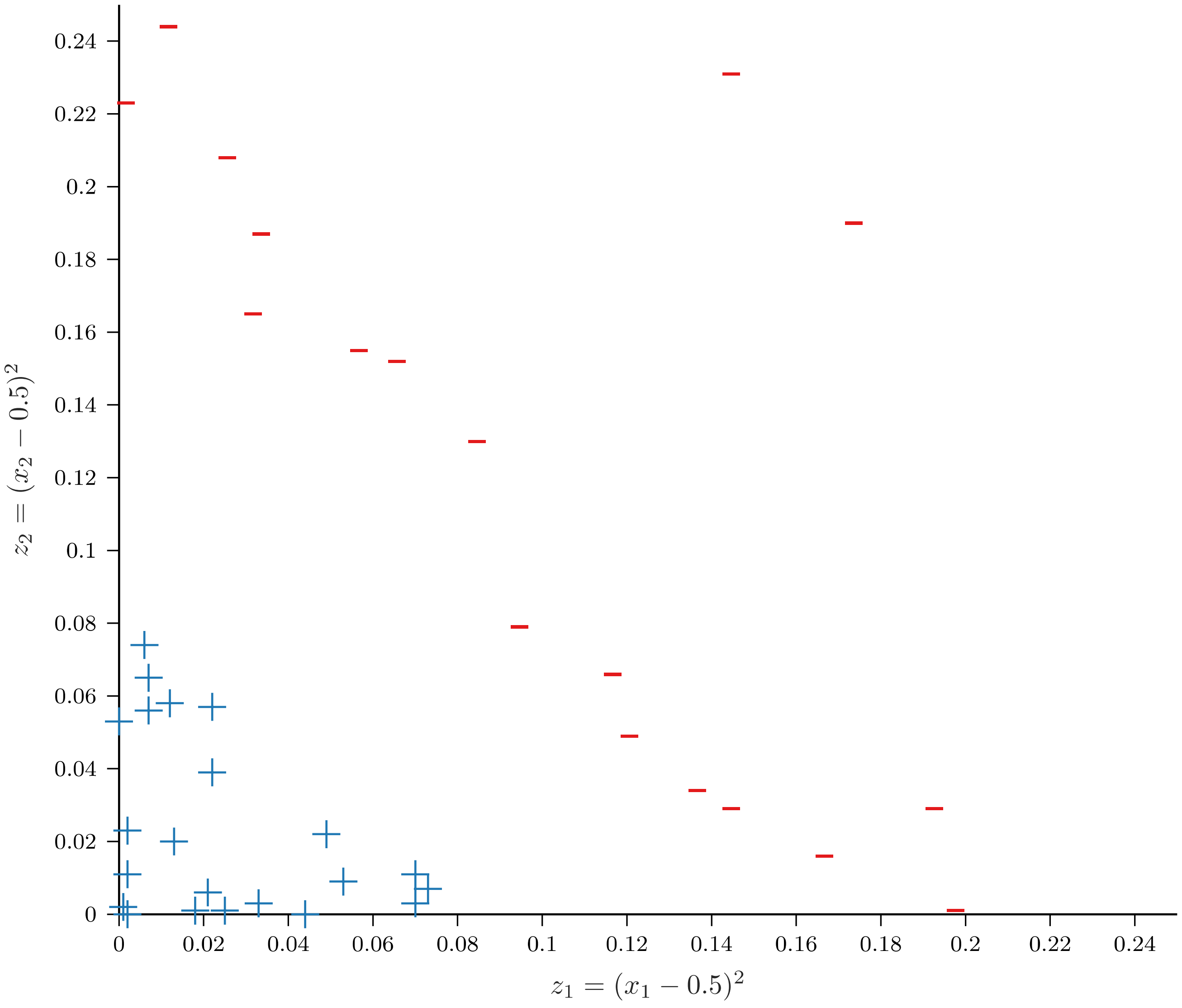
- Given D-dimensional inputs $\boldsymbol{x} = [x_1, \dots, x_D]$, first compute some transformation of our input, e.g.,

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

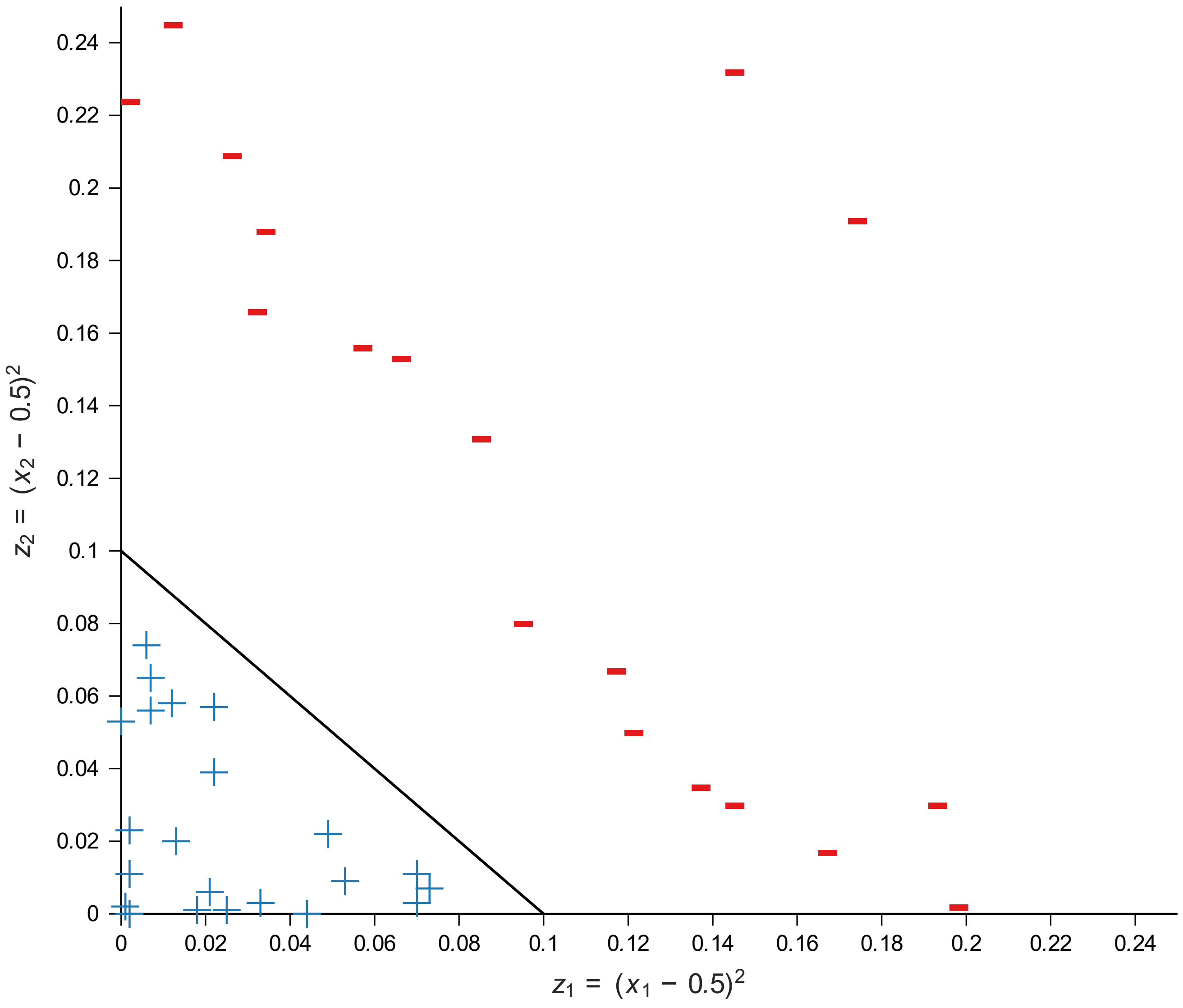
Nonlinear Models



Nonlinear Models



Nonlinear Models



Nonlinear Models