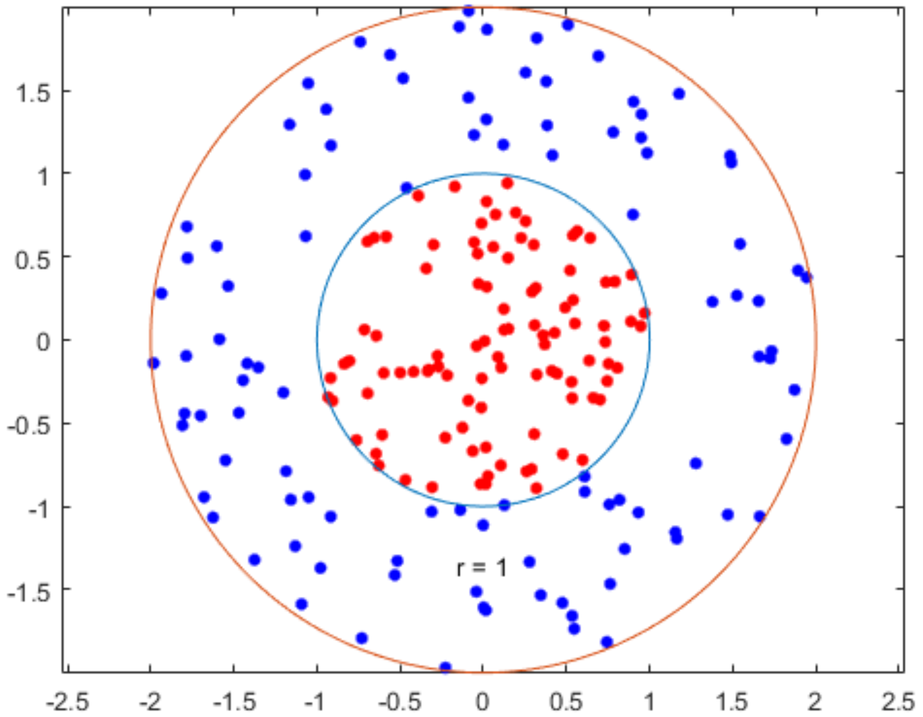
Data Representations

CMPUT 467/567: Machine Learning II

Projecting to Higher Dimensions Allows for Separability

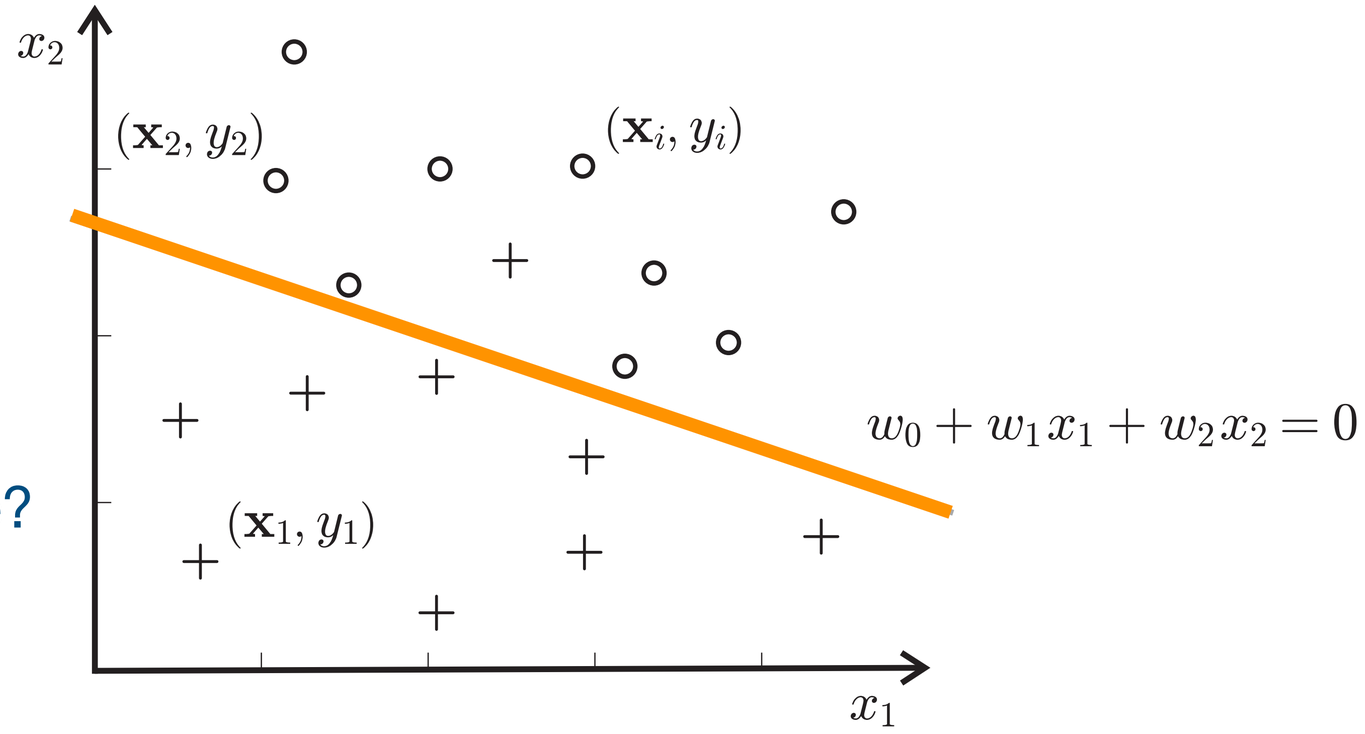
- Consider this simple example where increasing from 2 to 3 dimensions (in a careful way) allows us to obtain linear separability



Brief Reminder: Linear Separability

- Logistic regression learns a hyperplane that attempts to separate points
  - Observations on one side of decision boundary classified positive, other side negative
  - A dataset is linearly separable if there exists a linear decision boundary that perfectly classifies it

Is this dataset linearly separable?



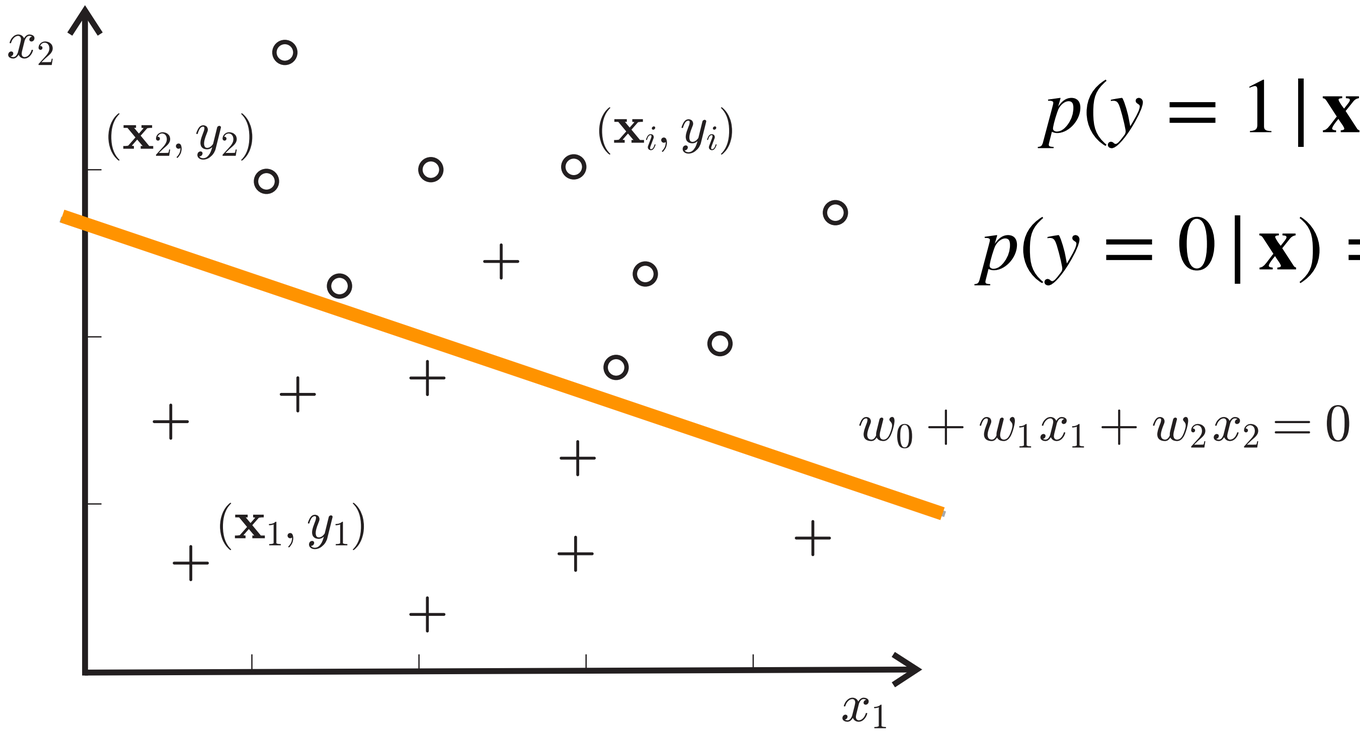
Defining the decision boundary

- Parameters $\mathbf{w}$ define a linear decision boundary
- Observations on one side of boundary classified positive, other side negative

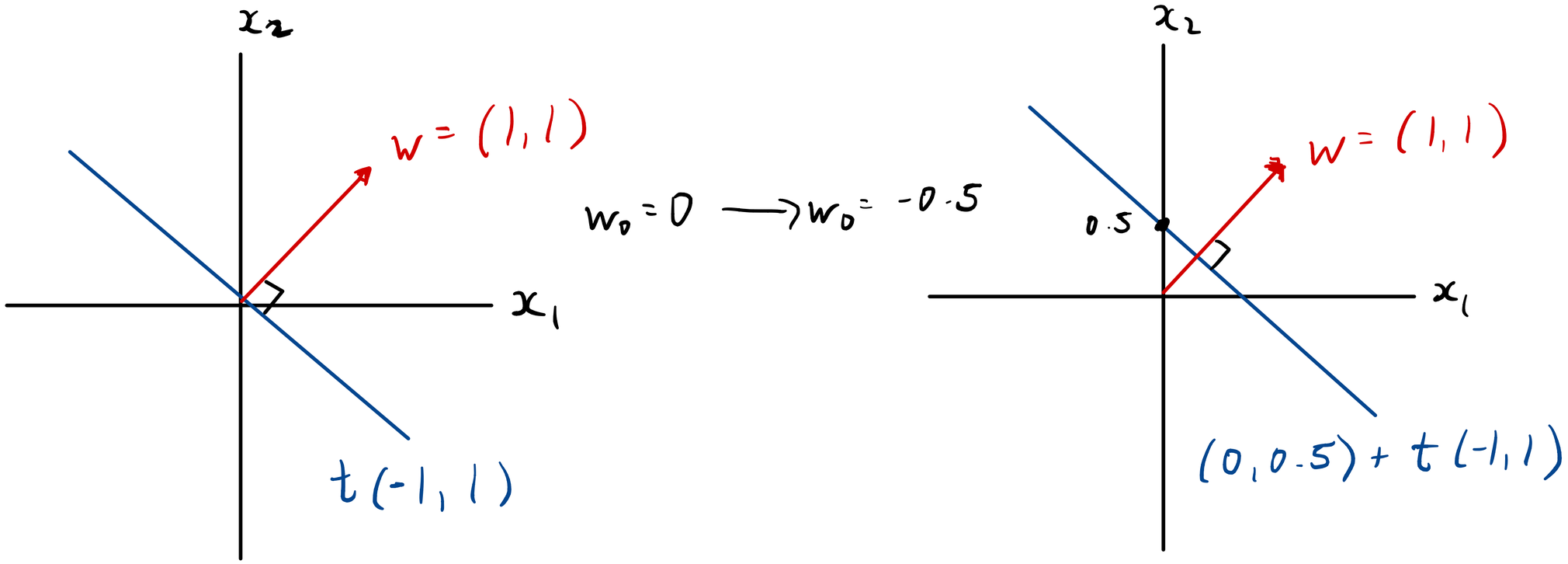
$$\boldsymbol{\phi}(\mathbf{x}) = [1, x_1, x_2]$$

$$p(y = 1 \,|\, \mathbf{x}) = \sigma(\boldsymbol{\phi}(\mathbf{x})\mathbf{w}) > 0.5 \text{ if } \boldsymbol{\phi}(\mathbf{x})\mathbf{w} > 0$$

$$p(y = 0 \,|\, \mathbf{x}) = 1 - \sigma(\boldsymbol{\phi}(\mathbf{x})\mathbf{w}) > 0.5 \text{ if } \boldsymbol{\phi}(\mathbf{x})\mathbf{w} < 0$$



Explaining the linear decision boundary

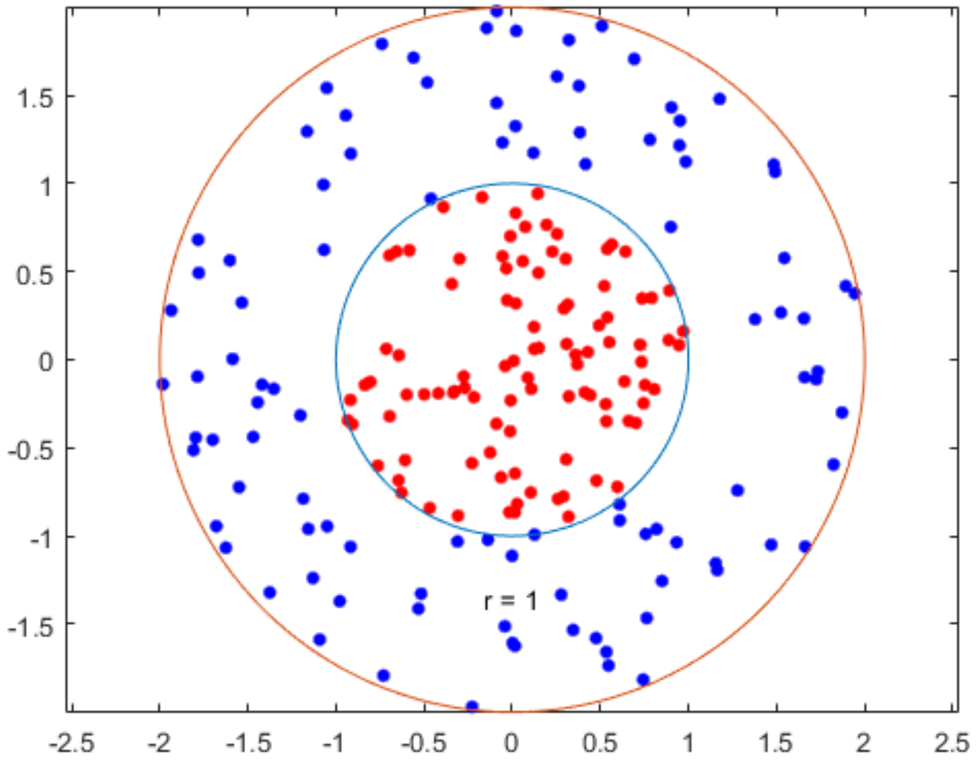


$$\langle (x_1, x_2), (w_1, w_2) \rangle$$
$$= \langle t(-1,1), (1,1) \rangle = t(0) = 0$$

$$-0.5 + \langle (0.5,0) + t(-1,1), (1,1) \rangle$$
$$= -0.5 + 0.5 + t0 = 0$$

Back to Our Example

$$x_1^2 + x_2^2 = 1 \quad f(x) = x_1^2 + x_2^2 - 1$$



$$x_1 = x_2 = 0$$

$$\implies f(x) = -1 < 0$$

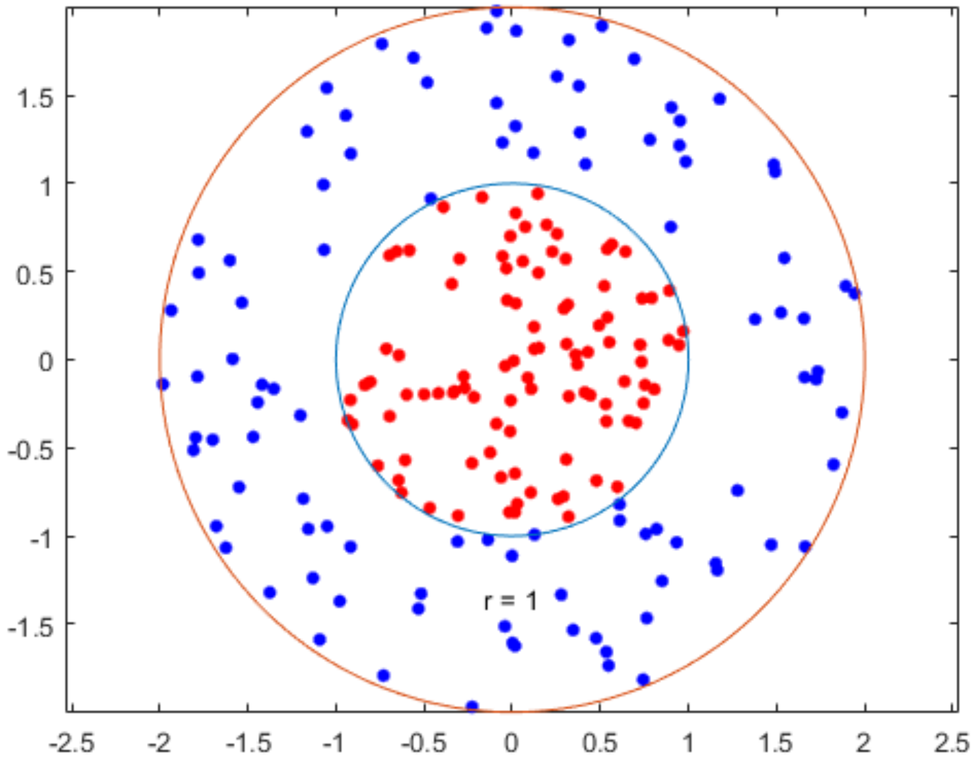
$$x_1 = 2, x_2 = -1$$

$$\implies f(x) = 4 + 1 - 1 = 4 > 0$$

How can we learn such an f(x) where f(x) > 0 predicts positive and f(x) < 0 predicts negative?

Back to Our Example

$x_1^2 + x_2^2 = 1 \quad f(x) = x_1^2 + x_2^2 - 1$



$$x_1 = x_2 = 0$$
$$\Longrightarrow \quad f(x) = -1 < 0$$

$$x_1 = 2, x_2 = -1$$
$$\Longrightarrow \quad f(x) = 4 + 1 - 1 = 4 > 0$$

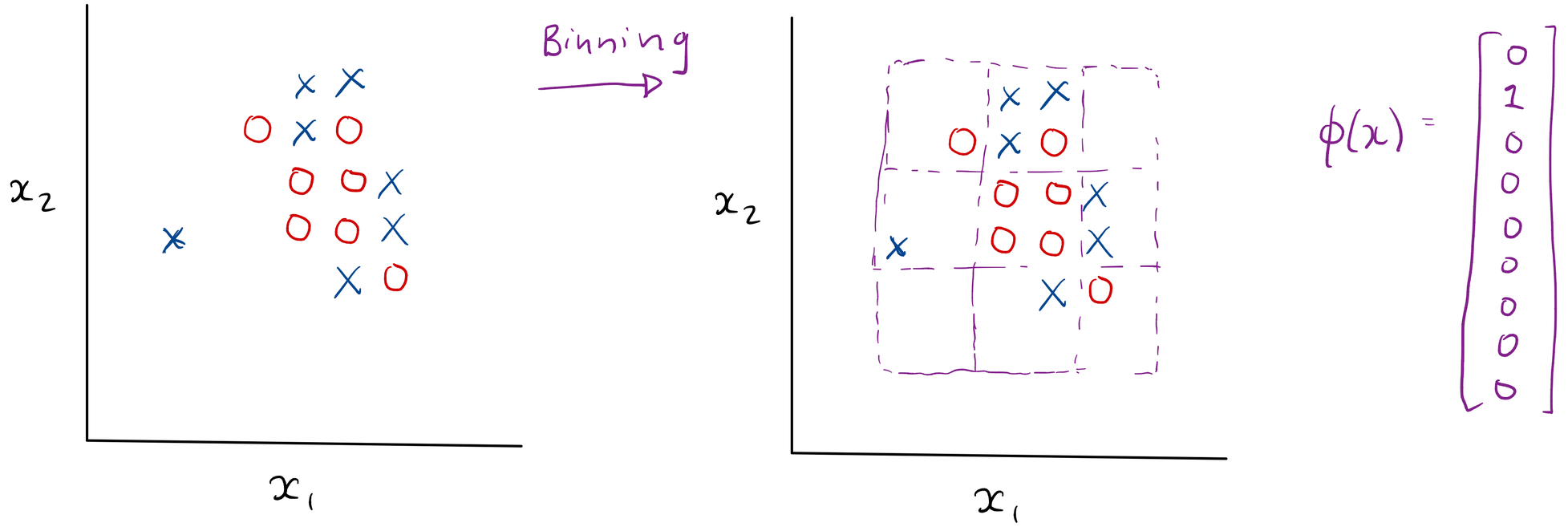
$f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})\mathbf{w}$ for $\boldsymbol{\phi}(\mathbf{x}) = [1, x_1^2, x_2^2]$

How can we learn such an f(x) where f(x) > 0 predicts positive and f(x) < 0 predicts negative? **Use logistic regression with new features**

May have to project higher

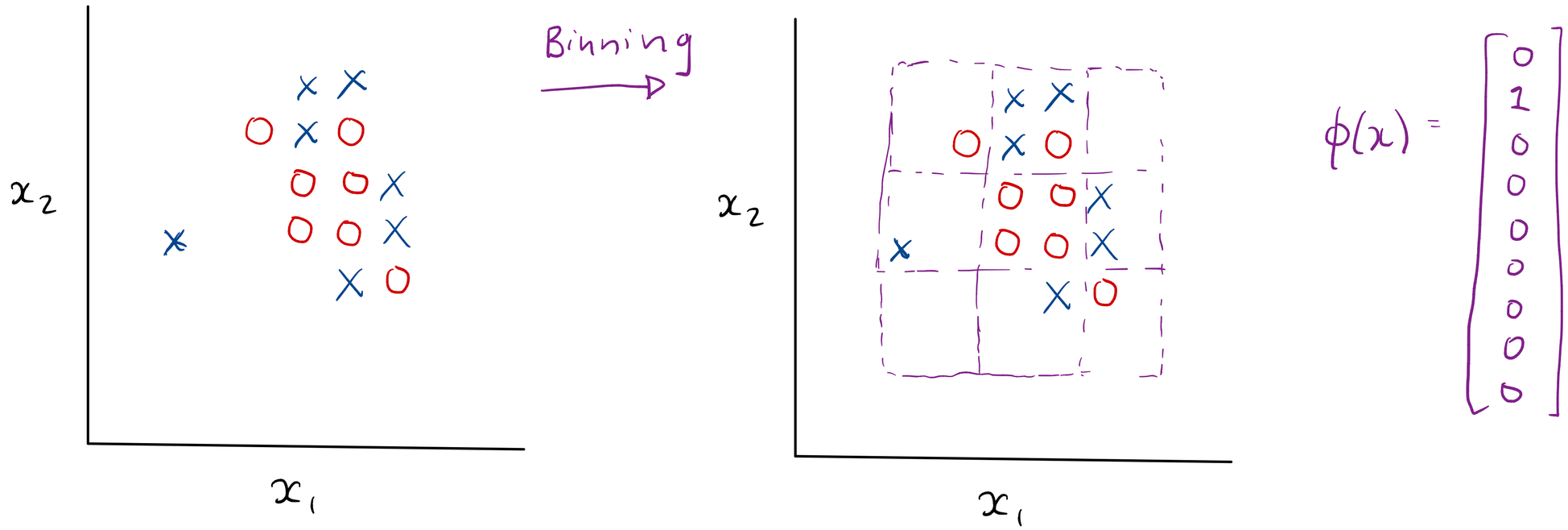
- **Cover's Theorem**: a dataset that is not linearly separable is highly likely to be separable by projecting to a higher-dimensional space with a nonlinear transformation

- One easy way to see this: consider a fine grained binning



May have to project higher

- One easy way to see this: consider a fine grained binning



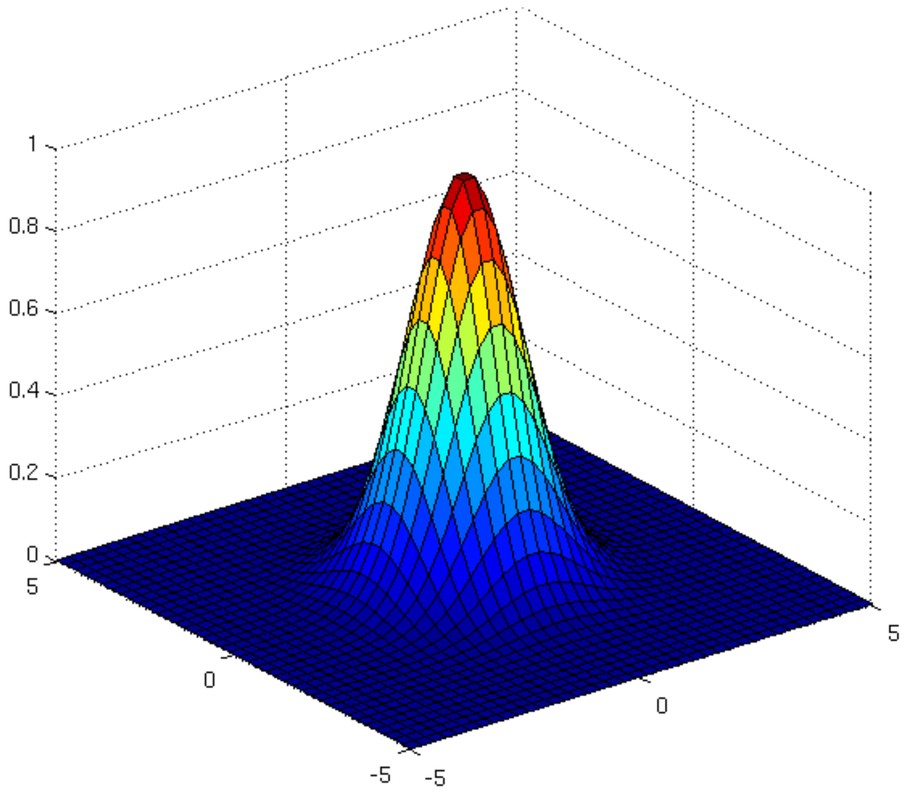
Exercise: What weights w are learned, assuming red circle is the negative class?

Recall: Classify as positive class if $\boldsymbol{\phi}(\mathbf{x})\mathbf{w} > 0$

This Theorem is One Motivation for Radial Basis Functions

$$k(\mathbf{x}, \mathbf{c}) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{c}\|_2^2\right)$$

$\mathbf{c}$ is a center or prototype

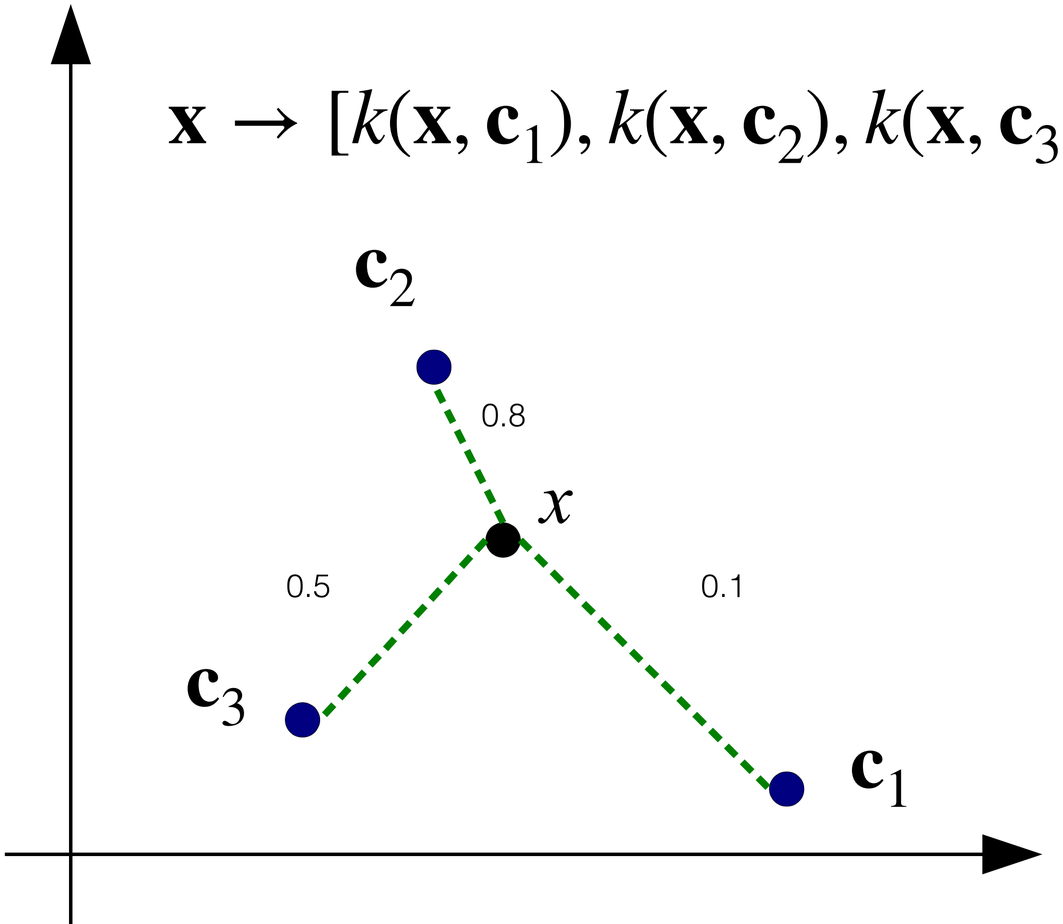


This Theorem is One Motivation for Radial Basis Functions

$$k(\mathbf{x}, \mathbf{c}) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{c}\|_2^2\right)$$

$\mathbf{c}_j$ is a center or prototype

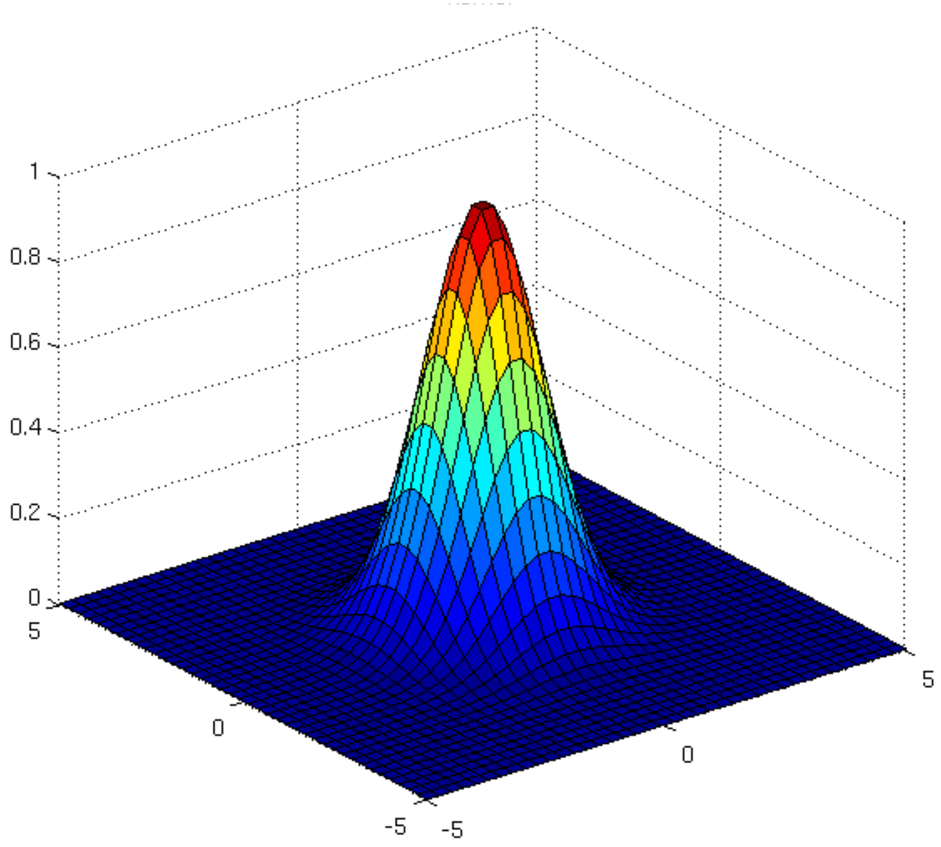
$$\mathbf{x} \rightarrow [k(\mathbf{x}, \mathbf{c}_1), k(\mathbf{x}, \mathbf{c}_2), k(\mathbf{x}, \mathbf{c}_3)] = [0.1, 0.8, 0.5]$$

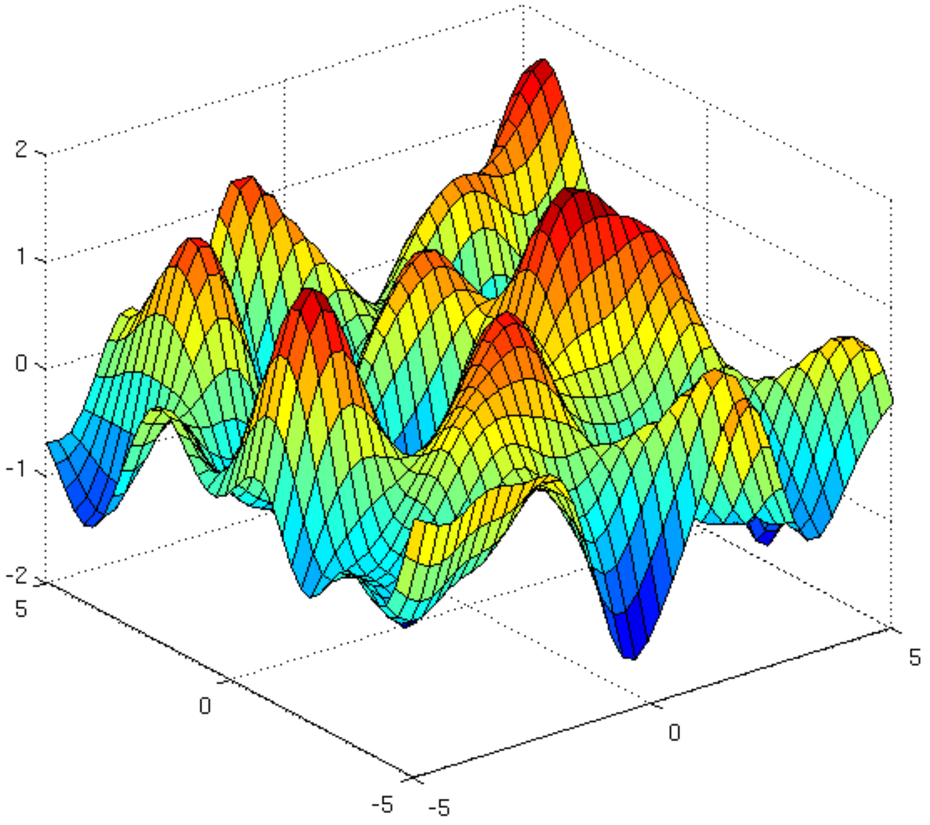


$$f(\mathbf{x}) = \sum_{j=1}^{p} k(\mathbf{x}, \mathbf{c}_j) w_j$$

Can learn a highly nonlinear function!

$$f(\mathbf{x}) = \sum_{j=1}^{p} k(\mathbf{x}, \mathbf{c}_j) w_j$$





Other (similarity) transforms

- Linear kernel: $k(\mathbf{x}, \mathbf{c}) = \langle \mathbf{x}, \mathbf{c} \rangle$

- Laplace kernel (Laplace distribution instead of Gaussian)

$$k(\mathbf{x}, \mathbf{c}) = \exp(-b\|\mathbf{x} - \mathbf{c}\|_1)$$

- Binning transformation

$$k(\mathbf{x}, \mathbf{c}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ in box around } \mathbf{c} \\ 0 & \text{else} \end{cases}$$

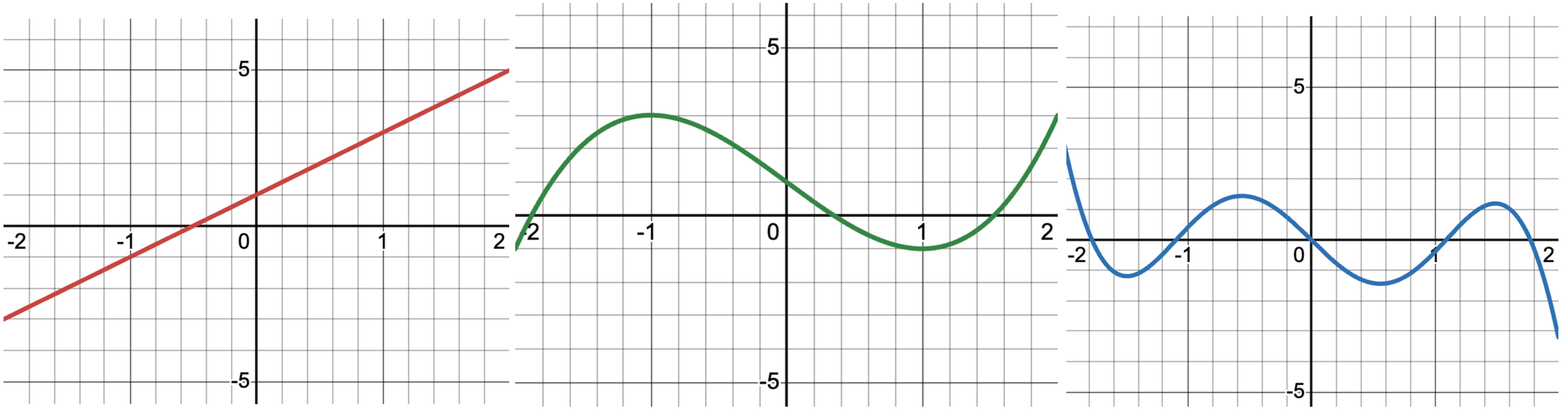
Picking prototypes

- Effectiveness of these methods depends heavily on how protoypes are picked
- One easy choice: **use all of your data**
  - Lots of features, really projected up!
- A more efficient choice: **subselect** a **representative** set of points
  - How? Many many algorithms, you will use l1 regularization later
  - Choice in your first assignment: random subset

Contrasting to polynomials

- How do functions with RBF features differ from polynomial functions?

Recall polynomials functions



(a) Degree 1 polynomial *(b) Degree 3 polynomial* *(c) Degree 7 polynomial*

Figure 3.4: Visualizing functions using polynomial features, with increasing degree. (a) Linear function (degree 1 polynomial) $f(x) = 2x + 1$ *with weights* $w_0 = 1$, $w_1 = 2$ *and feature vector* $\mathbf{x} = [1\ x]$. *(b) Degree 3 polynomial* $f(x) = x^3 - 3x + 1$ *with weights* $w_0 = 1$, $w_1 = -3$, $w_2 = 0$, $w_3 = 1$ *and feature vector* $\mathbf{x} = [1\ x\ x^2\ x^3]$. *(c) Degree 7 polynomial* $f(x) = 0.1x^7 - 1.5x^5 + 5x^3 - 4x$ *with weights* $w_0 = 0$, $w_1 = -4$, $w_2 = 0$, $w_3 = 5$, $w_4 = 0$, $w_5 = -1.5$, $w_6 = 0$, $w_7 = 0.1$ *and feature vector* $\mathbf{x} = [1\ x\ x^2\ x^3\ x^4\ x^5\ x^6\ x^7]$.

Contrasting RBFs features to polynomials features

- **Generalize differently**
  - **RBF** features are highly **localized**, **polynomial** features are **global**
  - If new point $\mathbf{x}$ far from observed prototypes, RBF features are all nearly zero, unlike polynomial features
  - Polynomial features can provide weird generalization outside of data (e.g., consider cubic or linear functions)

Contrasting RBFs features to polynomials features

- **Generalize differently**
  - **RBF** features are highly **localized**, **polynomial** features are **global**
- **Scale differently**
  - RBF features grow with number of prototypes, could leverage data lying on lower-dimensional manifold (lower effective dimension)
  - For degree q, polynomials grow as $\binom{d+q}{q}$. For smallish q, $\binom{d+q}{q} \approx \frac{q^d}{d!}$. Grows quickly.

Generalization Error

- **Generalization error** of a function (model) f assuming data is generated from $p(\mathbf{x}, y)$ is the **expected cost** $\mathbb{E}[\text{cost}(f(\mathbf{X}), Y)]$
-

Estimating generalization error

- Split dataset into training set of n samples and test set of k samples
  - e.g., for 10k samples, use n = 9000 and k = 1000
- $\text{GE}(f) \approx \text{Test-Error}(f) = \frac{1}{k} \sum_{i=n+1}^{n+k} \text{cost}(f(\mathbf{x}_i), y_i)$

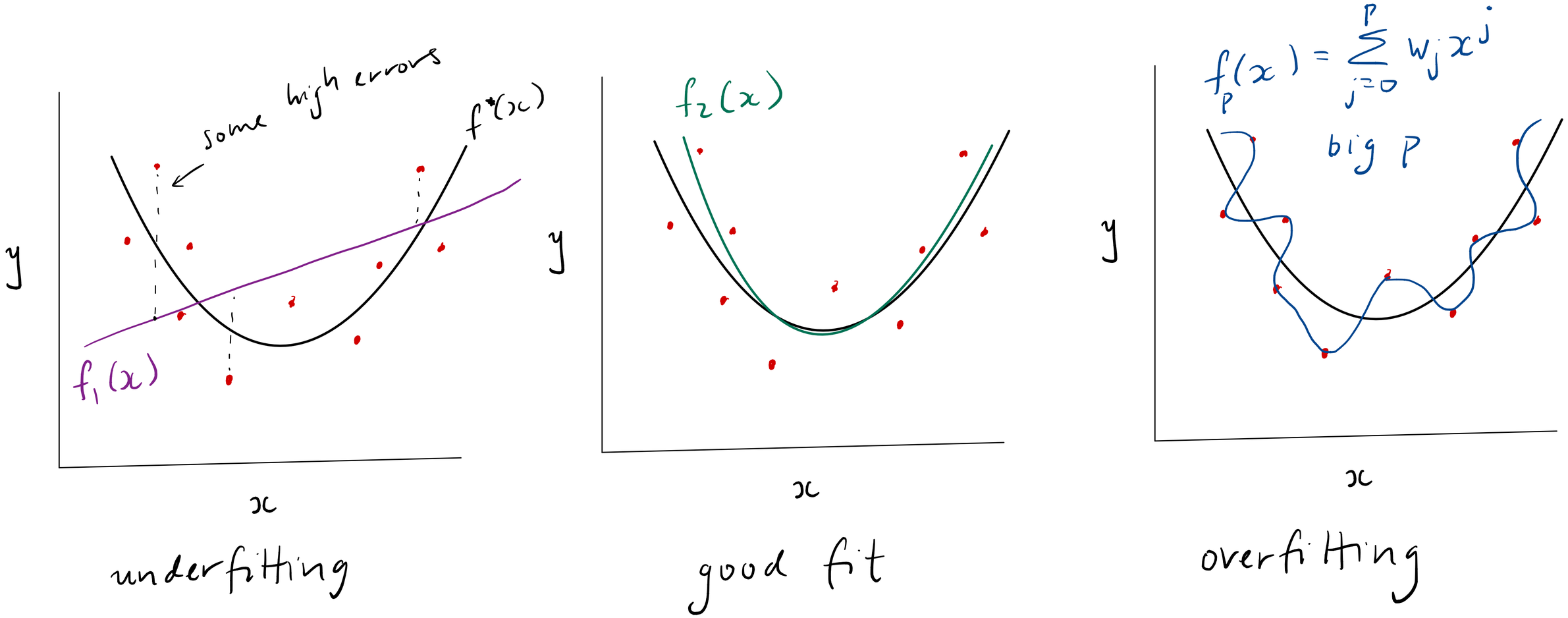
Estimating generalization error

- Split dataset into training set of n samples and test set of k samples
  - e.g., for 10k samples, use n = 9000 and k = 1000
- $\text{GE}(f) \approx \text{Test-Error}(f) = \frac{1}{k} \sum_{i=n+1}^{n+k} \text{cost}(f(\mathbf{x}_i), y_i)$
- Can also obtain confidence interval, such as 90% Student's t CI, e.g

  $[\text{Test-Error}(f) - \frac{1.646}{\sqrt{1000}}\hat{\sigma}, \text{Test-Error}(f) + \frac{1.646}{\sqrt{1000}}\hat{\sigma}]$
- **Question**: For a 90% Gaussian CI, we would have 1.645. Why do we use 1.646 instead of 1.645?

How do we know we did a good job?

Some possible outcomes

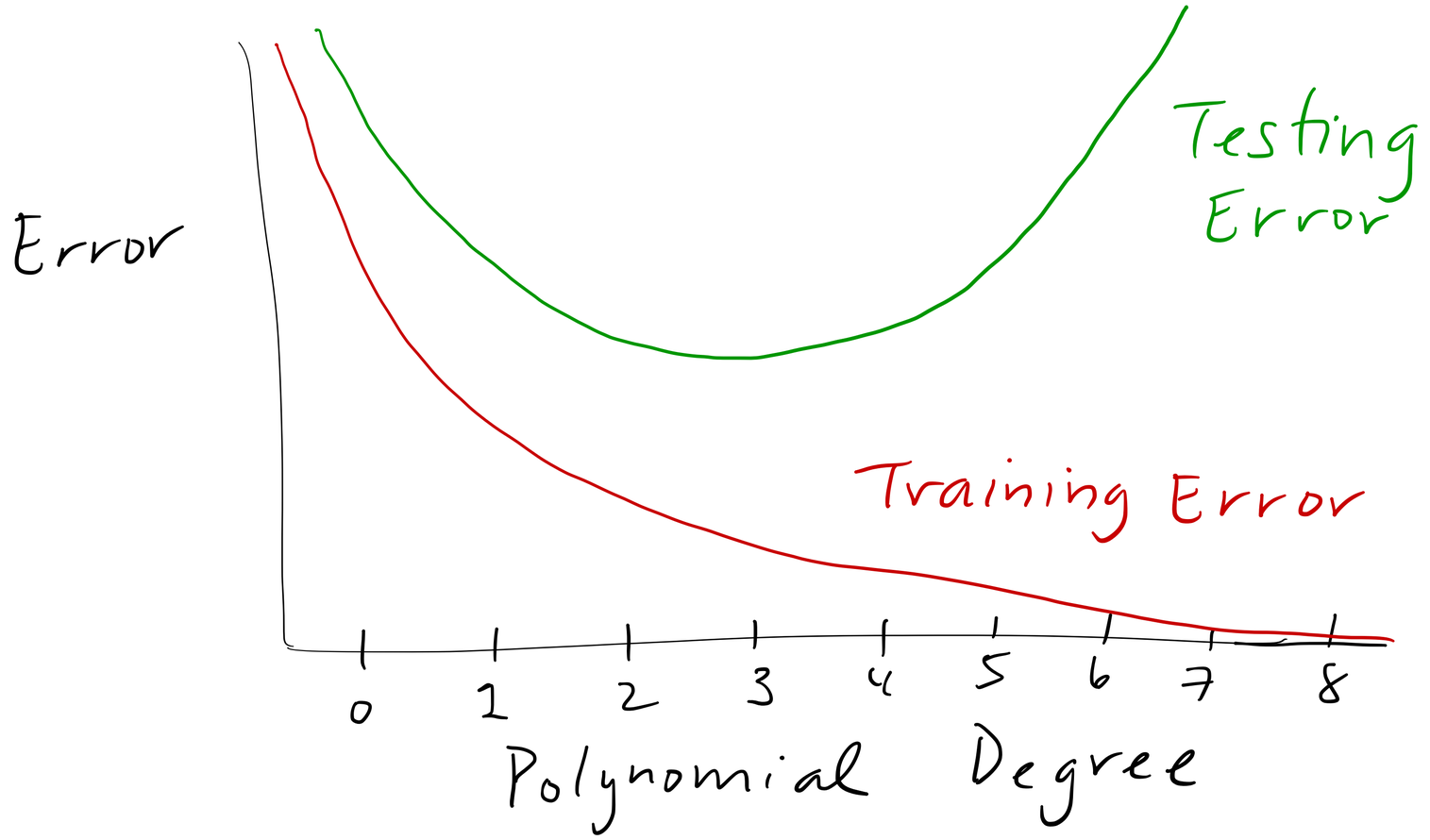


Question: what about an RBF representation with all the data as prototypes?

How do we know we did a good job?

- Signs of overfitting:
  - see an increase in test error when increasing model complexity

Hypothetical example of behavior with increasing model complexity



At what degree q do we see overfitting?

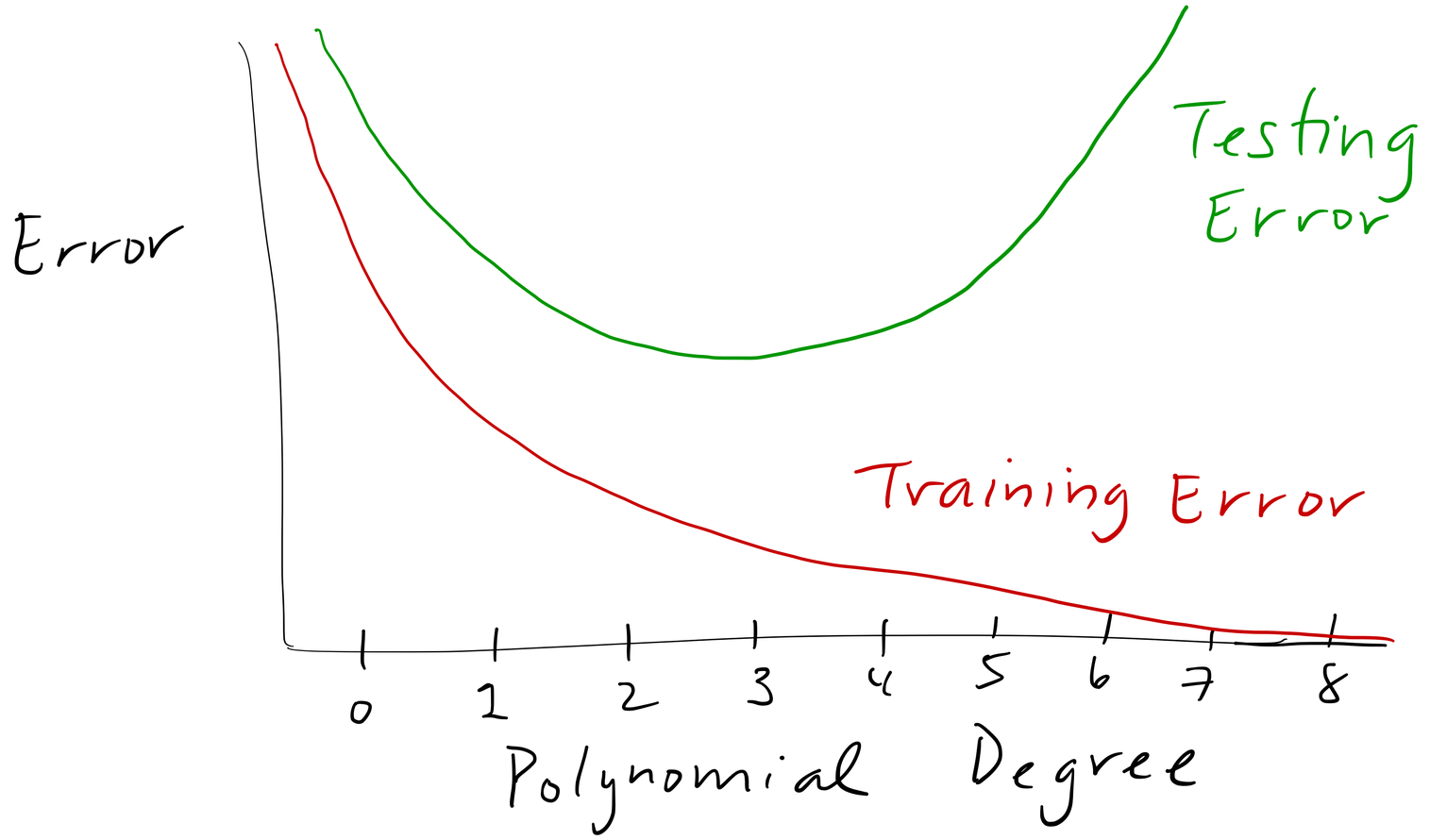
Why does training error always decrease?

Are there settings with polynomials where training error might increase?

How do we know we did a good job?

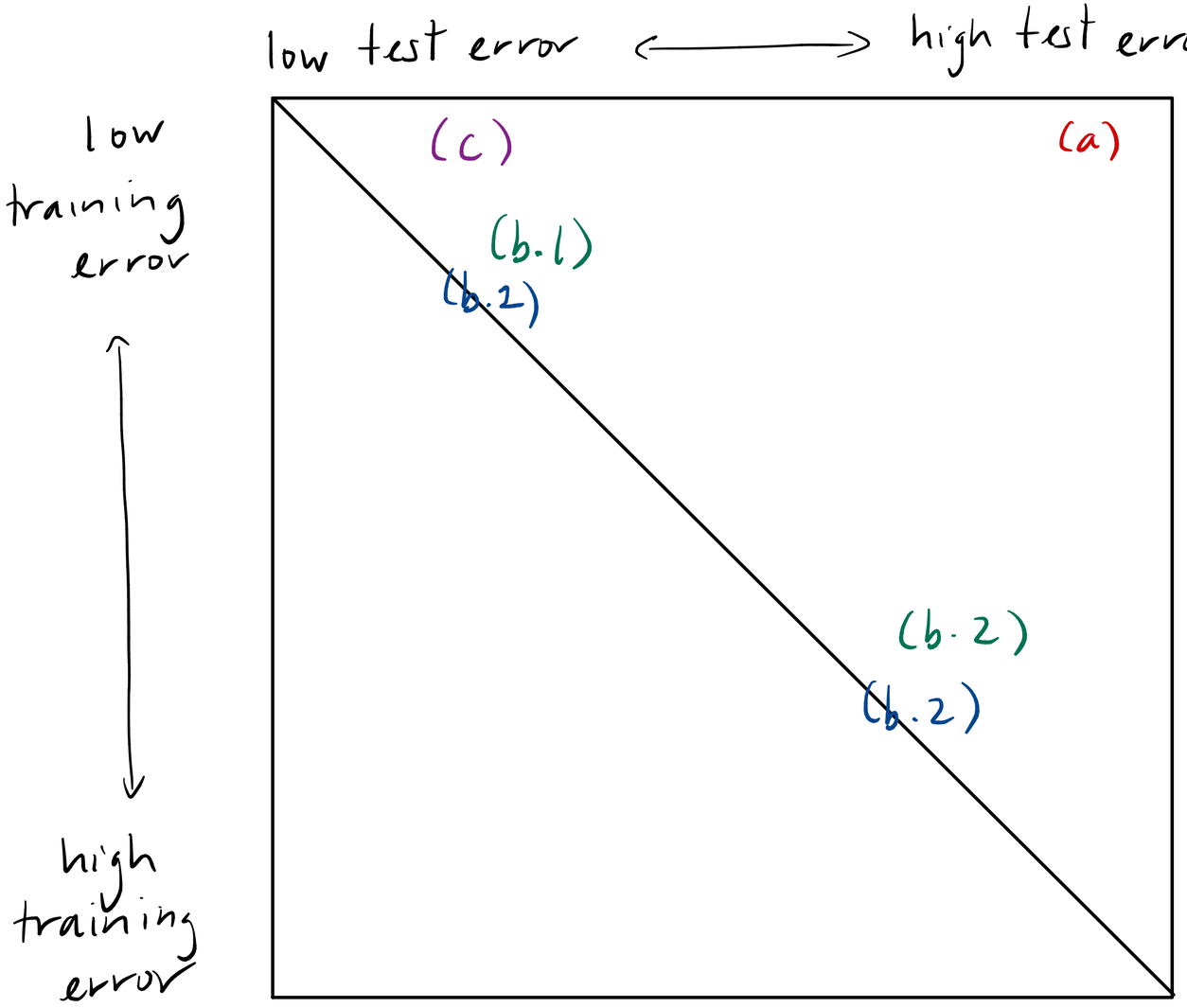
- Signs of overfitting:
  - see an increase in test error when increasing model complexity
  - high magnitude weights (see Example 6 in the notes and when we discuss bias-variance for linear regression)
- Signs of underfitting:
  - training and test error are similarly high, and can decrease both with increased model complexity

Hypothetical example of behavior with increasing model complexity



At what degree q do we see underfitting?

Contrasting different choices



(a) η small, $\mathcal{F}$ complex
(e.g. $\mathcal{F}$ = 8th degree polynomials)

(b) η small, $\mathcal{F}$ simple (e.g. linear)
Case 1: f_{true} simple
Case 2: f_{true} complex

(c) η big, $\mathcal{F}$ complex

(d) η big, $\mathcal{F}$ simple
Case 1: f_{true} simple
Case 2: f_{true} complex

Move to whiteboard

- To discuss bias and variance
- Can characterize in closed-form for linear regression