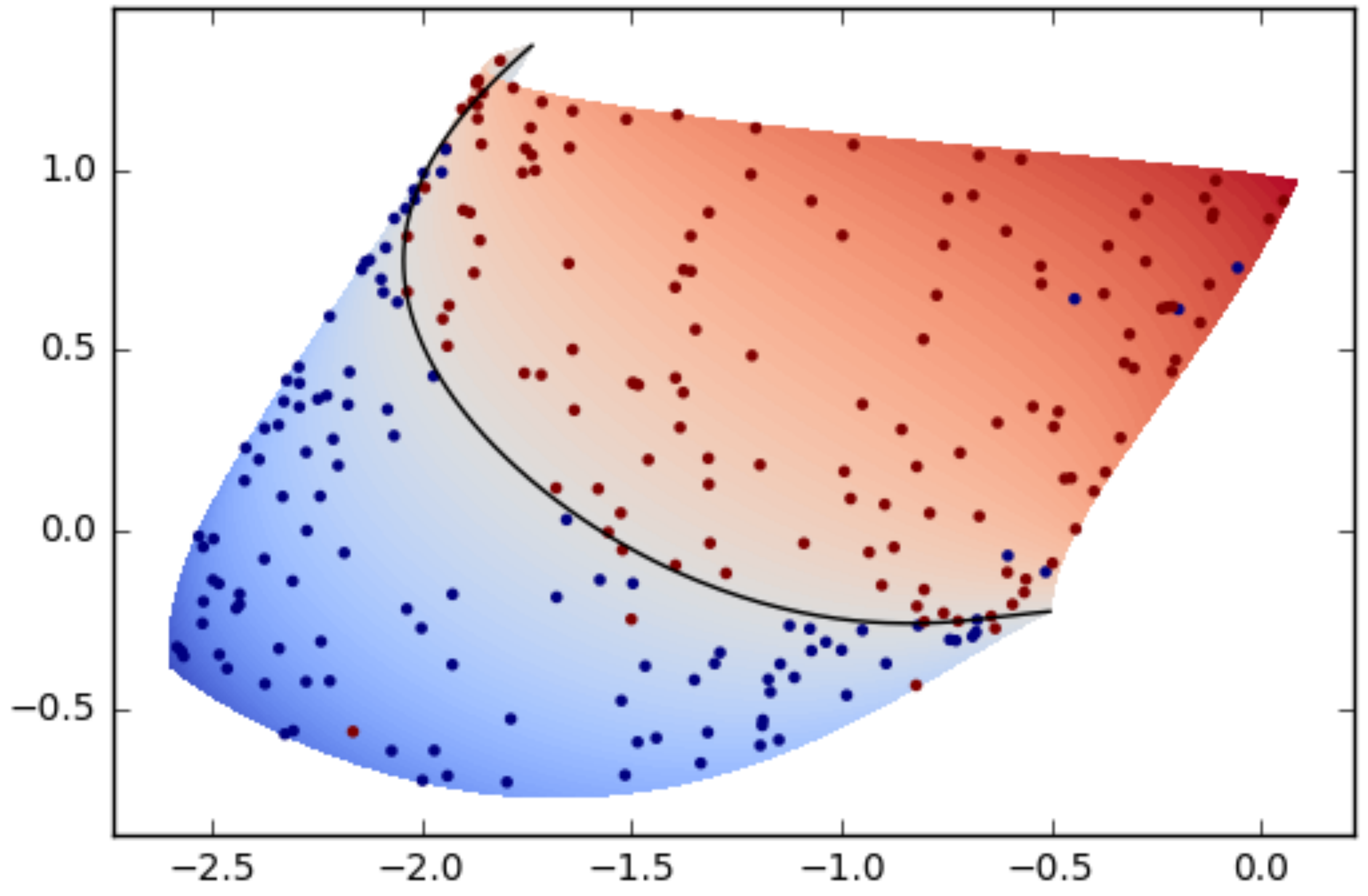
UCI University of California, Irvine

CS 273A: Machine Learning

Fall 2021

Lecture 11: Midterm Review

Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine



All slides in this course adapted from Alex Ihler & Sameer Singh

Midterm Logistics

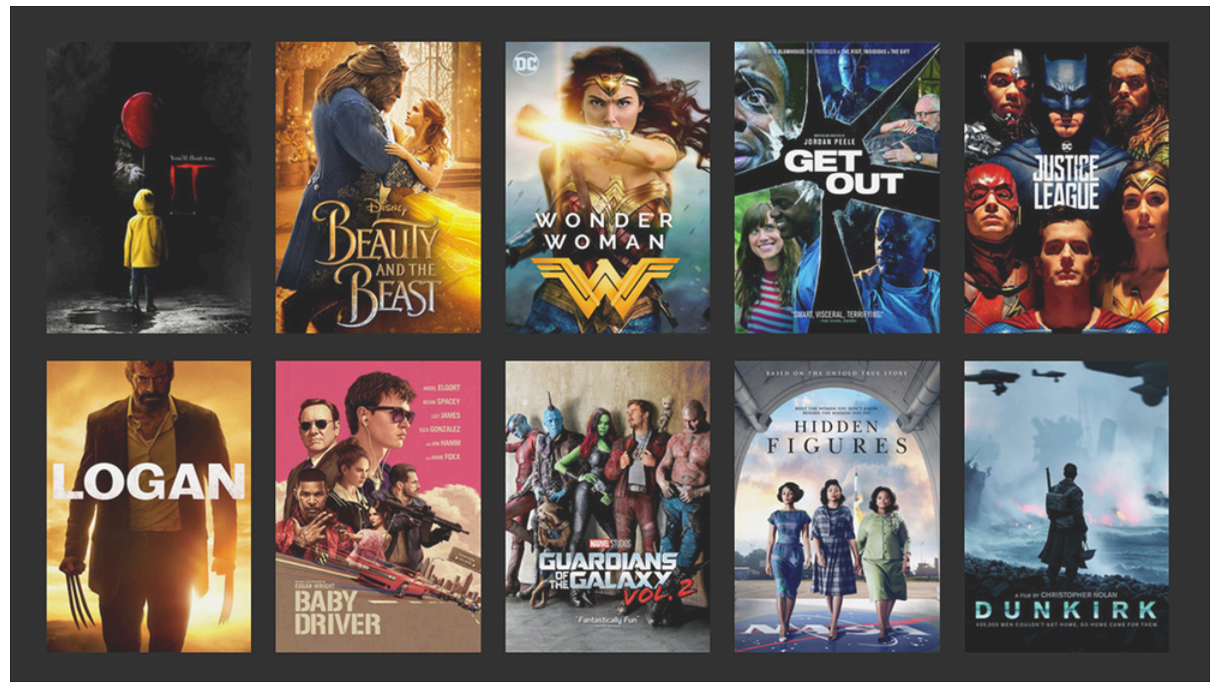
- Format:
  - Time: Thursday, November 4, 11am–12:20
  - Location: SH 128 (in person)
  - Should be doable in 1 hour
- You can use:
  - Self-prepared A4 / Letter-size two-sided single page with anything you'd like on it
  - A basic arithmetic calculator; no phones, no computers
  - Blank paper sheets for your calculations
  - Brainpower and good vibes

Exam suggestions

- Look at past exams
  - Train yourself by reading some solutions, evaluate yourself on held-out exams
- Organize / join study groups (e.g. on Ed)
- During the exam:
  - Start with questions you find easy
  - Don't get bogged down by exact calculations
  - Leave expressions unsolved and come back to them later
  - Turn in your calculation sheet(s)
    - They won't be graded, but can be used for regrading

Learning settings (1): supervised learning

- How can we learn $f : x \mapsto y$ that achieves good performance $v(x, y)$?
- Supervised learning
  - Data: examples of instances x and good decisions y (labels / targets)
  - Given a training dataset $\mathscr{D}$, find f that agrees with $\mathscr{D}$'s labels on its instances
  - Classification: y is a class in a small set
  - Regression: y is continuous



https://medium.com/@gnabr/machine-learning-c28daf3cf60a

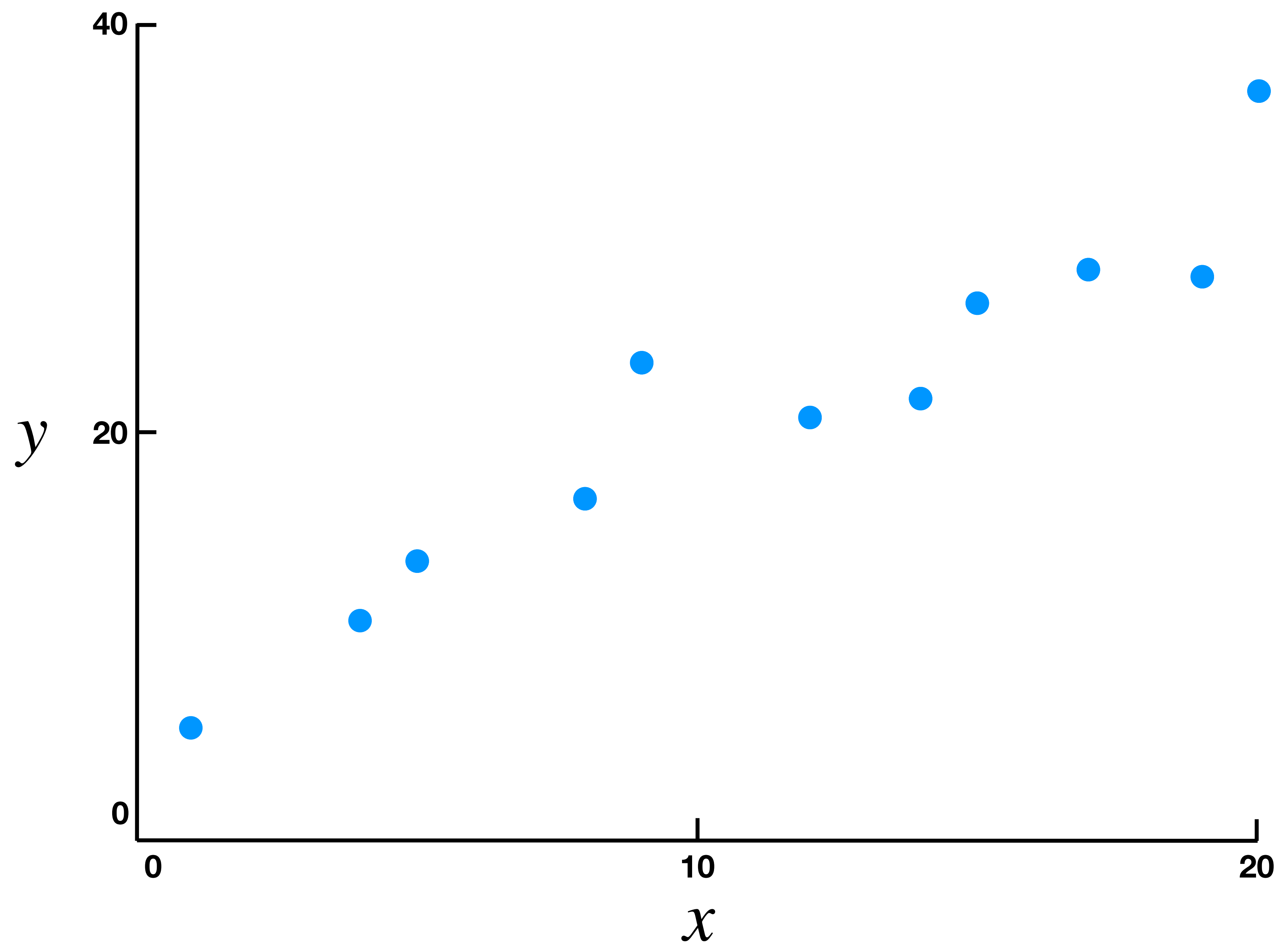
Know thy data

- ML is a data science
  - Look at your data, know what it is, get a "feel" for it
- How many data points?
- What are the features of every data point? What are their data types?
  - Booleans (spam, inbound/outbound, control group)
  - Discrete categories (country/state, protocol, user ID)
  - Integers (1–5 stars, # of bedrooms, year of birth)
  - Reals — up to digital representation (pixel intensity, price, timestamp)
- Is there missing data? Unreasonable values? Surprisingly missing / repeated values?

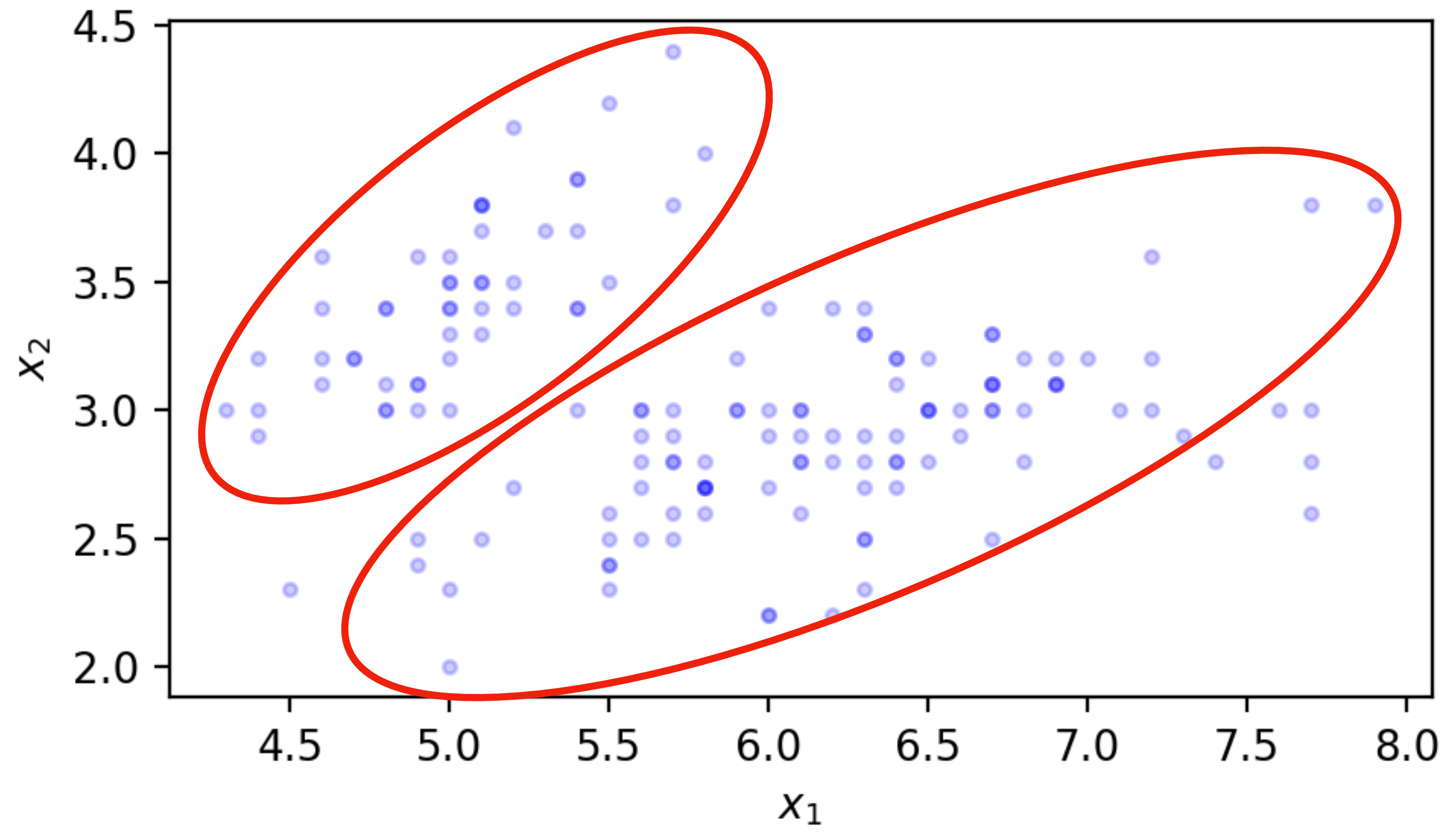
Supervised learning

- Data shows trend
- But also noise

regression

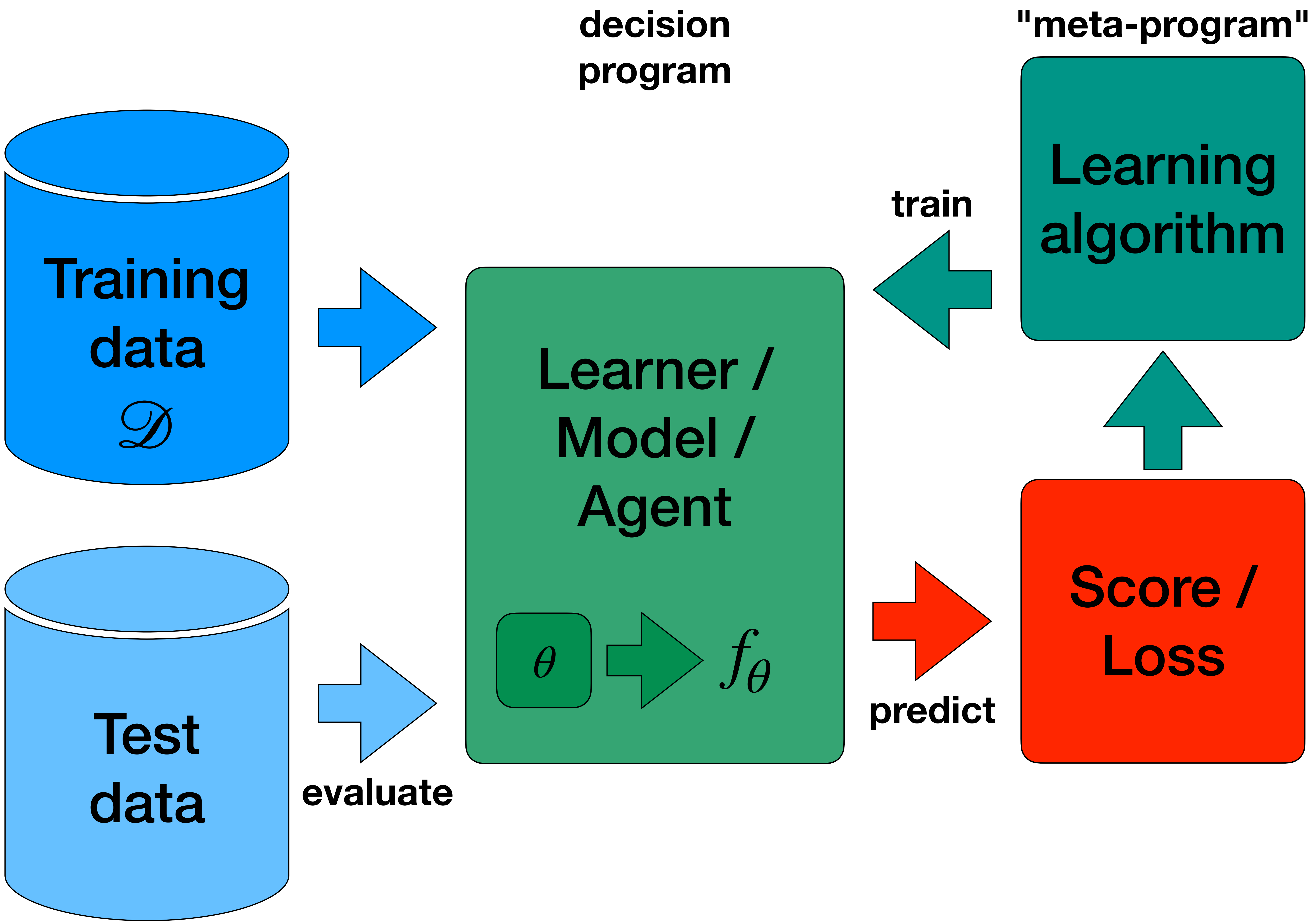


classification



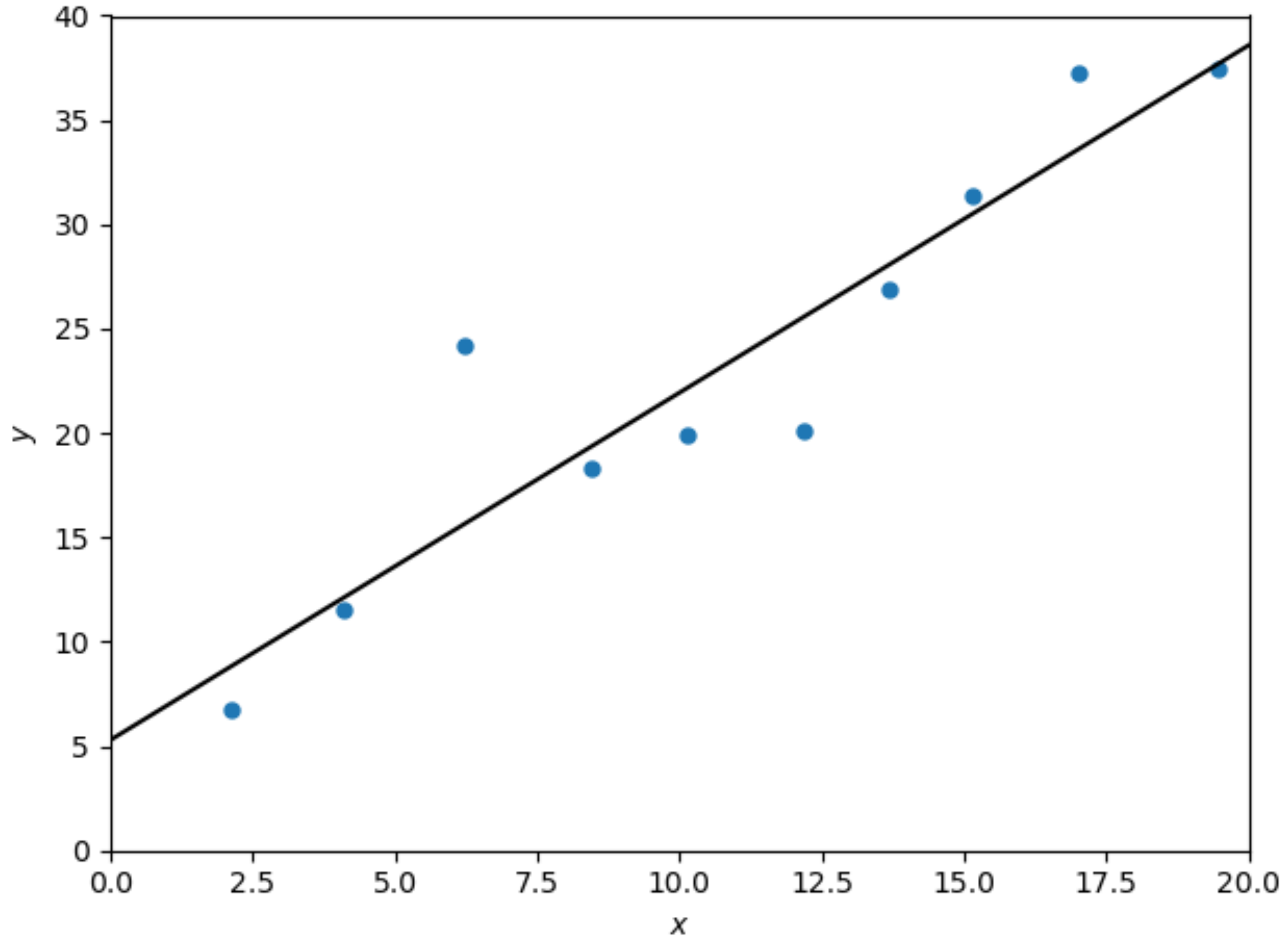
- Given some instance x, what is a good y?

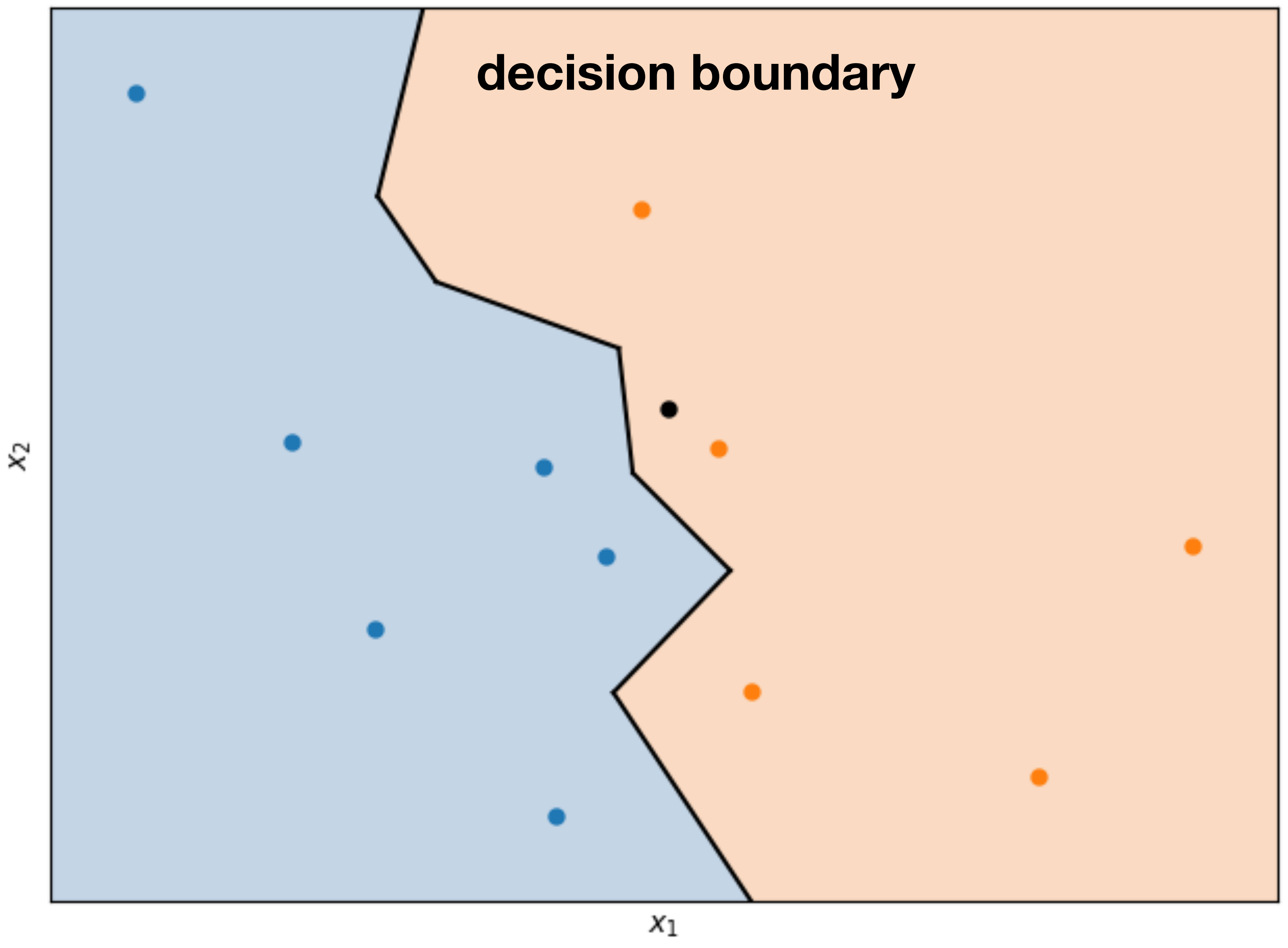
What is machine learning?



Visualizing learned decision function

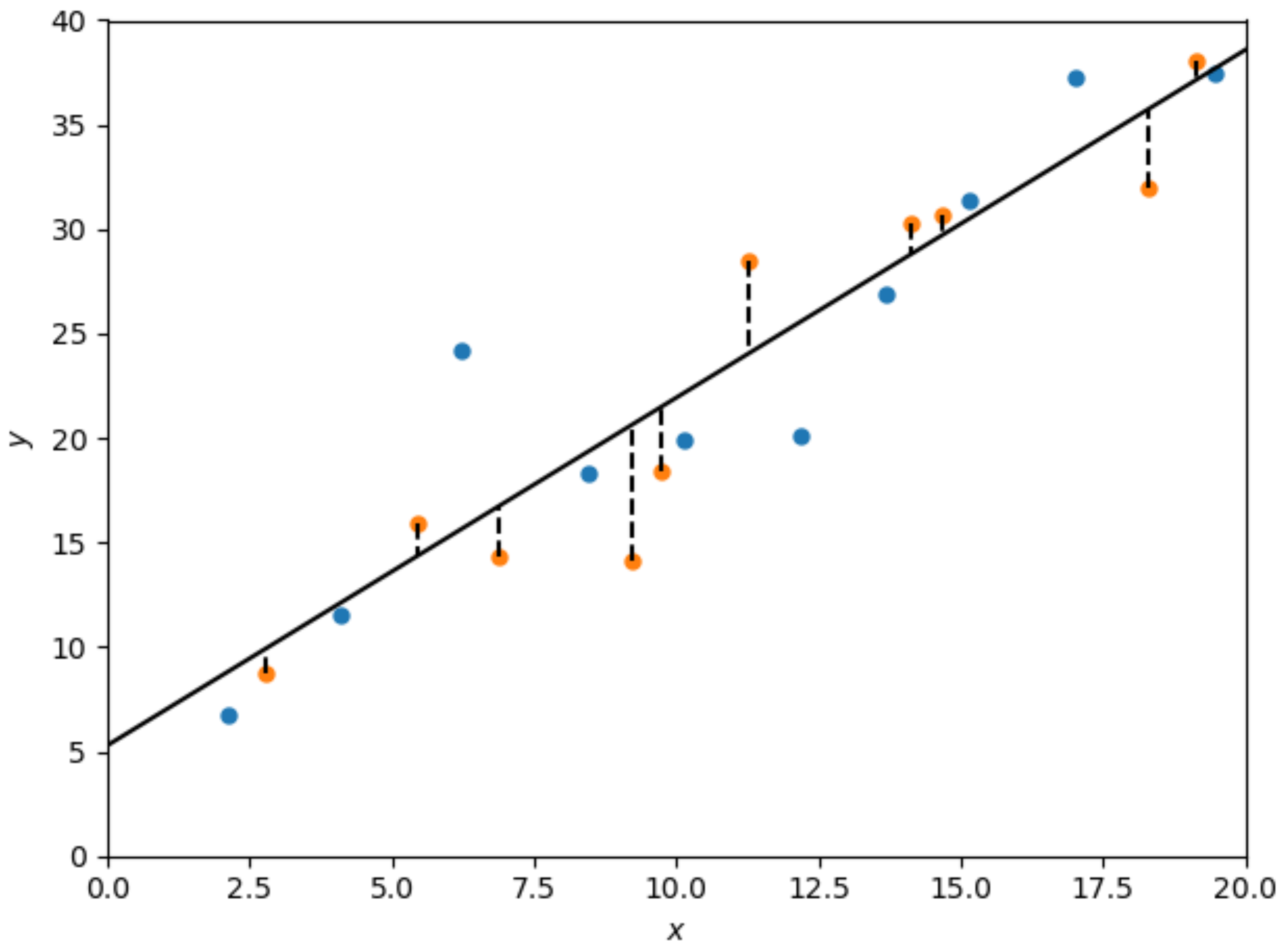
$$f(x) = \theta_0 + \theta_1 x$$

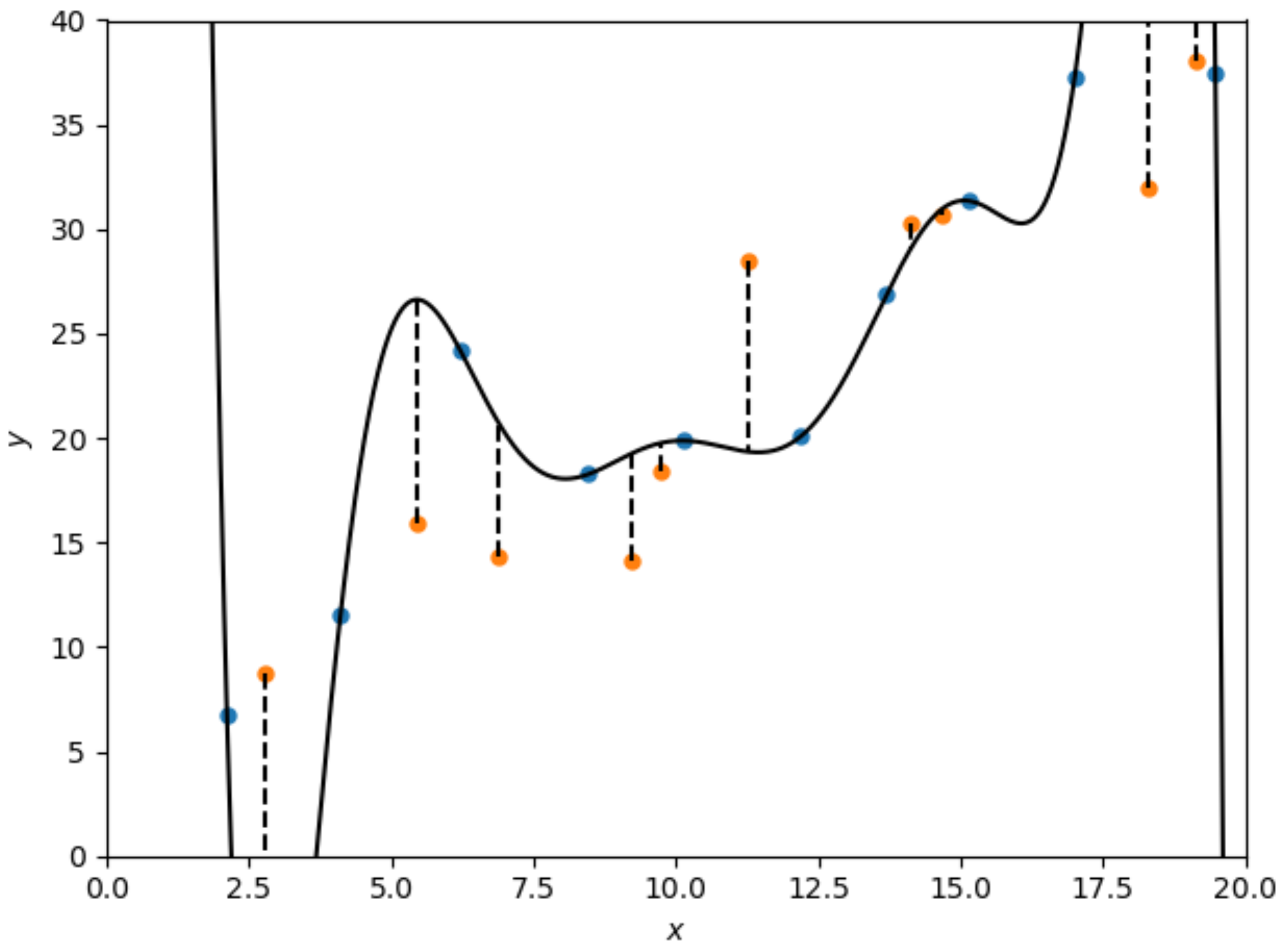




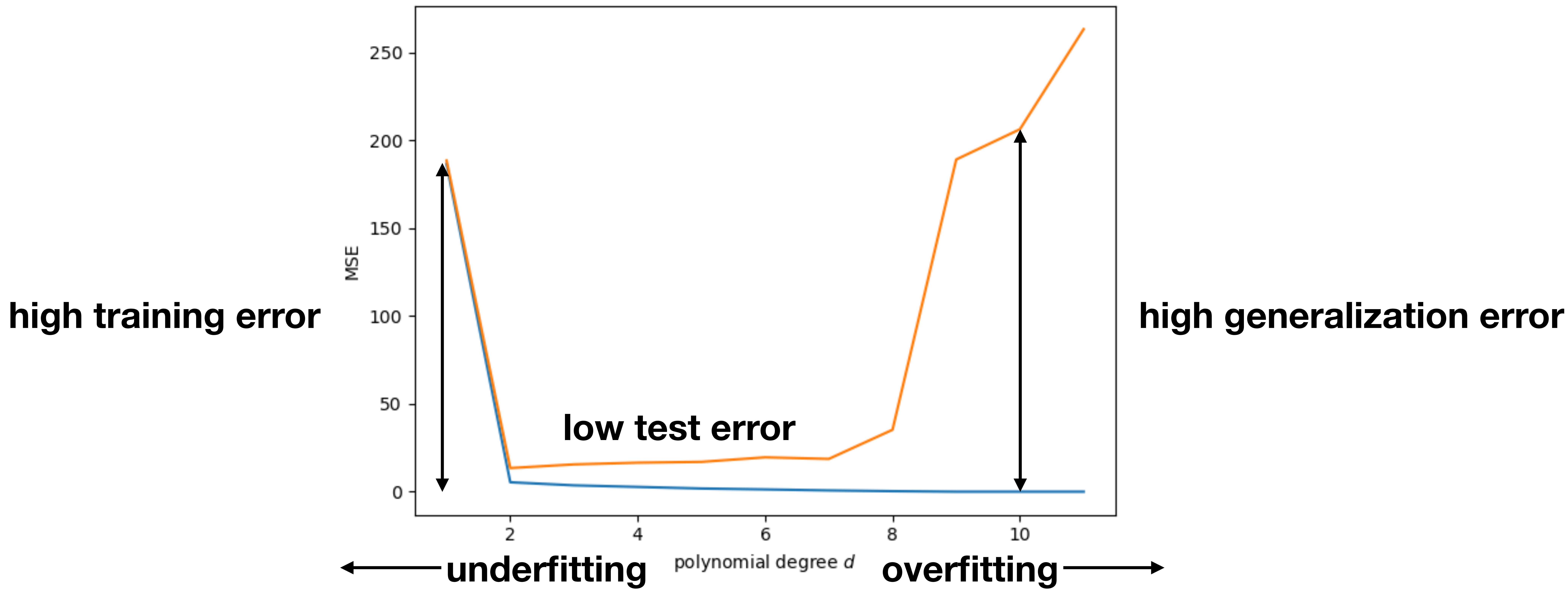
Inductive bias

- Inductive bias = assumptions we make to generalize to data we haven't seen
- Without any assumptions, there is no generalization
  - “Anything is possible” in the test data
- Occam's razor: prefer simpler explanations of the data



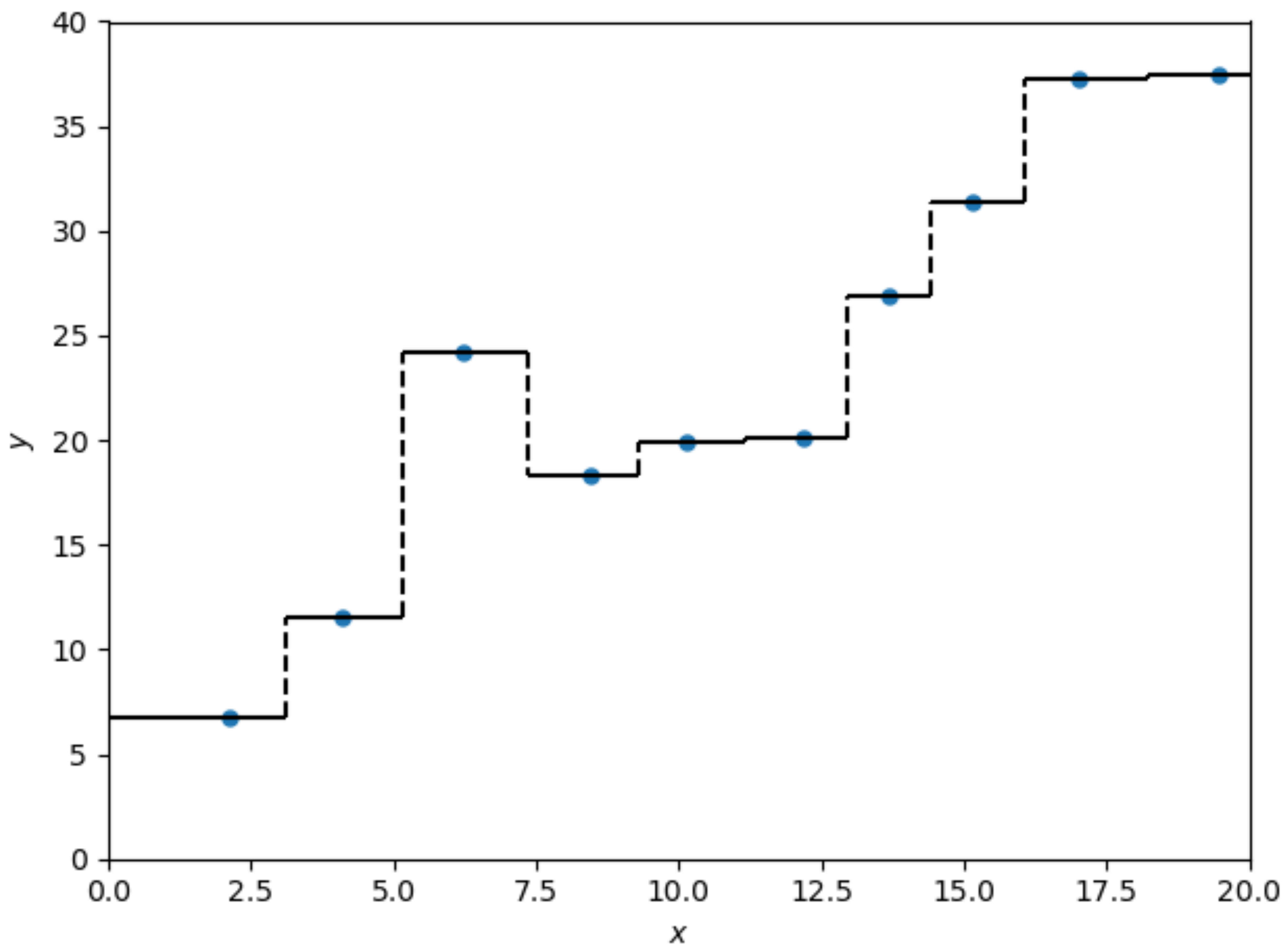


How overfitting affects prediction error



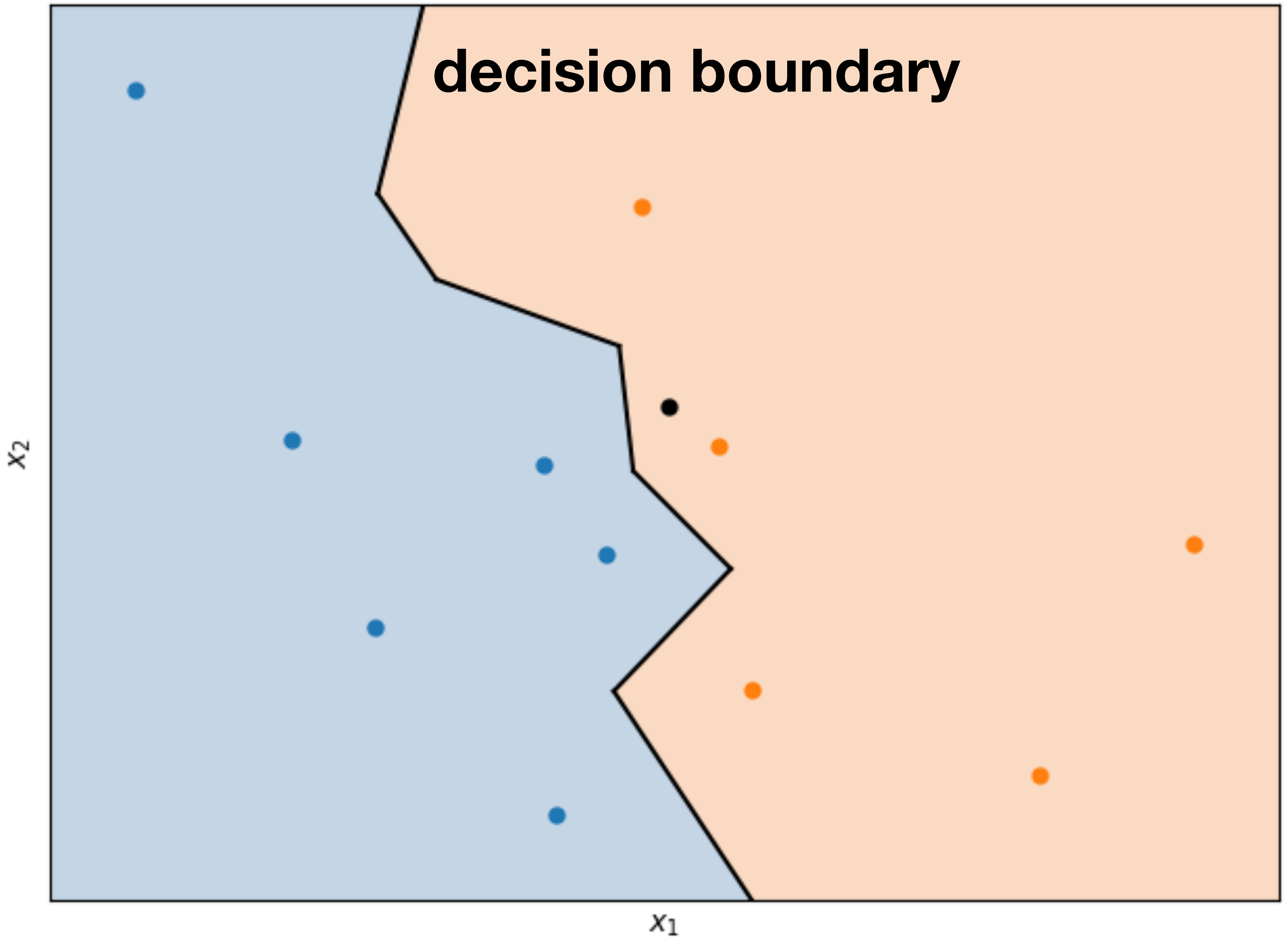
- Low model complexity → underfitting
  - High test error = high training error + low generalization error
- High model complexity → overfitting
  - High test error = low training error + high generalization error

Nearest-Neighbor regression



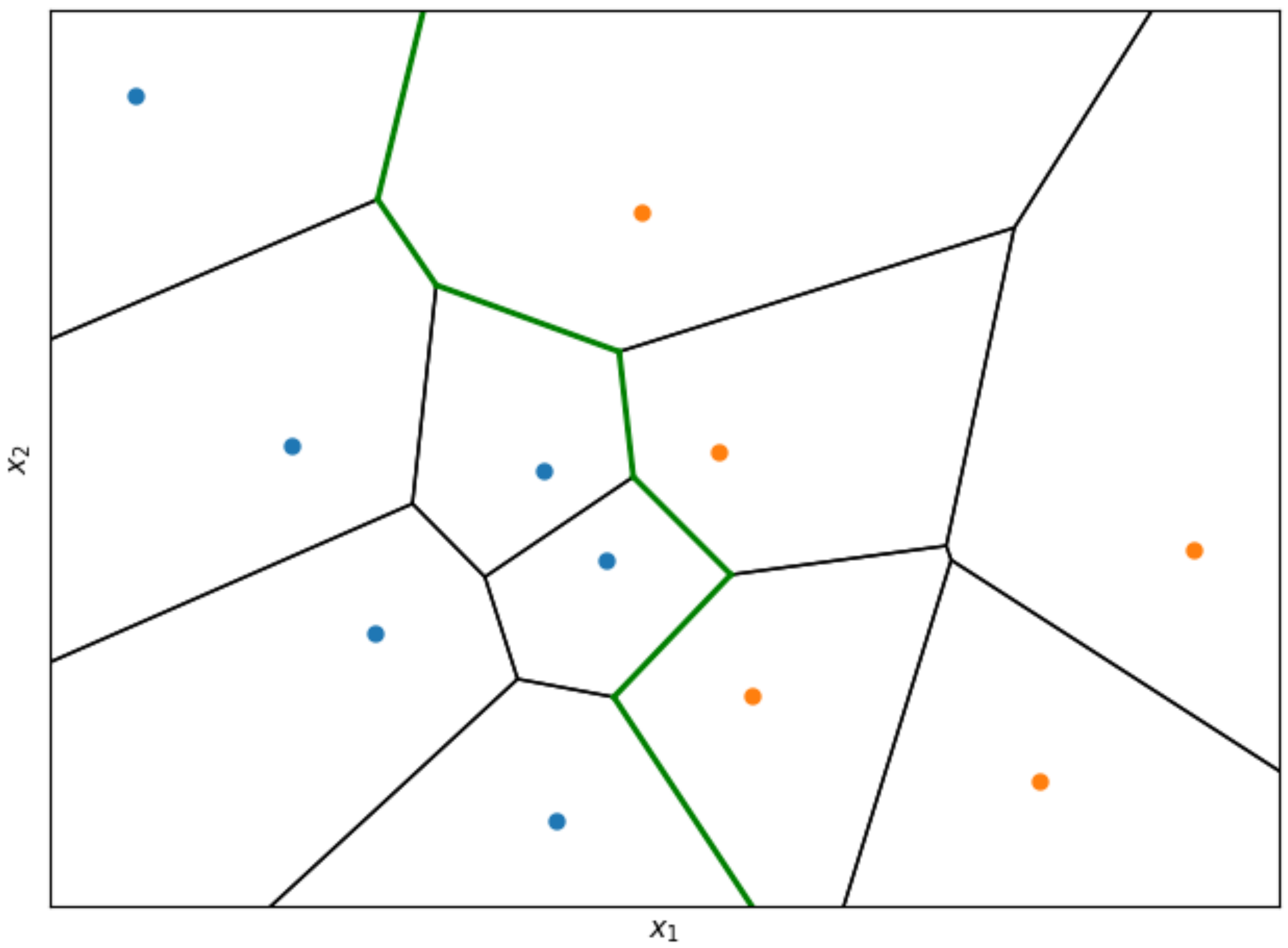
- Decision function $f : x \mapsto y$ is piecewise constant (for 1D x)
- Data induces f implicitly; f is never stored explicitly, but can be computed

Classification



- Using colors as our "third dimension", we can visualize in 2D
- Particularly clear for classification, where y is discrete

Voronoi tessellation

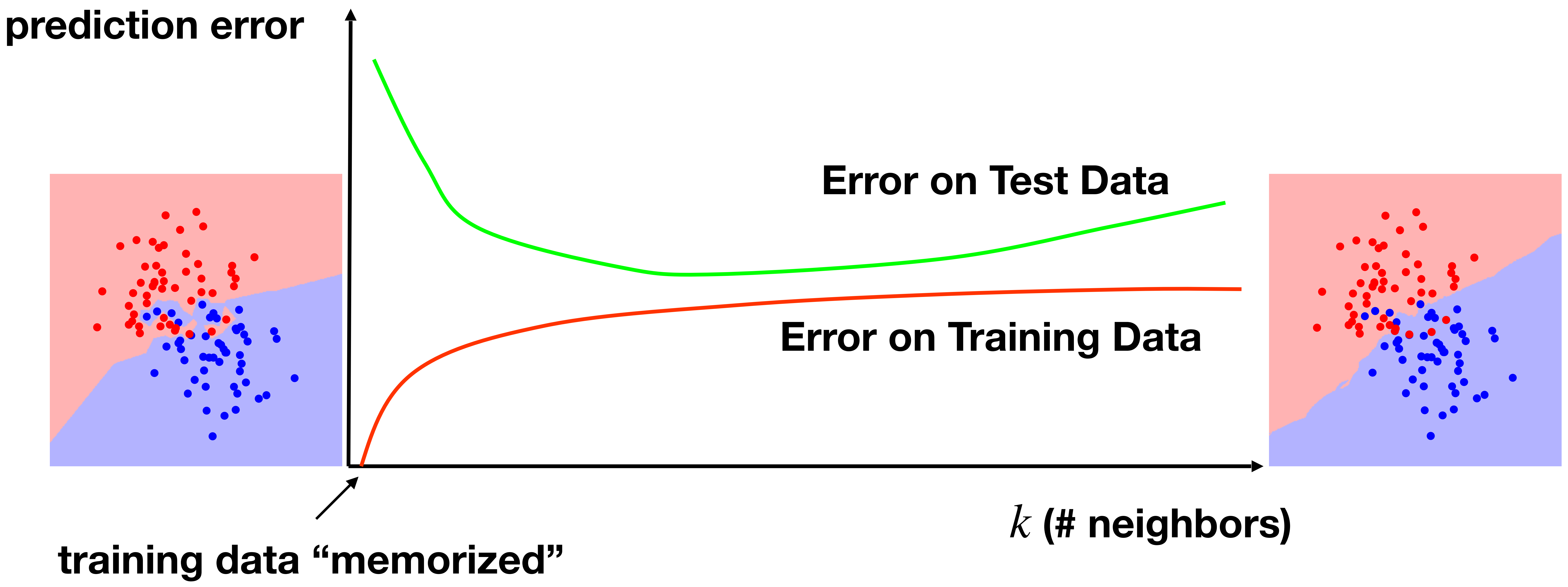


- Each data point has a region in which it is the nearest neighbor
  - This region is a polygon
- The decision boundary consists of the edges that cross classes

k-Nearest Neighbor (kNN)

- Find the k nearest neighbors to x in the dataset
  - Given x, rank the data points by their distance from x, $d(x, x^{(j)})$
    - Usually, Euclidean distance $d(x, x^{(j)}) = \sqrt{\sum_i (x_i - x_i^{(j)})^2}$
  - Select the k data points which are have smallest distance to x
- What is the prediction?
  - Regression: average $y^{(j)}$ for the k closest training examples
  - Classification: take a majority vote among $y^{(j)}$ for the k closest training examples
    - No ties in 2-class problems when k is odd

Error rates and k



- A complex model fits training data but generalizes poorly
- $k = 1$: perfect memorization of examples = complex
- $k = m$: predict majority class over entire dataset = simple
- We can select k with validation

Probabilistic modeling of data

- Assume data with features x and discrete labels y
- Prior probability of each class: $p(y)$
  - Prior = before seeing the features
  - E.g., fraction of applicants that have good credit
- Distribution of features given the class: $p(x \mid y = c)$
  - How likely are we to see x in applicants with good credit?
- Joint distribution: $p(x, y) = p(x)p(y \mid x) = p(y)p(x \mid y)$
- Bayes' rule: posterior $p(y \mid x) = \dfrac{p(y)p(x \mid y)}{p(x)} = \dfrac{p(y)p(x \mid y)}{\sum_c p(y = c)p(x \mid y = c)}$

models:

$x \longrightarrow y$

$y \longrightarrow x$

does not imply causality!

Bayes classifiers

- Learn a "class-conditional" model for the data
  - Estimate the probability for each class $p(y = c)$
  - Split training data by class $\mathcal{D}_c = \{x^{(j)} : y^{(j)} = c\}$
  - Estimate from $\mathcal{D}_c$ the conditional distribution $p(x \,|\, y = c)$
- For discrete x, can represent as a contingency table

Features	# bad	# good
X=0	42	15
X=1	338	287
X=2	3	5
p(y)	**383/690**	**307/690**

p(x|y=0)	p(x|y=1)
42/383	15/307
338/383	287/307
3/383	5/307

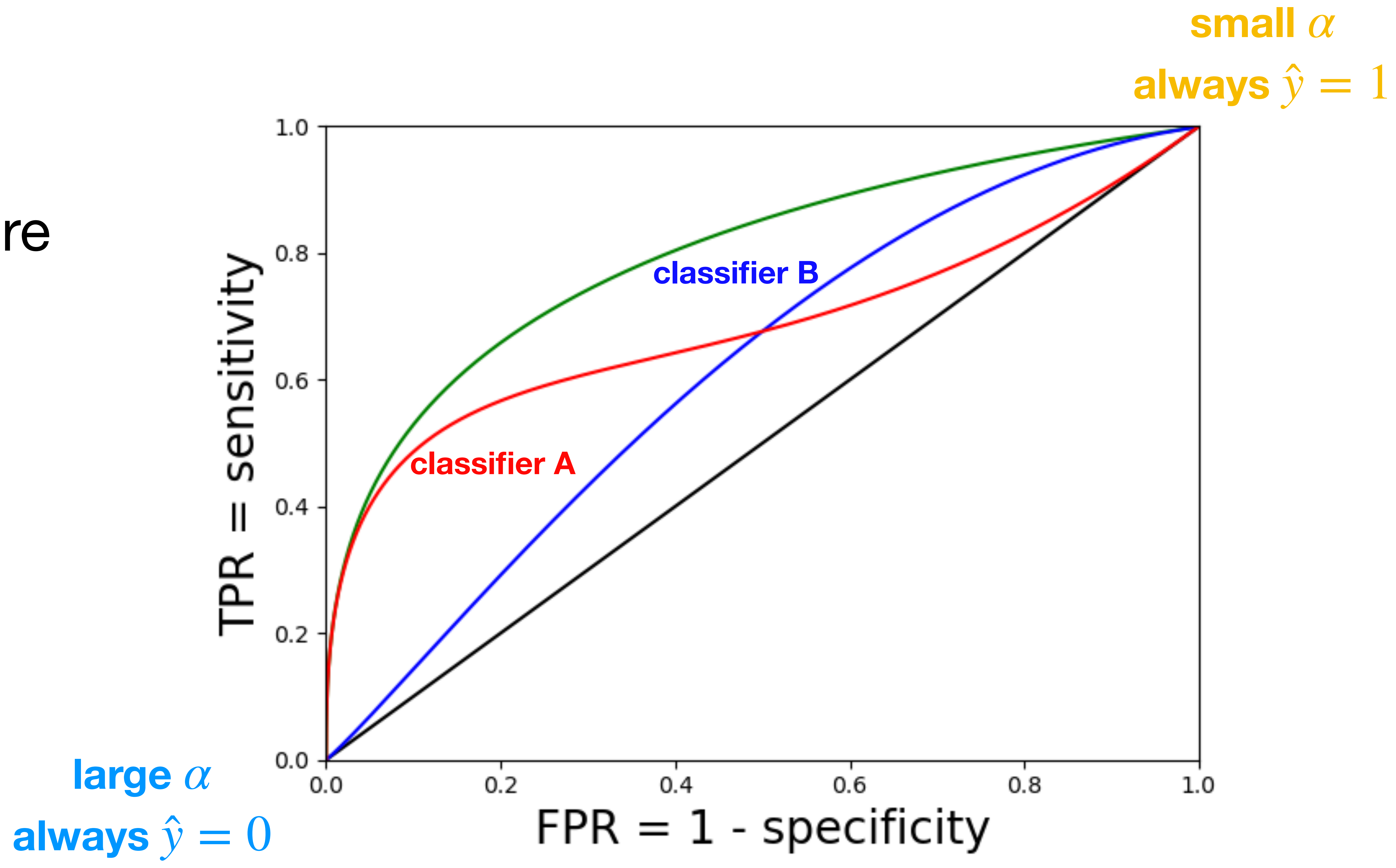
p(y=0|x)	p(y=1|x)
.7368	.2632
.5408	.4592
.3750	.6250

Bayes-optimal decision

- Maximum posterior decision: $\hat{p}(y=0 \,|\, x) \lessgtr \hat{p}(y=1 \,|\, x)$
  - Optimal for the error-rate (0–1) loss: $\mathbb{E}_{x,y\sim p}[\hat{y}(x) \neq y]$
- What if we have different cost for different errors? α_{FP}, α_{FN}
  - $\mathscr{L} = \mathbb{E}_{x,y\sim p}[\alpha_{\text{FP}} \cdot \#(y=0, \hat{y}(x)=1) + \alpha_{\text{FN}} \cdot \#(y=1, \hat{y}(x)=0)]$
- Bayes-optimal decision: $\alpha_{\text{FP}} \cdot \hat{p}(y=0 \,|\, x) \lessgtr \alpha_{\text{FN}} \cdot \hat{p}(y=1 \,|\, x)$
  - Log probability ratio: $\log \frac{\hat{p}(y=1 \,|\, x)}{\hat{p}(y=0 \,|\, x)} \lessgtr \log \frac{\alpha_{\text{FP}}}{\alpha_{\text{FN}}} = \alpha$

Comparing classifiers

- Which classifier (A or B) performs "better"?
    - A is better for high specificity
    - B is better for high sensitivity
    - Need single performance measure
- Area Under Curve (AUC)
    - 0.5 ≤ AUC ≤ 1
    - AUC = 0.5: random guess
    - AUC = 1: no errors



Estimating joint distributions

- Can we estimate $p(x \mid y)$ from data?
- Count how many data points for each x?
  - If $m \ll 2^n$, most instances never occur
    - Do we predict that missing instances are impossible?
      - What if they occur in test data?
- Difficulty to represent and estimate go hand in hand
  - Model complexity → overfitting!

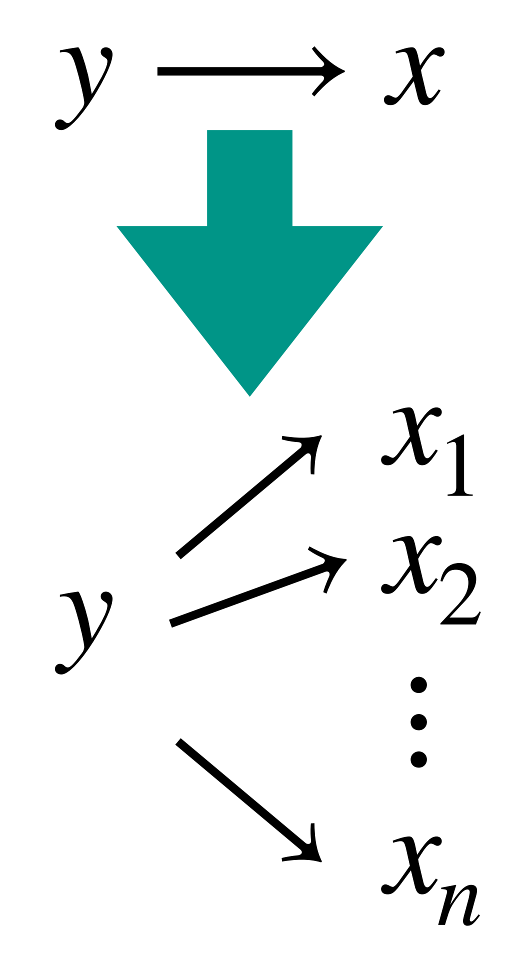
A	B	C	p(A,B,C | y=1)
0	0	0	4/10
0	0	1	1/10
0	1	0	0/10
0	1	1	0/10
1	0	0	1/10
1	0	1	2/10
1	1	0	1/10
1	1	1	1/10

Regularization

- Reduce effective size of model class
  - Hope to avoid overfitting
- One way: make the model more "regular", less sensitive to data quirks
- Example: add small "pseudo-count" to the counts (before normalizing)
  - $\hat{p}(x \mid y = c) = \dfrac{\#_c(x) + \alpha}{m_c + \alpha \cdot 2^n}$
  - Not a huge help here, most cells will be uninformative $\dfrac{\alpha}{m_c + \alpha \cdot 2^n}$

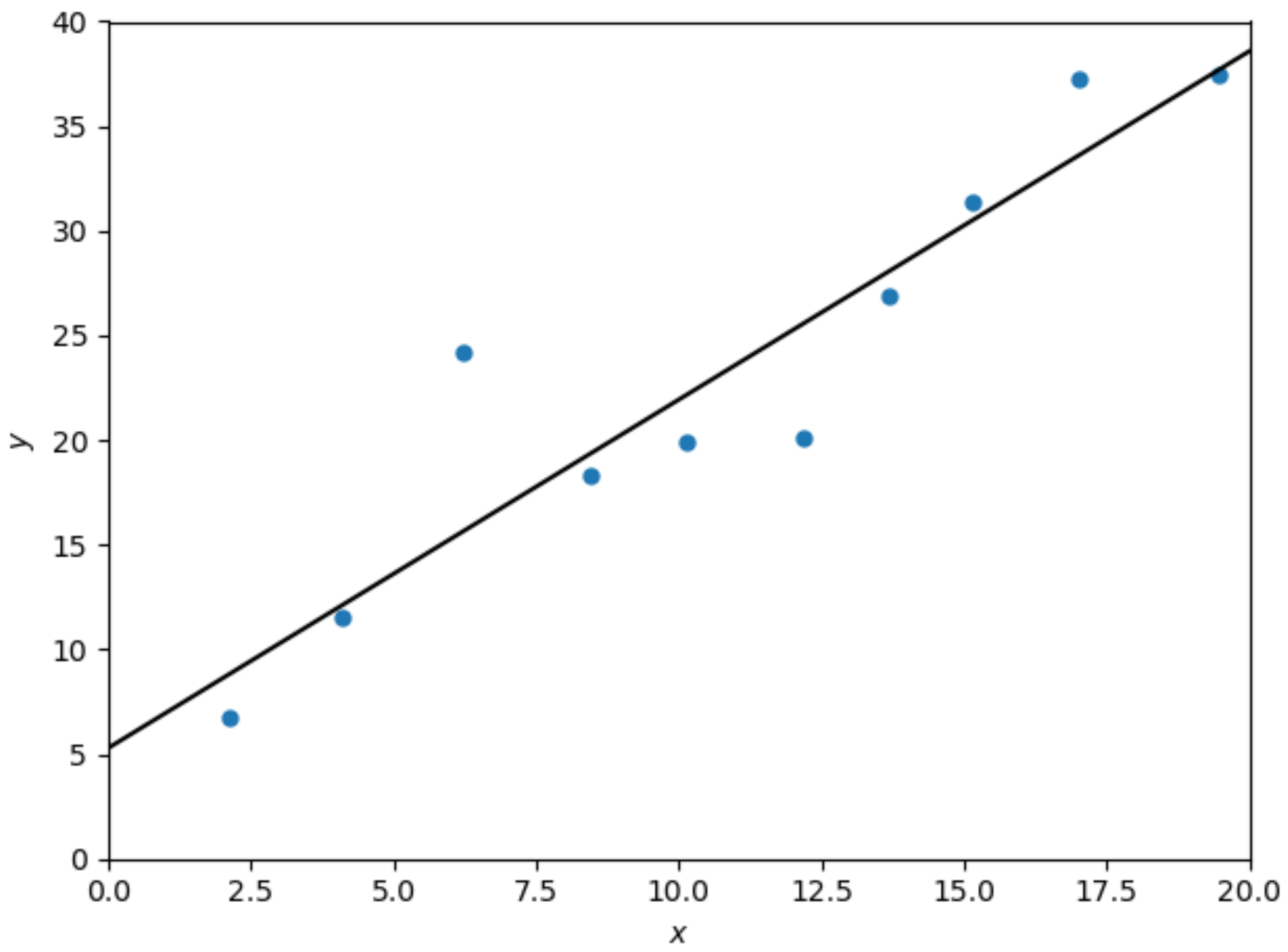
Naïve Bayes models

- We want to predict some value y, e.g. auto accident next year
- We have many known indicators for y (covariates) $x = x_1, \ldots, x_n$
  - E.g., age, income, education, zip code, ...
  - Learn $p(y \mid x_1, \ldots, x_n)$ — but cannot represent / estimate $O(2^n)$ values
- Naïve Bayes
  - Estimate prior distribution $\hat{p}(y)$
  - Assume $p(x_1, \ldots, x_n \mid y) = \prod_i p(x_i \mid y)$, estimate covariates independently $\hat{p}(x_i \mid y)$
  - Model: $\hat{p}(y \mid x) \propto \hat{p}(y) \prod_i \hat{p}(x_i \mid y)$



causal structure wrong!
(but useful...)

Linear regression

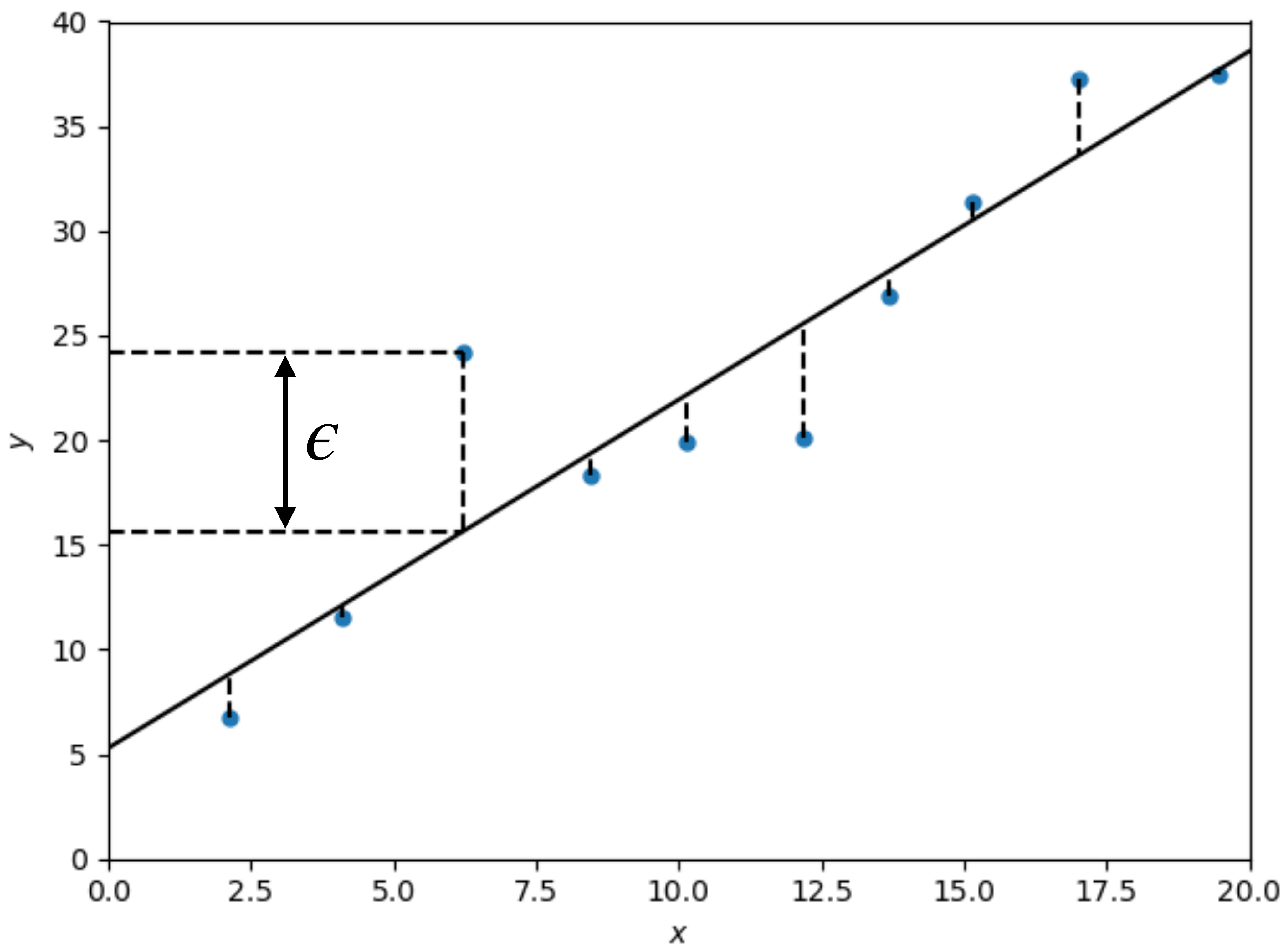


- Decision function $f : x \mapsto y$ is linear, $f(x) = \theta_0 + \theta_1 x$
- f is stored by its parameters $\theta = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix}$

Measuring error

observation y

prediction
$\hat{y} = f(x)$

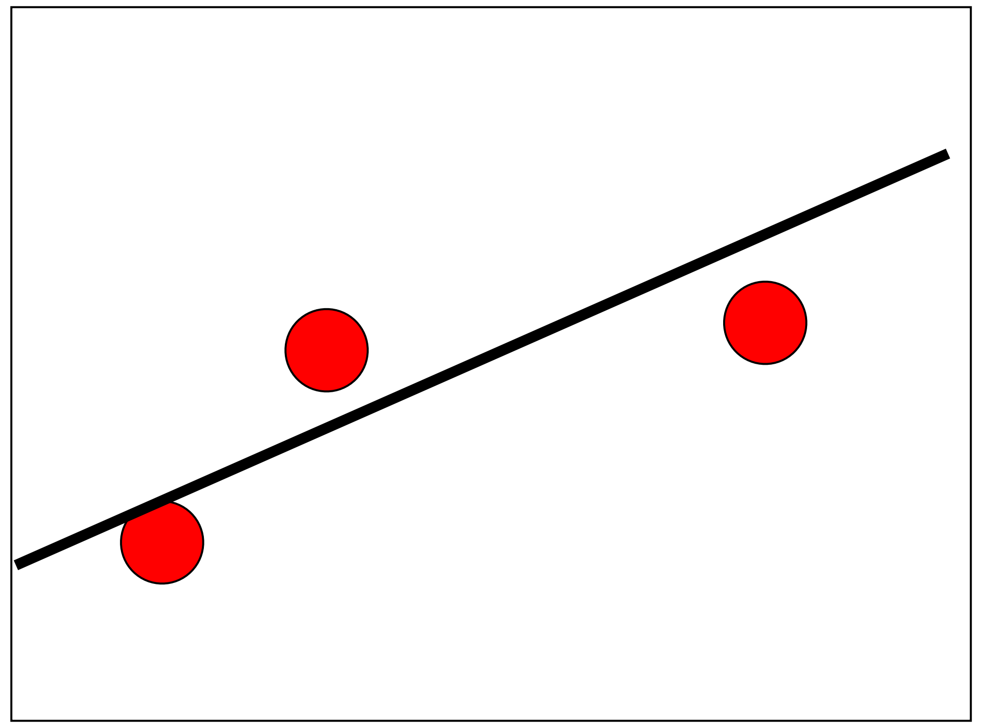


- Error / residual: $\epsilon = y - \hat{y}$

- Mean square error (MSE): $\frac{1}{m}\sum_j (\epsilon^{(j)})^2 = \frac{1}{m}\sum_j (y^{(j)} - \hat{y}^{(j)})^2$

Least Squares

- The minimum is achieved when the gradient is 0



$$\nabla_\theta \mathscr{L}_\theta = -\frac{2}{m}(y - \theta^\intercal X)X^\intercal = 0$$

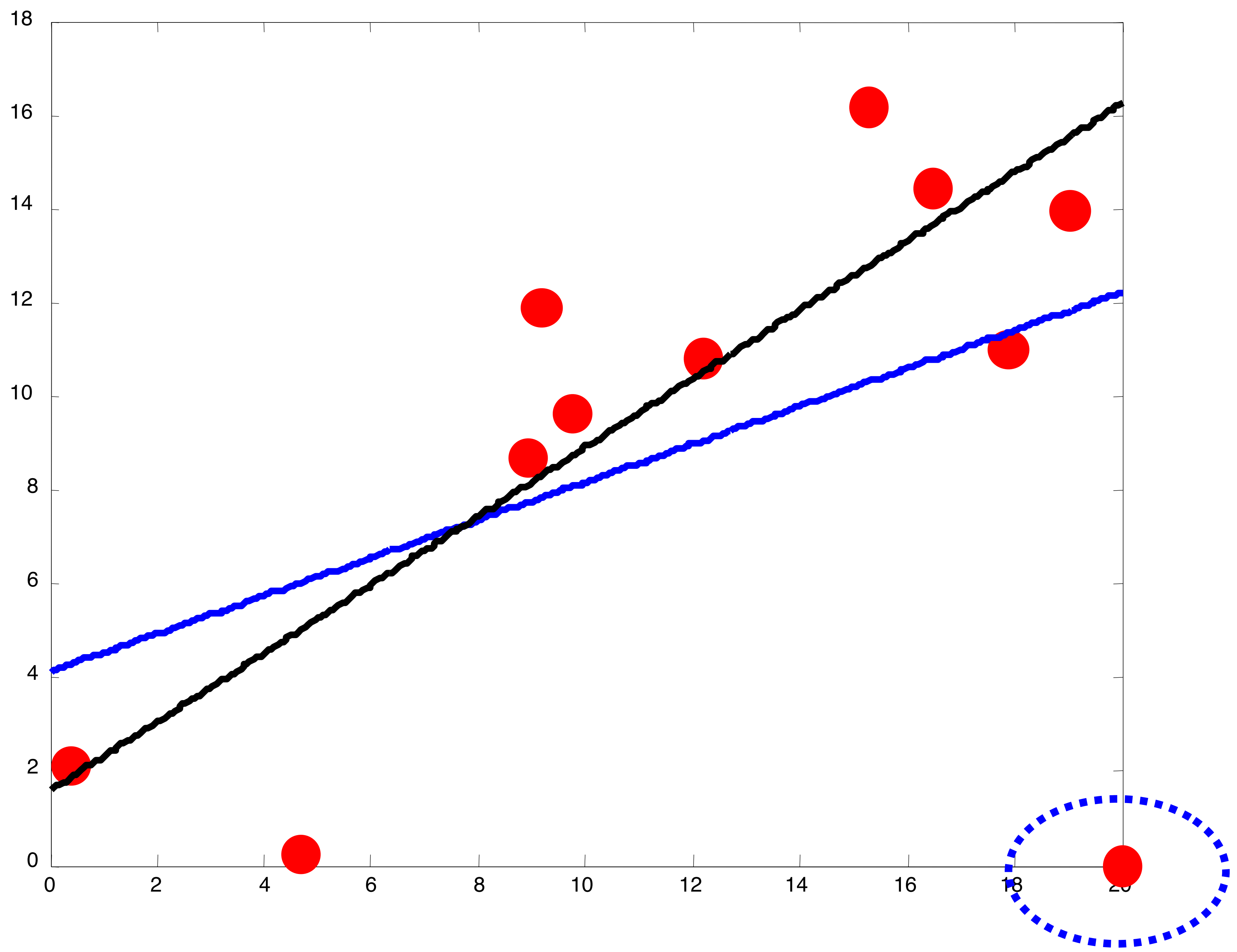
$$\theta^\intercal X X^\intercal = y X^\intercal$$

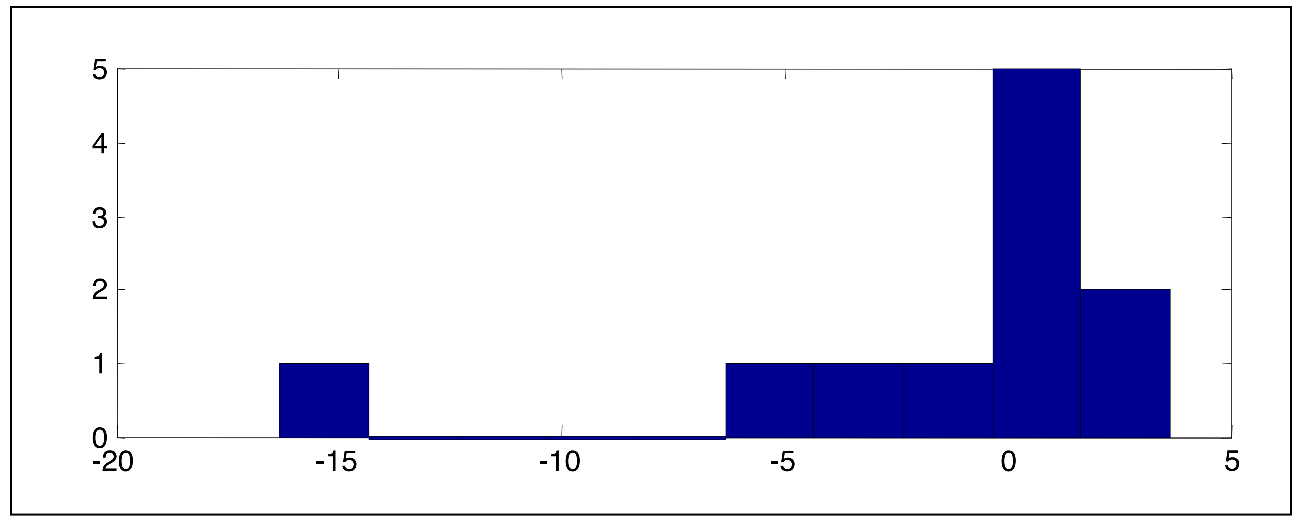
$$\theta^\intercal = y X^\intercal (X X^\intercal)^{-1}$$

- $XX^\intercal$ is invertible when X has linearly independent rows = features
- $X^\dagger = X^\intercal (XX^\intercal)^{-1}$ is the Moore-Penrose pseudo-inverse of X
  - $X^\dagger = X^{-1}$ when the inverse exists
  - Can define $X^\dagger$ via Singular Value Decomposition (SVD) when $XX^\intercal$ isn't invertible
- $\theta^\intercal = yX^\dagger$ is the Least Squares fit of the data (X, y)

MSE and outliers

- MSE is sensitive to outliers





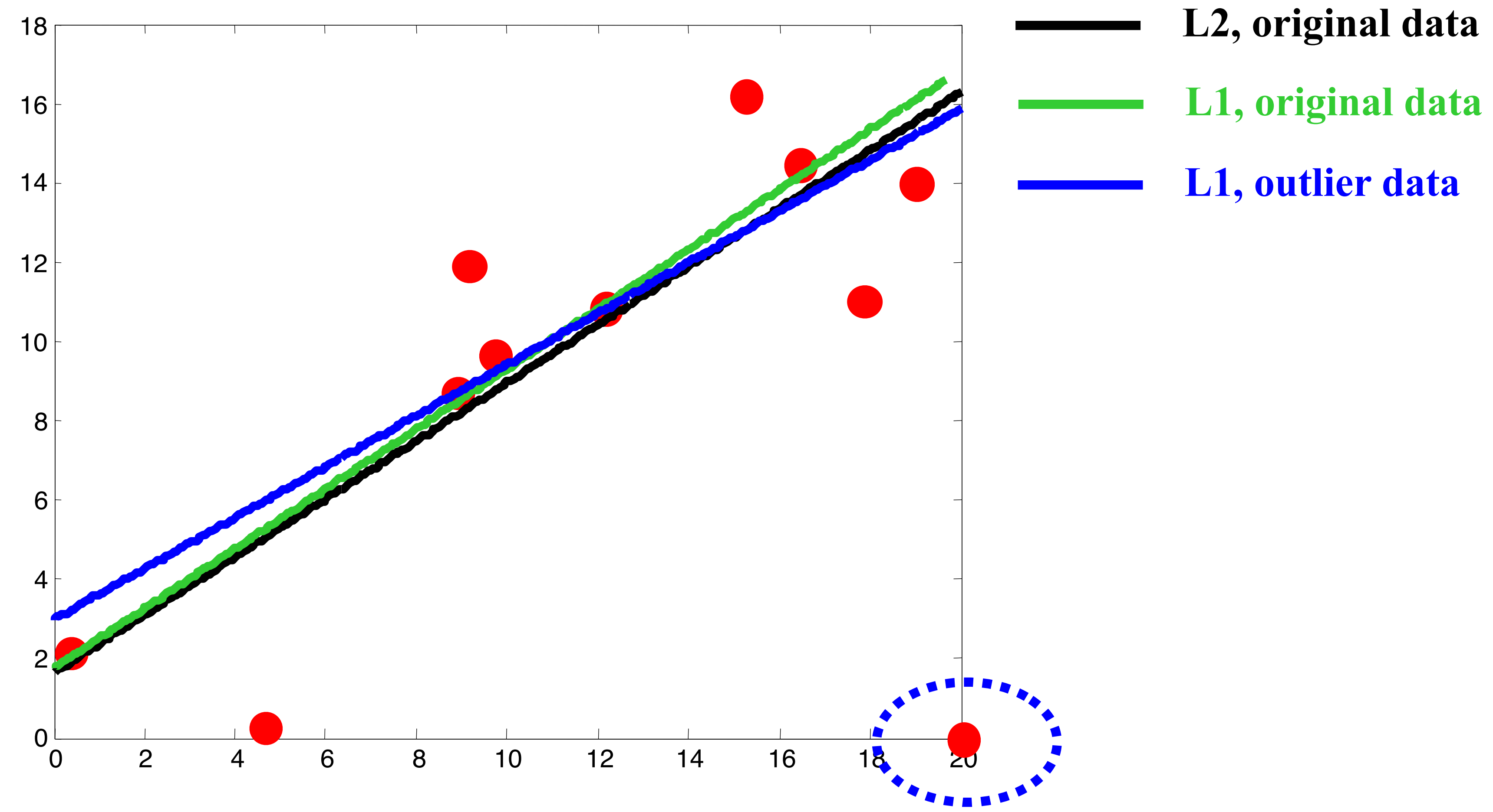
- Square error $\approx 16^2$ throws off entire optimization

Mean Absolute Error (MAE)

- MSE uses the L_2 norm of the error $\|y - \theta^\intercal X\|_2^2 = \sum_j (y - \theta^\intercal X)^2$

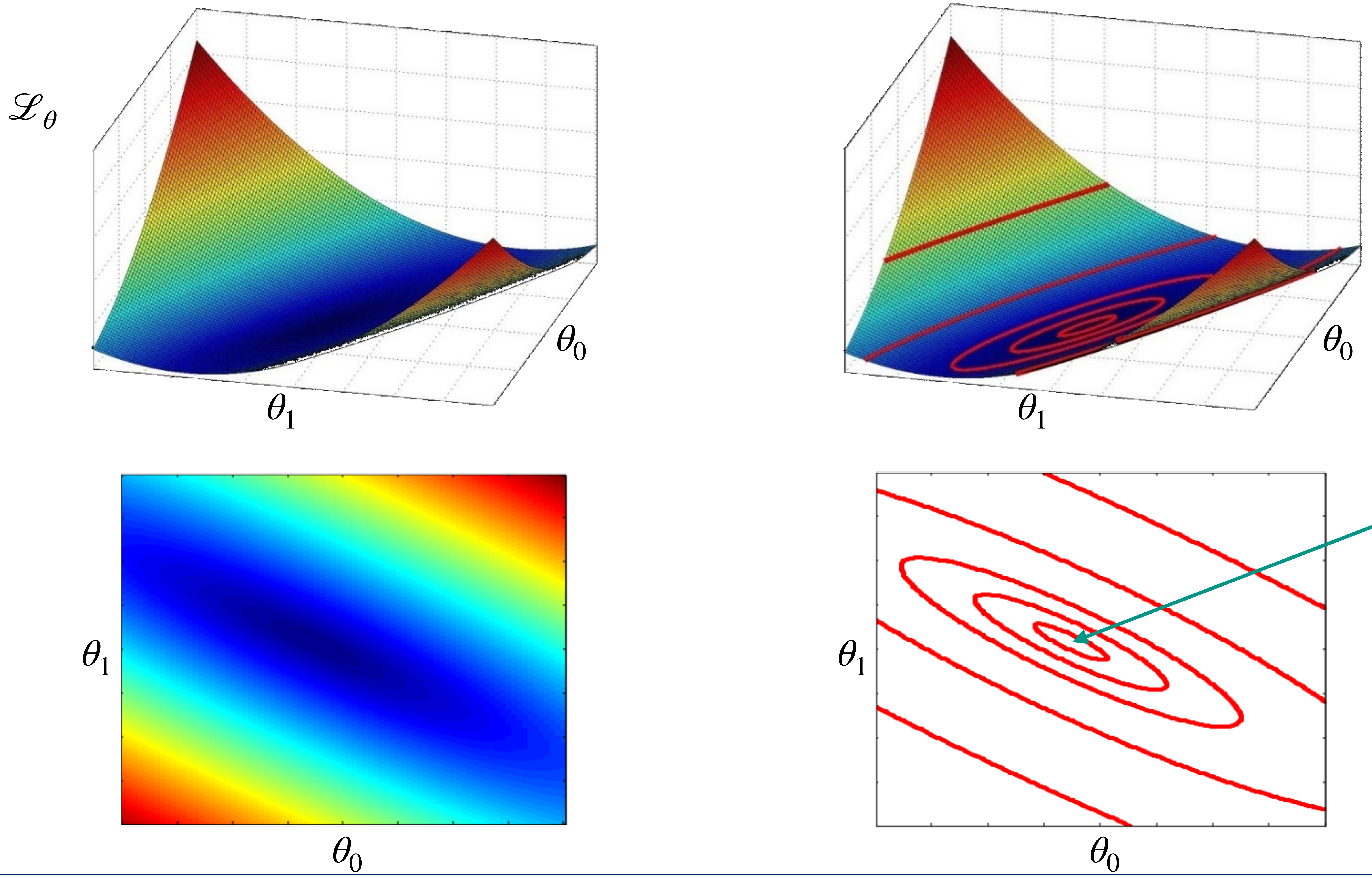
- What if we use the L_1 norm $\|y - \theta^\intercal X\|_1 = \sum_j |y - \theta^\intercal X|$?

  - Mean Absolute Error (MAE): $\frac{1}{m} \sum_j |y - \theta^\intercal X|$



Loss landscape

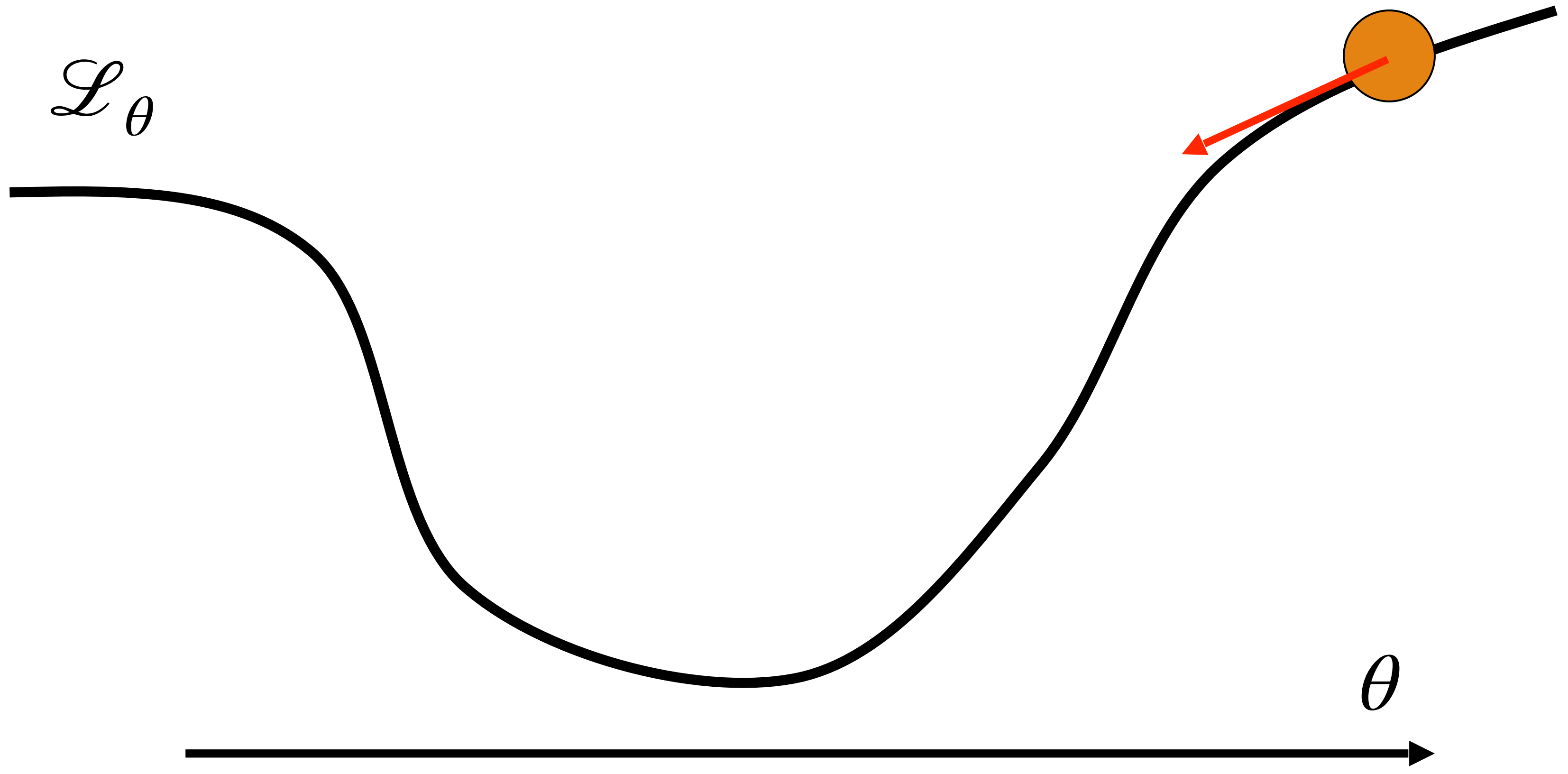
- $\mathscr{L}_\theta(\mathscr{D}) = \frac{1}{m}(y - \theta^\intercal X)(y - \theta^\intercal X)^\intercal = \frac{1}{m}(\theta^\intercal X X^\intercal \theta - 2yX^\intercal\theta + yy^\intercal)$ ← **quadratic!**



minimum loss

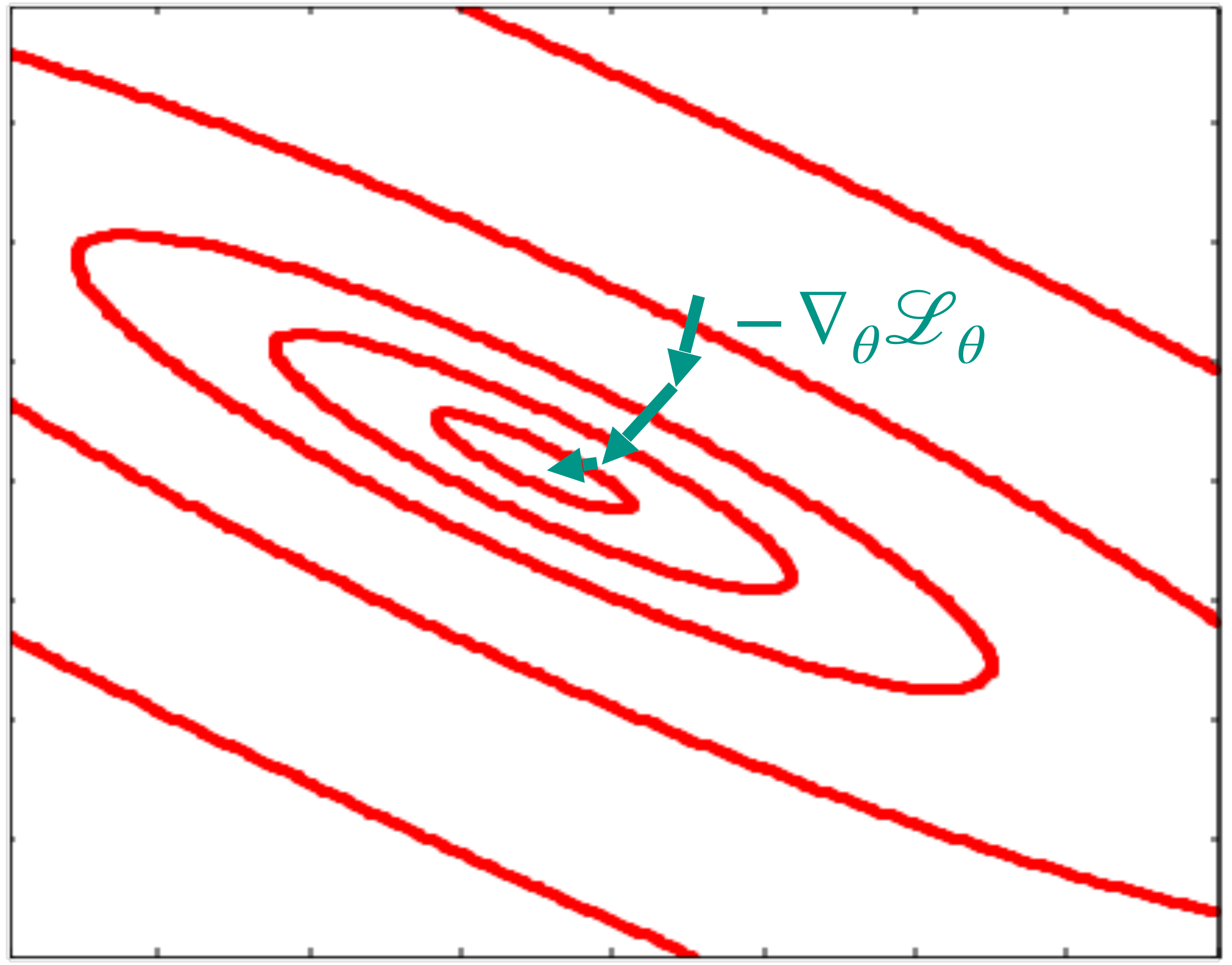
Gradient descent

- How to vary $\theta \in \mathbb{R}^{n+1}$ to improve the loss $\mathscr{L}_\theta$?
  - Find a direction in parameter space in which $\mathscr{L}_\theta$ is decreasing
- Derivative $\partial_\theta \mathscr{L}_\theta = \lim_{\delta\theta \to 0} \frac{\mathscr{L}_{\theta+\delta\theta} - \mathscr{L}_\theta}{\delta\theta}$
  - Positive = loss increases with θ
  - Negative = loss decreases with θ



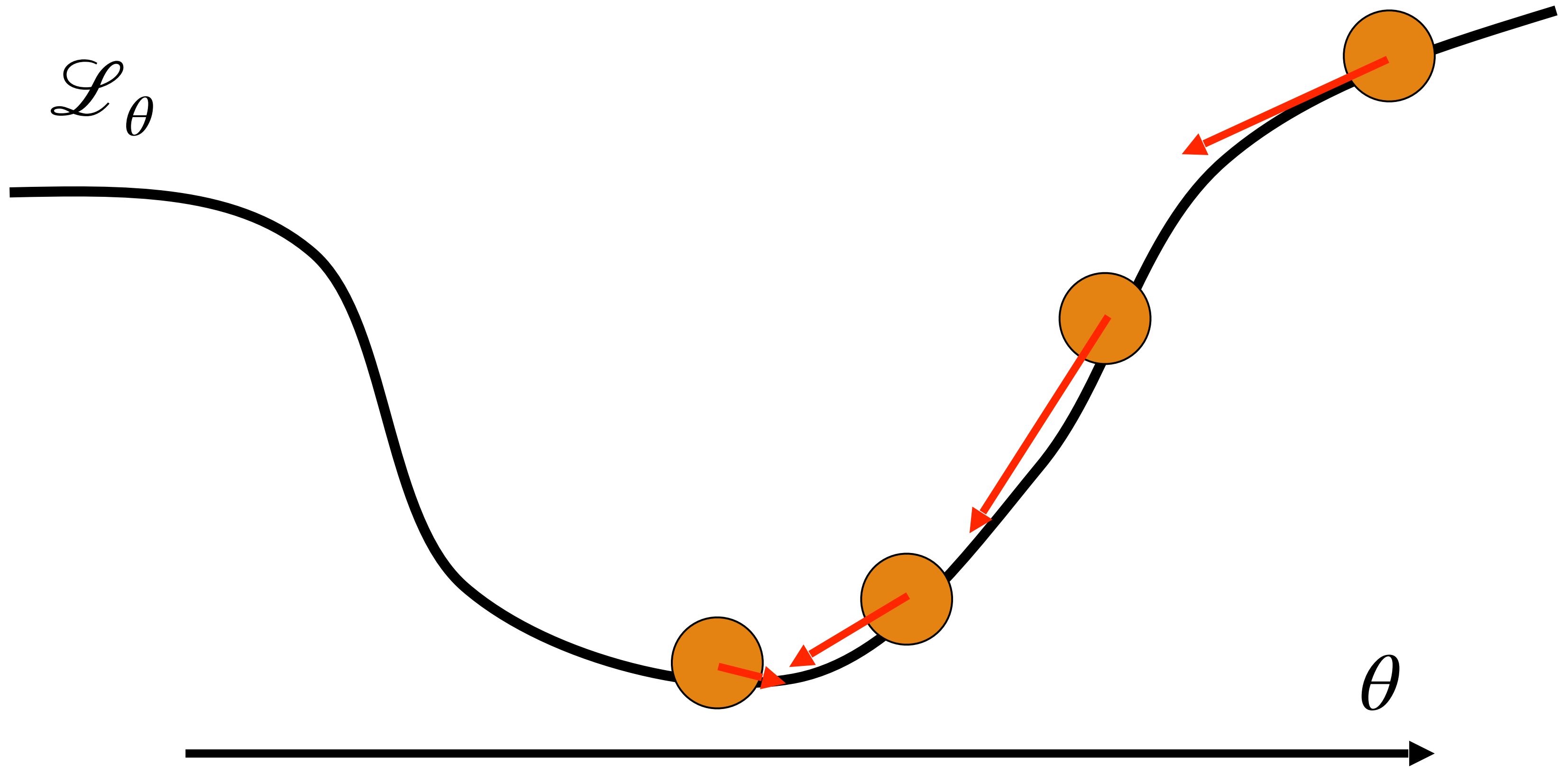
Gradient descent in higher dimension

- Gradient vector: $\nabla_\theta \mathscr{L}_\theta = \begin{bmatrix} \partial_{\theta_0} \mathscr{L}_\theta & \cdots & \partial_{\theta_n} \mathscr{L}_\theta \end{bmatrix}$
- Taylor expansion: $\mathscr{L}(\theta + \delta\theta) = \mathscr{L}(\theta) + (\delta\theta)^\intercal \nabla_\theta \mathscr{L}_\theta + o(\|\delta\theta\|^2)$
  - If we take a small step $\delta\theta$, the best one is in direction $\nabla_\theta \mathscr{L}_\theta$
  - Gradient = direction of steepest ascent (negative = steepest descent)



Gradient Descent

- Initialize θ
- Do
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathscr{L}_\theta$
- While $||\alpha \nabla_\theta \mathscr{L}_\theta|| \leq \epsilon$
- Learning rate: α
  - Can change in each iteration



Stochastic / Online Gradient Descent

- Estimate $\nabla_\theta \mathscr{L}_\theta$ fast on a sample of data points
- For each data point:

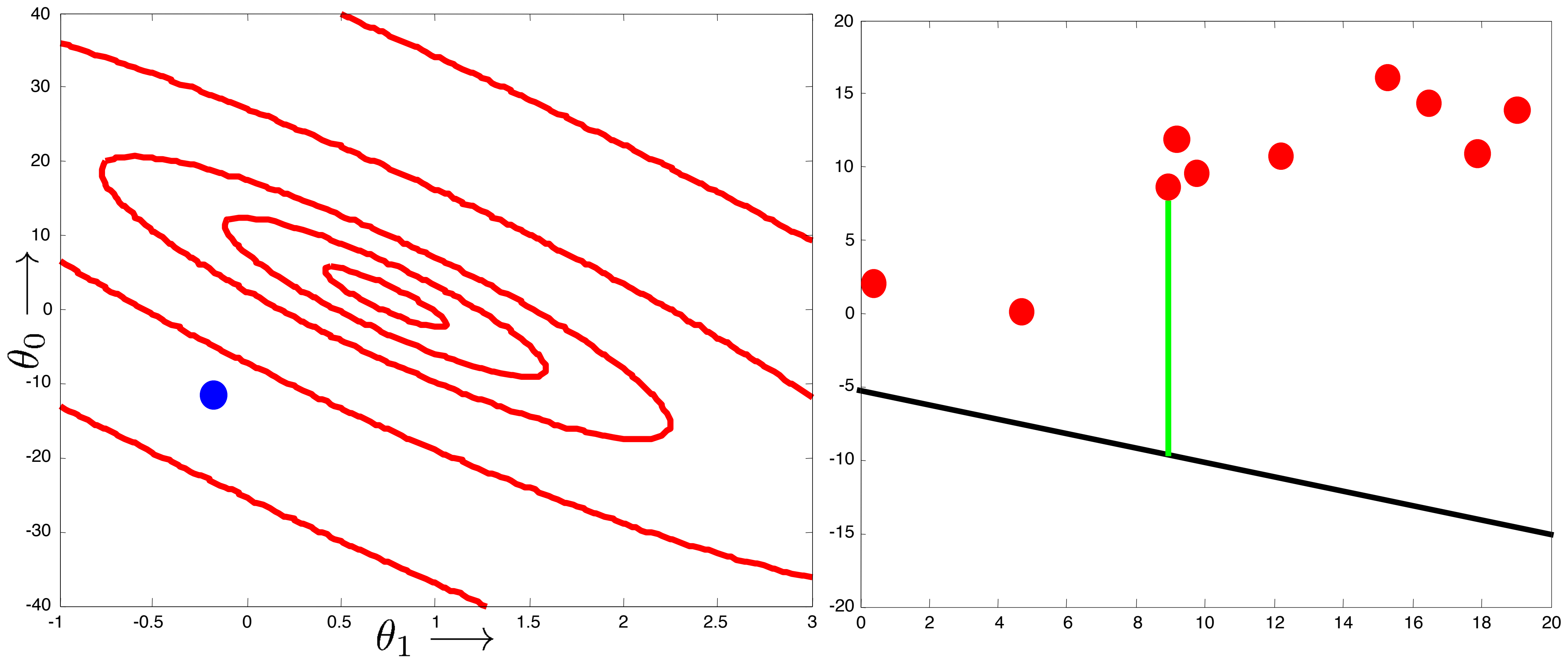
$$\nabla_\theta \mathscr{L}_\theta(x^{(j)}, y^{(j)}) = \nabla_\theta (y^{(j)} - \theta^\intercal x^{(j)})^2 = -2(y^{(j)} - \theta^\intercal x^{(j)})(x^{(j)})^\intercal$$

- This is an unbiased estimator of the gradient, i.e. in expectation

$$\mathbb{E}_{j \sim \text{Uniform}(1,\ldots,m)}[\nabla_\theta \mathscr{L}_\theta^{(j)}] = \frac{1}{m} \sum_j \nabla_\theta \mathscr{L}_\theta^{(j)} = \nabla_\theta \mathscr{L}_\theta(\mathscr{D})$$

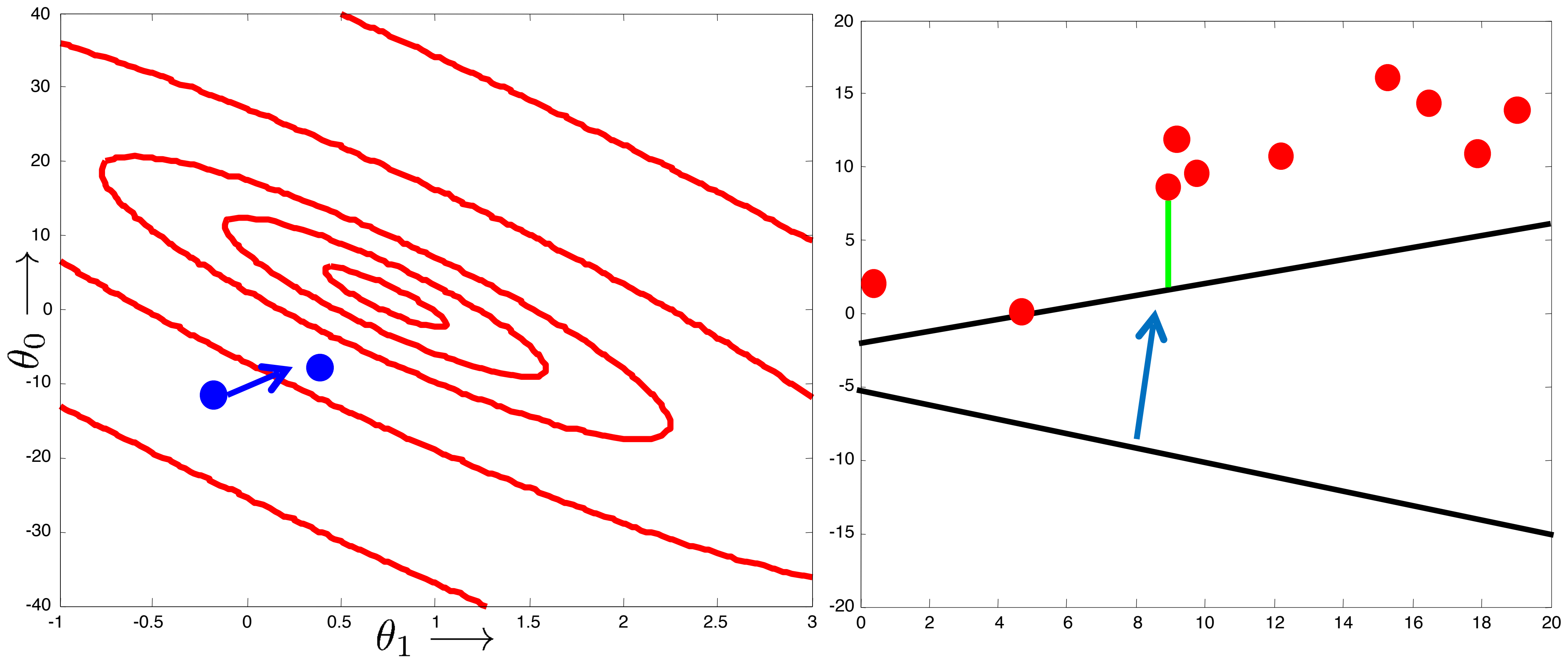
- $\nabla_\theta \mathscr{L}_\theta(\mathscr{D})$ is already a noisy unbiased estimator of true gradient $\mathbb{E}_{x,y \sim p}[\nabla_\theta \mathscr{L}_\theta(x, y)]$
  - SGD is even more noisy

Stochastic Gradient Descent



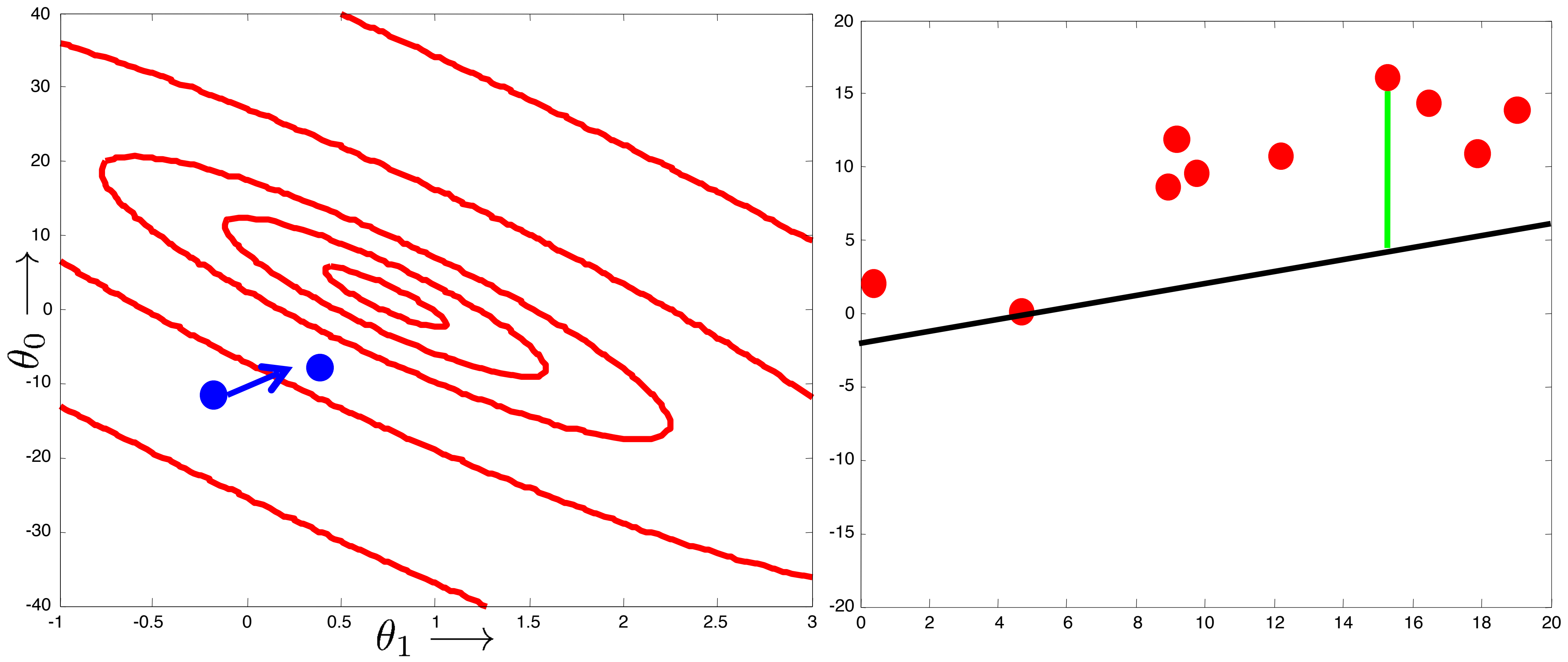
- Initialize θ
- Repeat:
  - Sample $j \sim \text{Uniform}(1,\ldots,m)$
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_\theta^{(j)}$
- Until some stop criterion; e.g., no average improvement in $\mathcal{L}_\theta^{(j)}$ for a while

Stochastic Gradient Descent



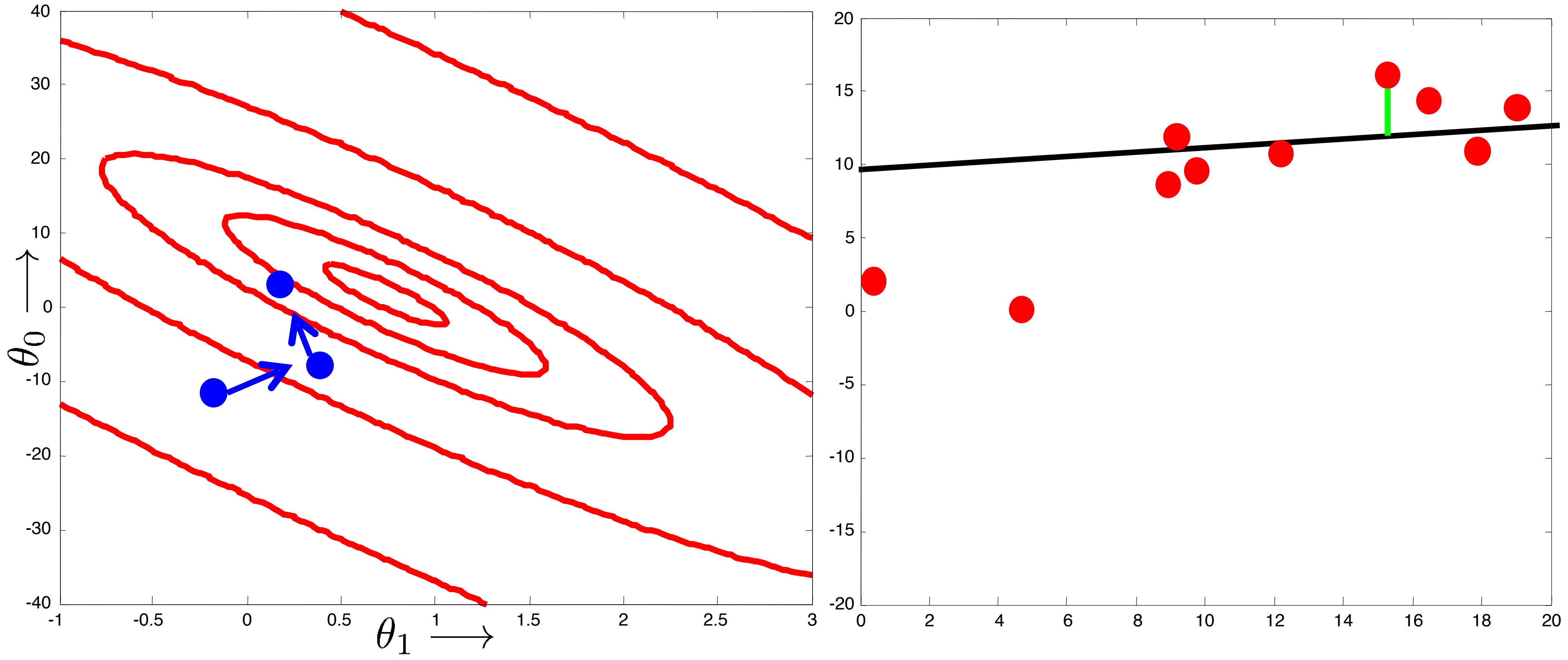
- Initialize θ
- Repeat:
  - Sample $j \sim \text{Uniform}(1,\ldots,m)$
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathscr{L}_\theta^{(j)}$
- Until some stop criterion; e.g., no average improvement in $\mathscr{L}_\theta^{(j)}$ for a while

Stochastic Gradient Descent



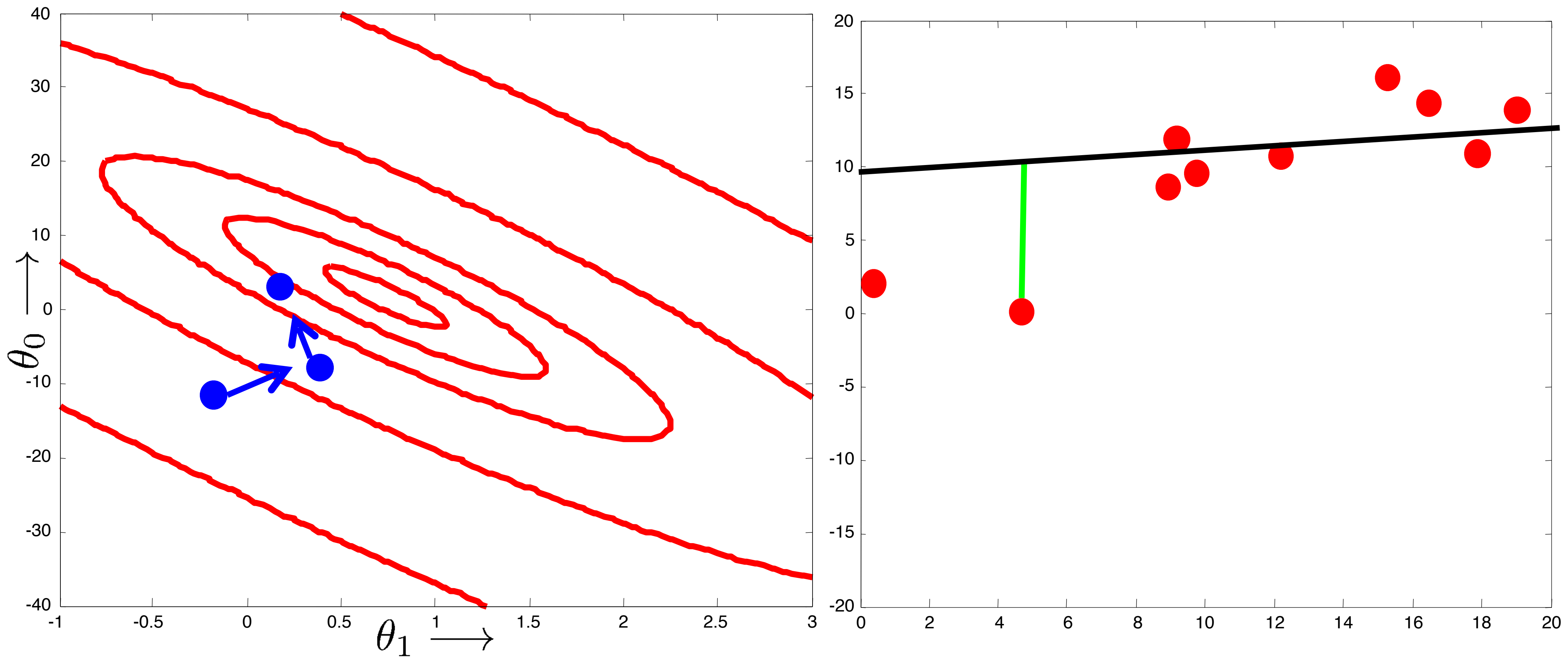
- Initialize θ
- Repeat:
    - Sample $j \sim \text{Uniform}(1,\ldots,m)$
    - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathscr{L}_\theta^{(j)}$
- Until some stop criterion; e.g., no <u>average</u> improvement in $\mathscr{L}_\theta^{(j)}$ for a while

Stochastic Gradient Descent



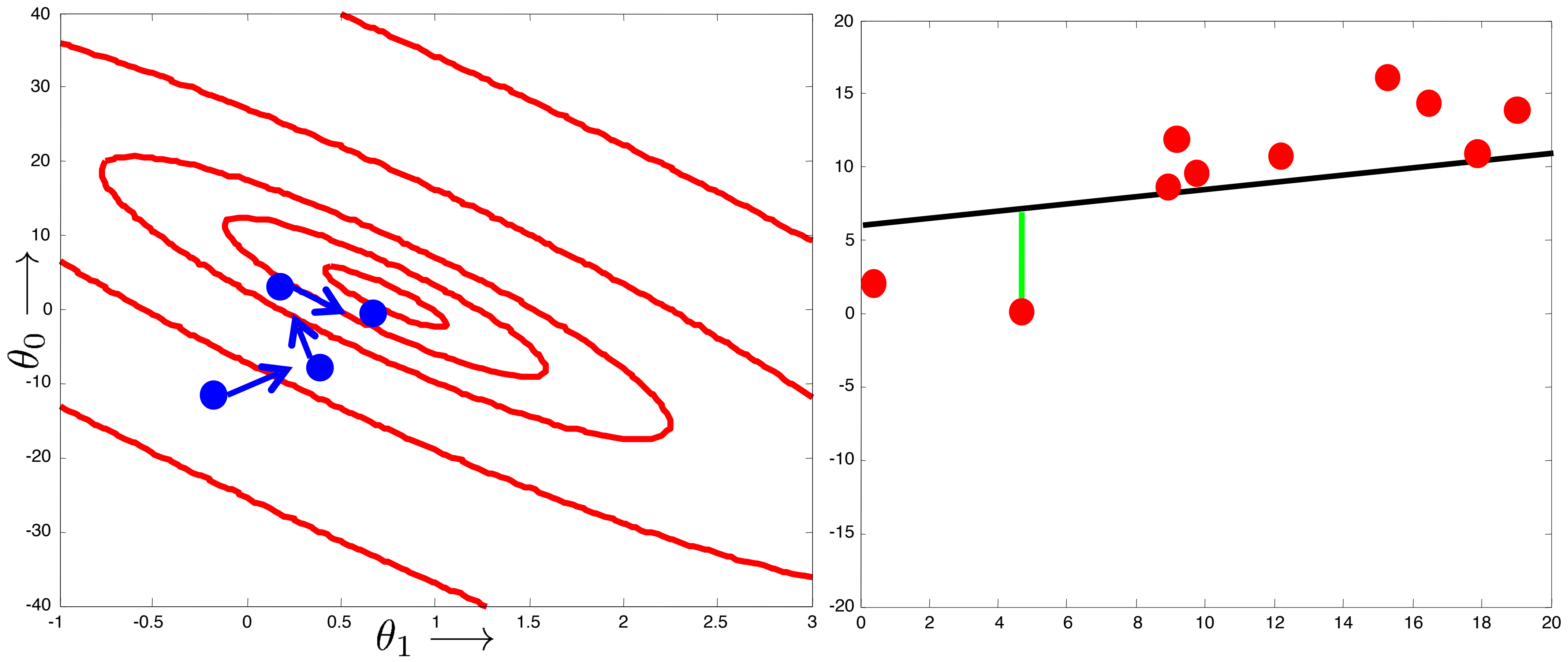
- Initialize θ
- Repeat:
  - Sample $j \sim \text{Uniform}(1,\ldots,m)$
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathscr{L}_\theta^{(j)}$
- Until some stop criterion; e.g., no average improvement in $\mathscr{L}_\theta^{(j)}$ for a while

Stochastic Gradient Descent



- Initialize θ
- Repeat:
  - Sample $j \sim \text{Uniform}(1,\ldots,m)$
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathscr{L}_\theta^{(j)}$
- Until some stop criterion; e.g., no <u>average</u> improvement in $\mathscr{L}_\theta^{(j)}$ for a while

Stochastic Gradient Descent



- Initialize θ

- Repeat:

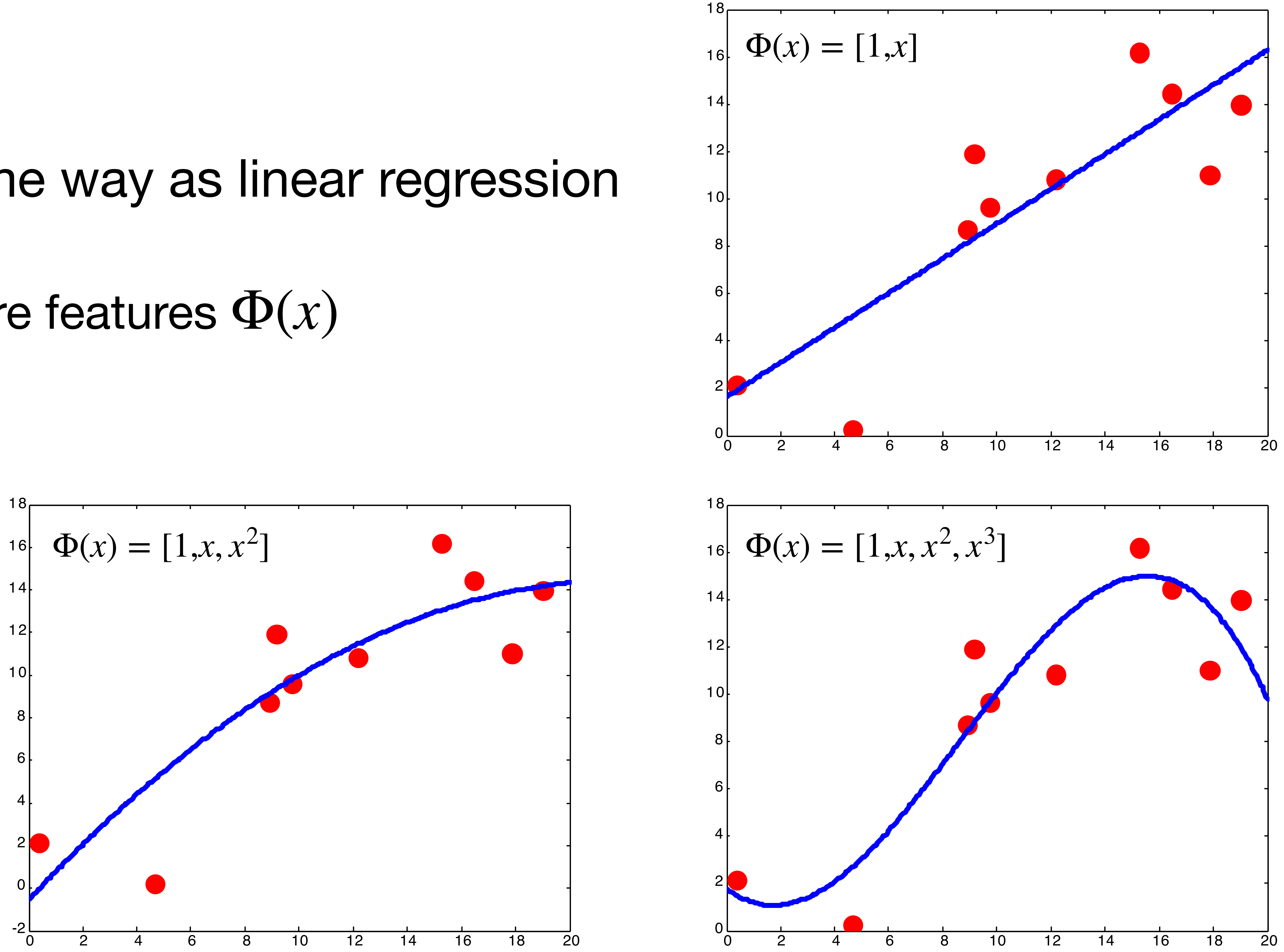
  - Sample $j \sim \text{Uniform}(1, \ldots, m)$

  - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathscr{L}_\theta^{(j)}$

- Until some stop criterion; e.g., no <u>average</u> improvement in $\mathscr{L}_\theta^{(j)}$ for a while

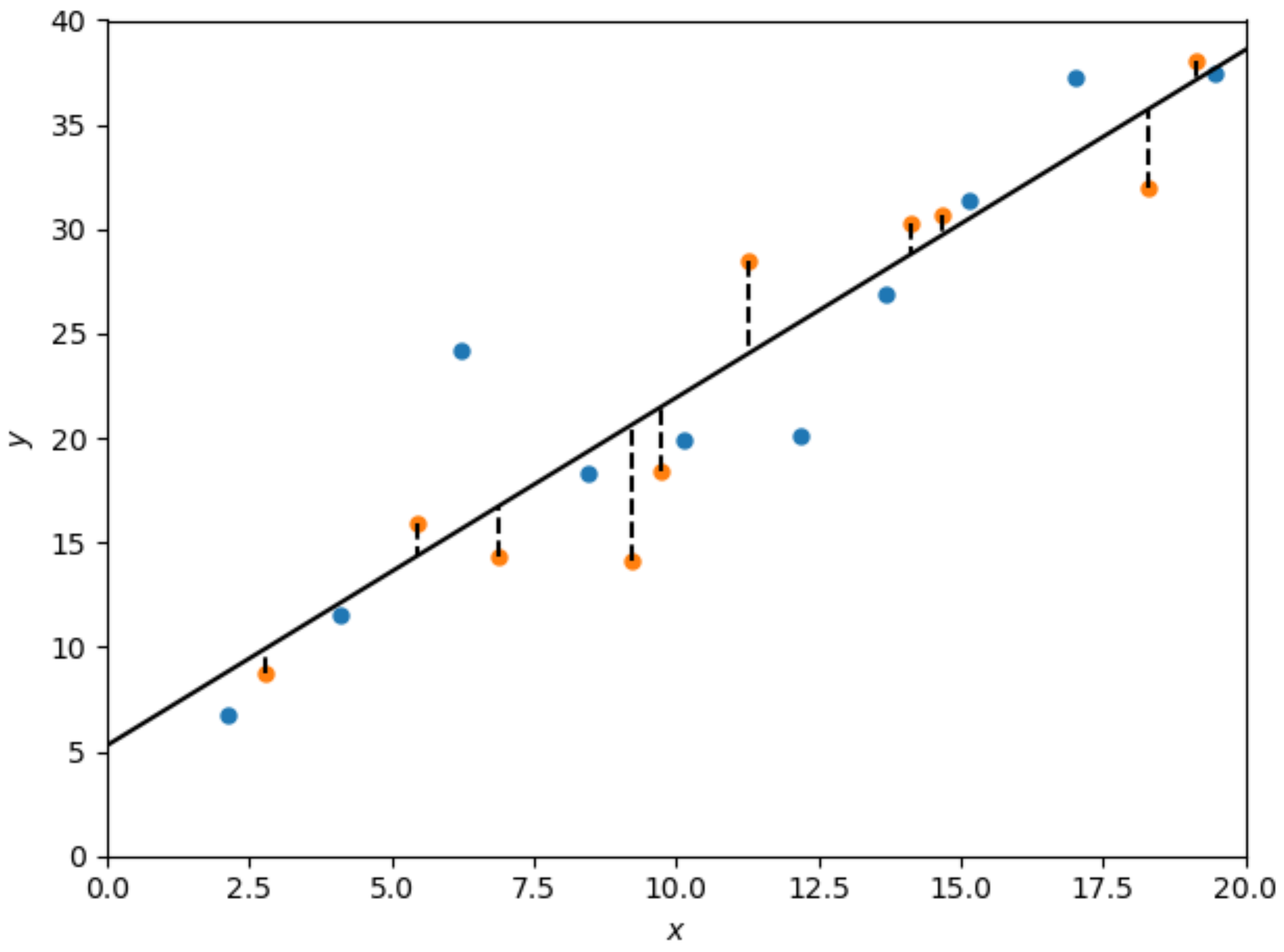
Polynomial regression

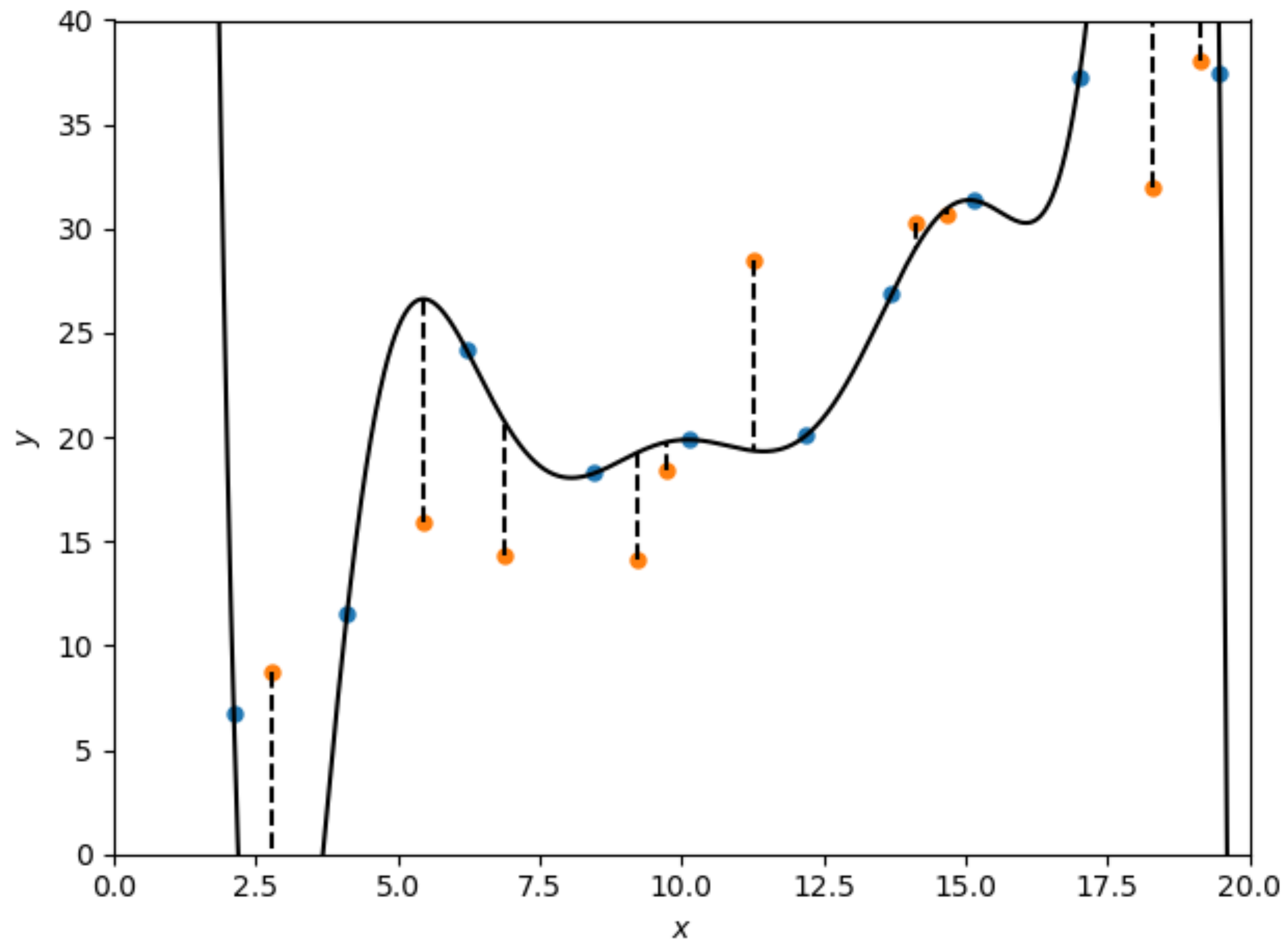
- Fit the same way as linear regression
  - With more features $\Phi(x)$

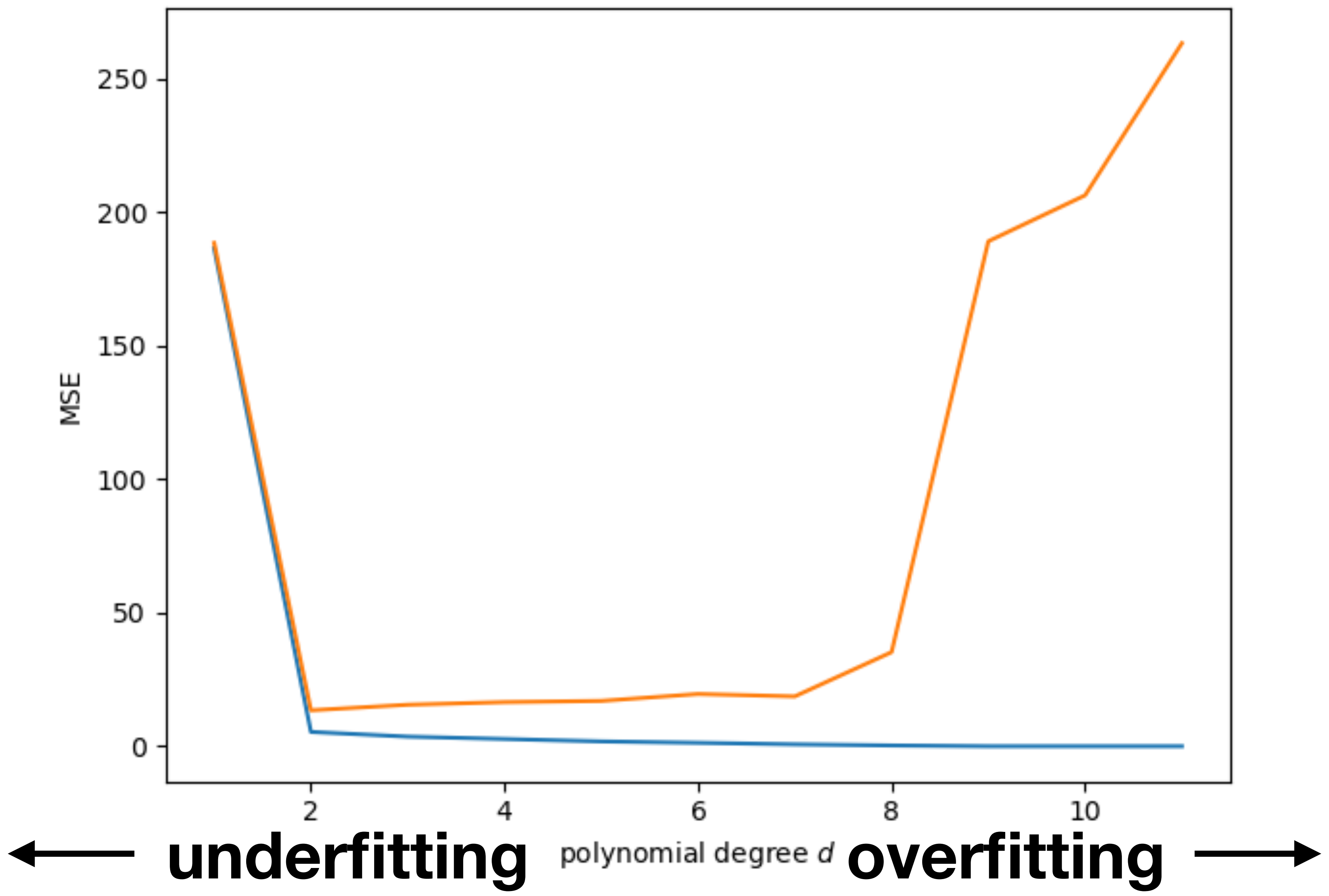


How many features to add?

- The more features we add, the more complex the model class
- Learning can always fall back to simpler model with $\theta_4 = \theta_5 = \cdots = 0$
- But generally it won't, it will overfit
  - Better training data fit, worse test data fit





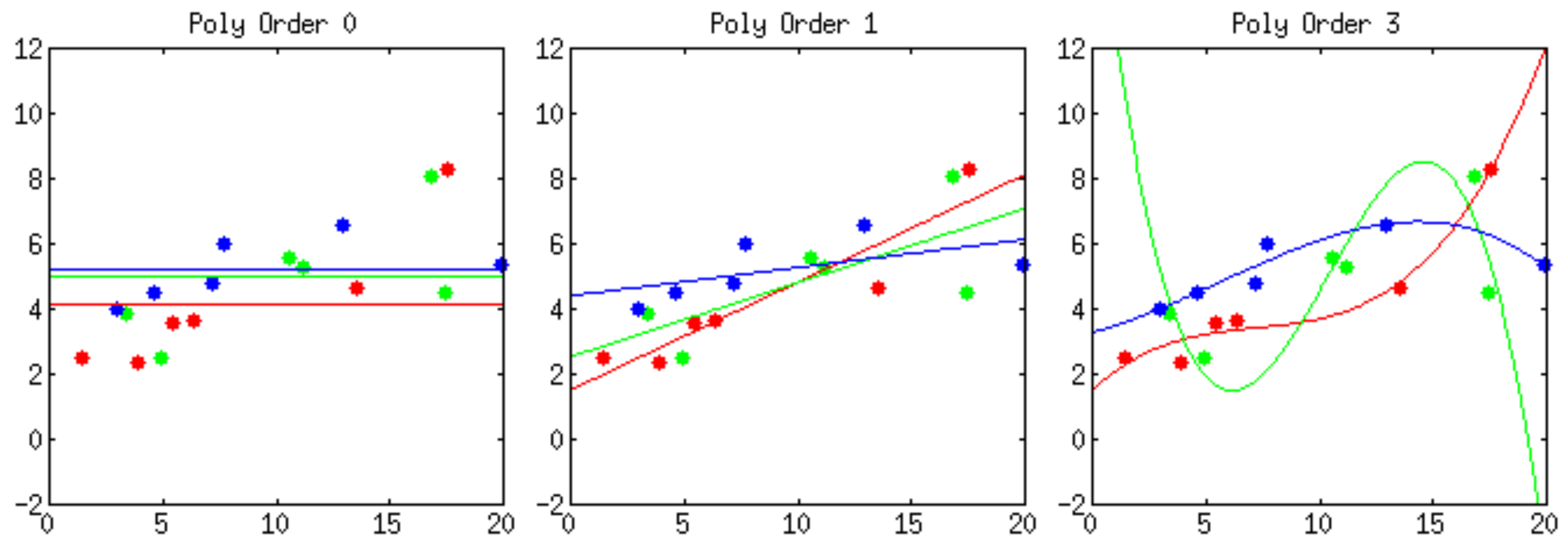


Bias–variance tradeoff

- For given test (x, y)
  - Expected MSE over datasets decomposes into bias and variance:

$$\mathbb{E}_{\mathscr{D}}[(y - \hat{y}_{\theta(\mathscr{D})}(x))^2] = (\mathbb{E}_{\mathscr{D}}[\hat{y}] - y)^2 + \mathbb{E}_{\mathscr{D}}[(\hat{y} - \mathbb{E}_{\mathscr{D}}[\hat{y}])^2] = (\mathrm{bias}_{\mathscr{D}}[\hat{y}])^2 + \mathrm{var}_{\mathscr{D}}[\hat{y}]$$

- Both components contribute equally to the quality of our algorithm
  - We can generally improve one at the expense of the other
    - Bias generally decreases with complexity
    - Variance generally increases with complexity

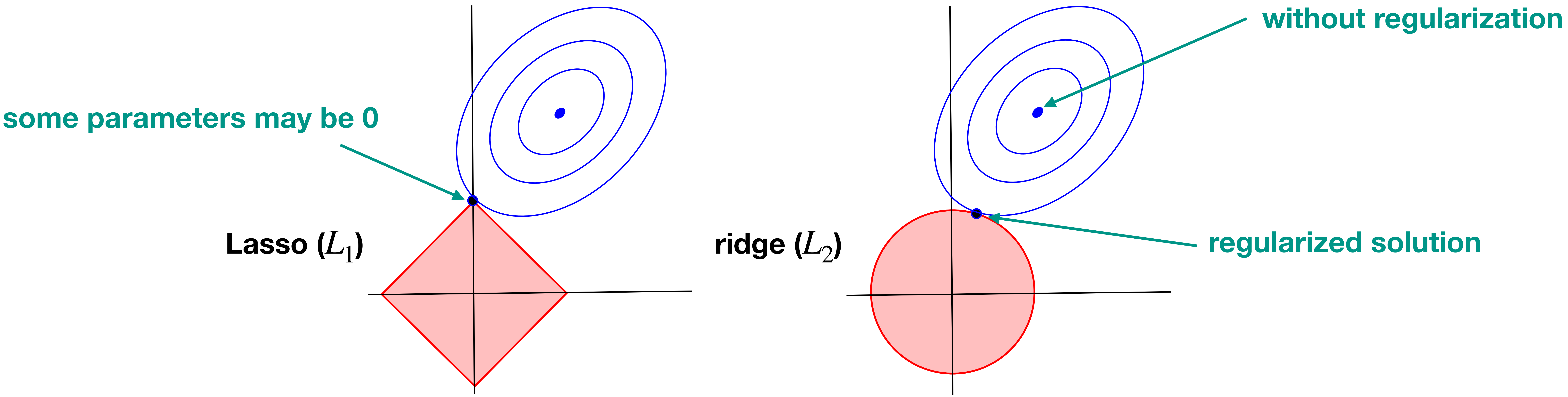


L_2 regularization

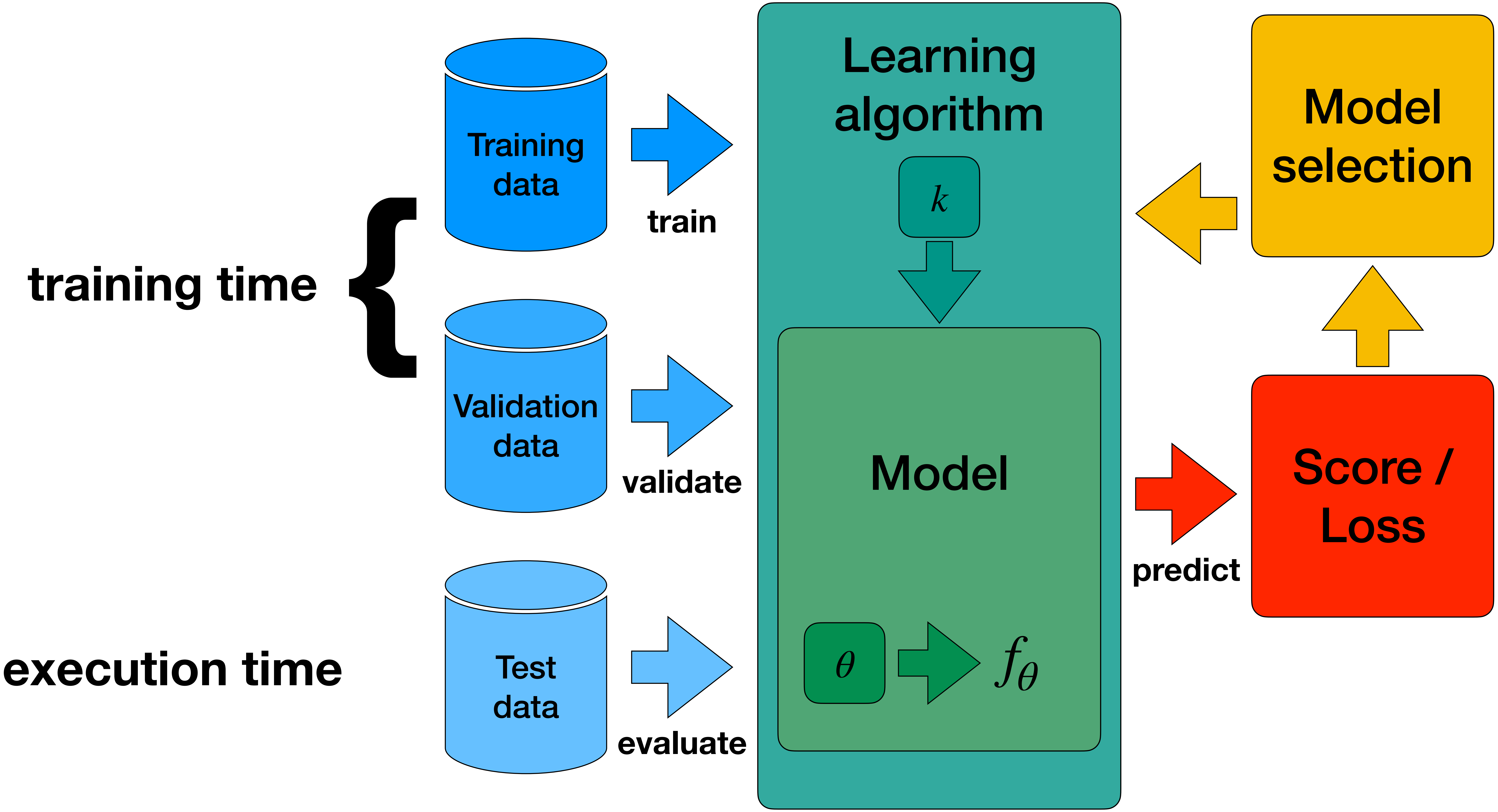
- Modify the loss function by adding a regularization term
- L_2 regularization (ridge regression) for MSE: $\mathscr{L}_\theta = \frac{1}{2}(\|y - \theta^\intercal X\|^2 + \alpha\|\theta\|^2)$
- Optimally: $\theta^\intercal = yX^\intercal(XX^\intercal + \alpha I)^{-1}$
  - αI moves $XX^\intercal$ away from singularity → inverse exists, better "conditioned"
  - Shrinks θ towards 0 (as expected)
    - At the expense of training MSE
- Regularization term $\alpha\|\theta\|^2$ independent of data = prior?

Regularization: L_1 vs. L_2

- θ estimate balances training loss and regularization
- Lasso (L_1) tends to generate sparser solutions than ridge (L_2) regularizer

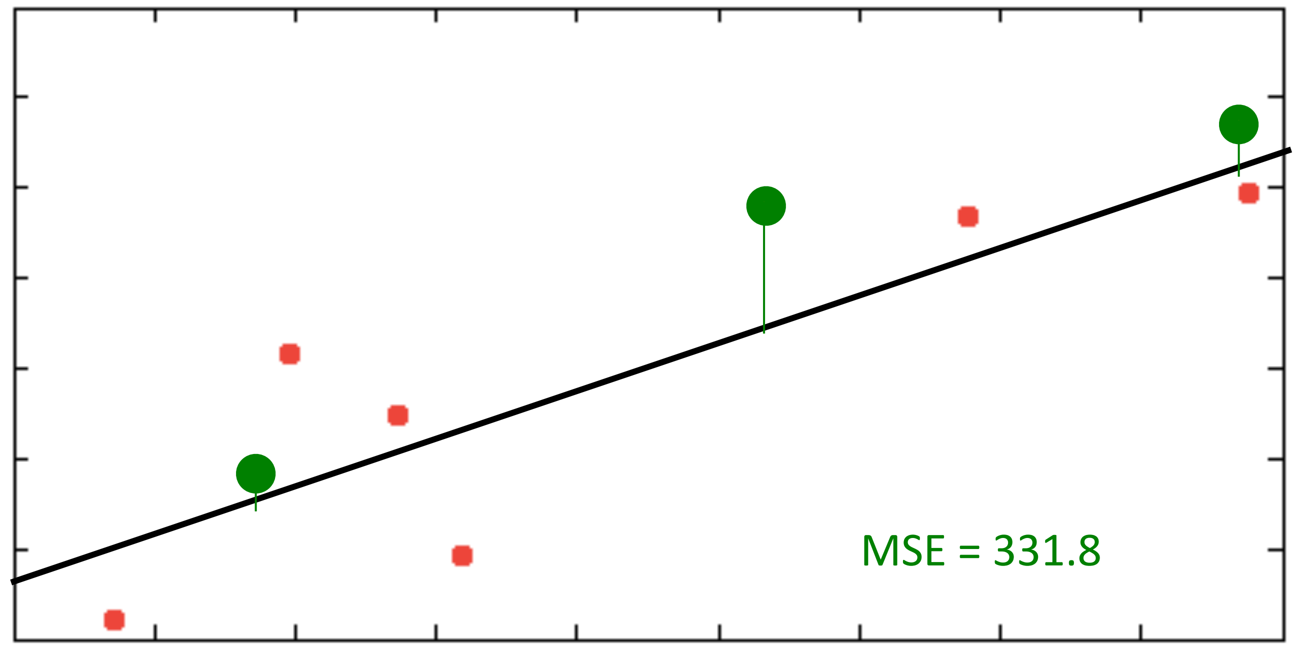


Model selection



Hold-out method

- Hold out some data for validation; e.g., random 30% of the data
  - Don't just sample training + validation with repetitions — they must be disjoint
- How to split?
  - Too few training data points → poor training, bad θ
  - Too few validation data points → poor validation, bad loss estimate
- Can we use more splits?



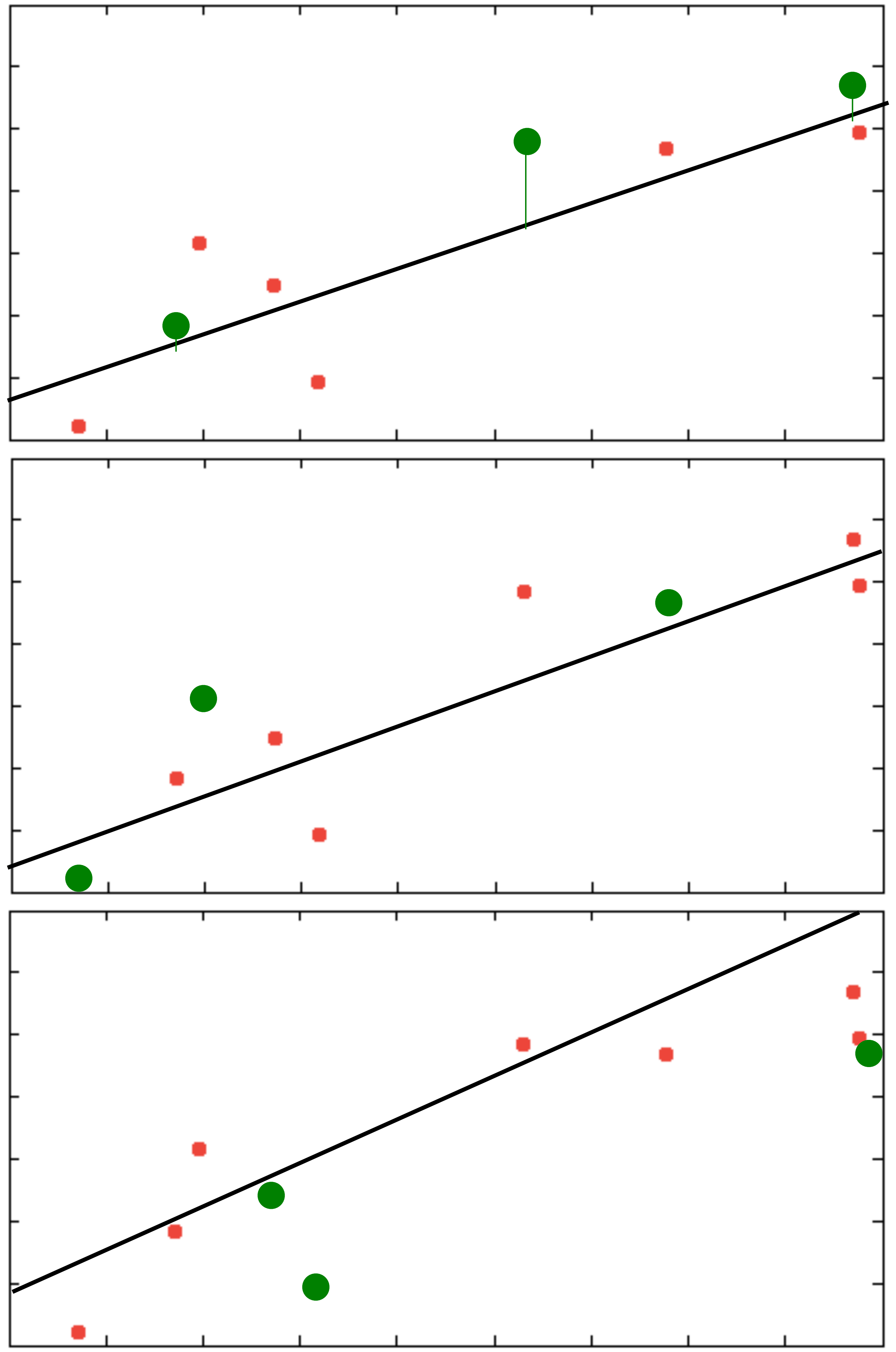
$x^{(i)}$	$y^{(i)}$
88	79
32	-2
27	30
68	73
7	-16
20	43
53	77
17	16
87	94

training data

validation data

k-fold cross-validation method

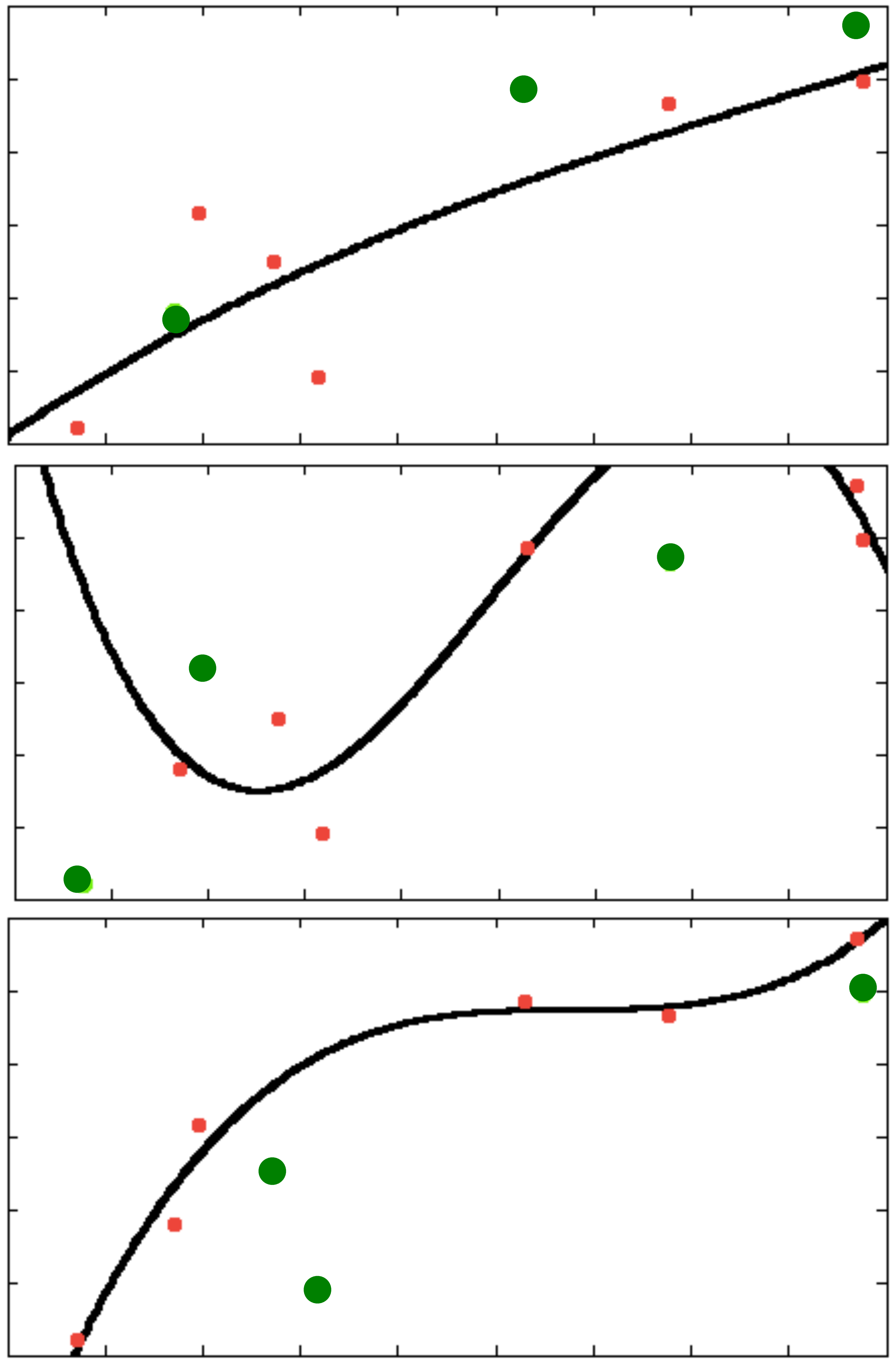
- Benefits:
  - Use all data for validation
  - Use all data to train final model



$x^{(i)}$	$y^{(i)}$
88	79
32	-2
27	30
68	73
7	-16
20	43
53	77
17	16
87	94

k-fold cross-validation method

- Benefits:
  - Use all data for validation
  - Use all data to train final model
- Drawbacks:
  - Trains k (+1) models
  - Each model still gets noisy validation from $\frac{m}{k}$ data points
  - No validation for the final model
- When $k = m$: Leave-One-Out (LOO)



Split 1:
MSE = 280.5

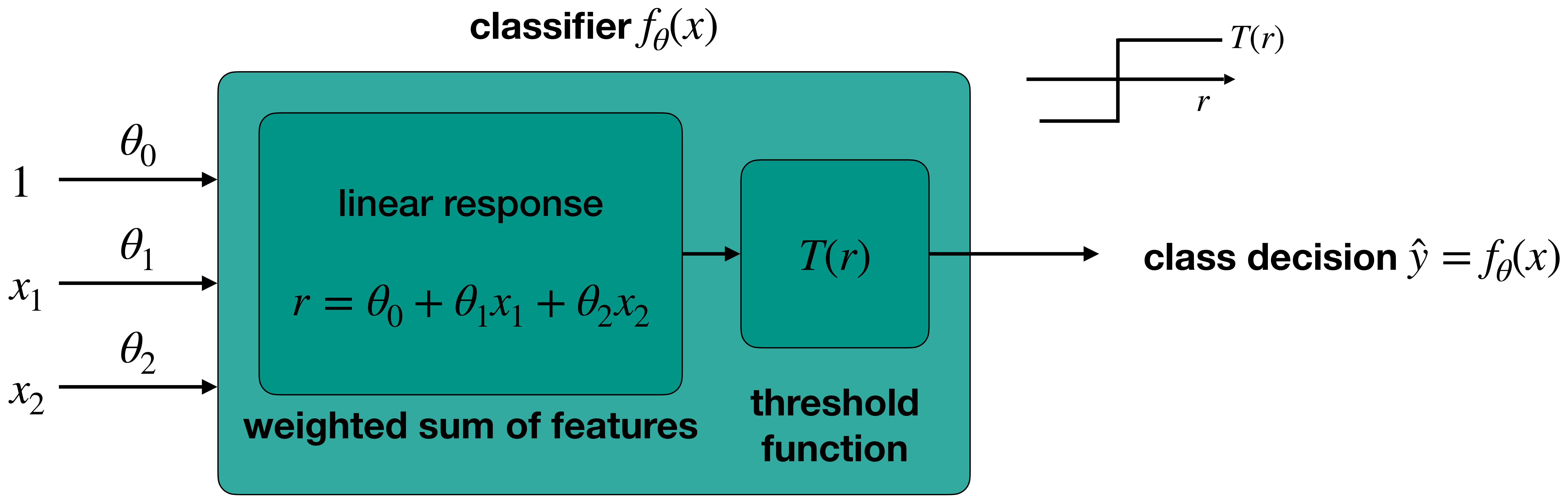
Split 2:
MSE = 3081.3

Split 3:
MSE = 1640.1

3-Fold X-Val MSE
= 1667.3

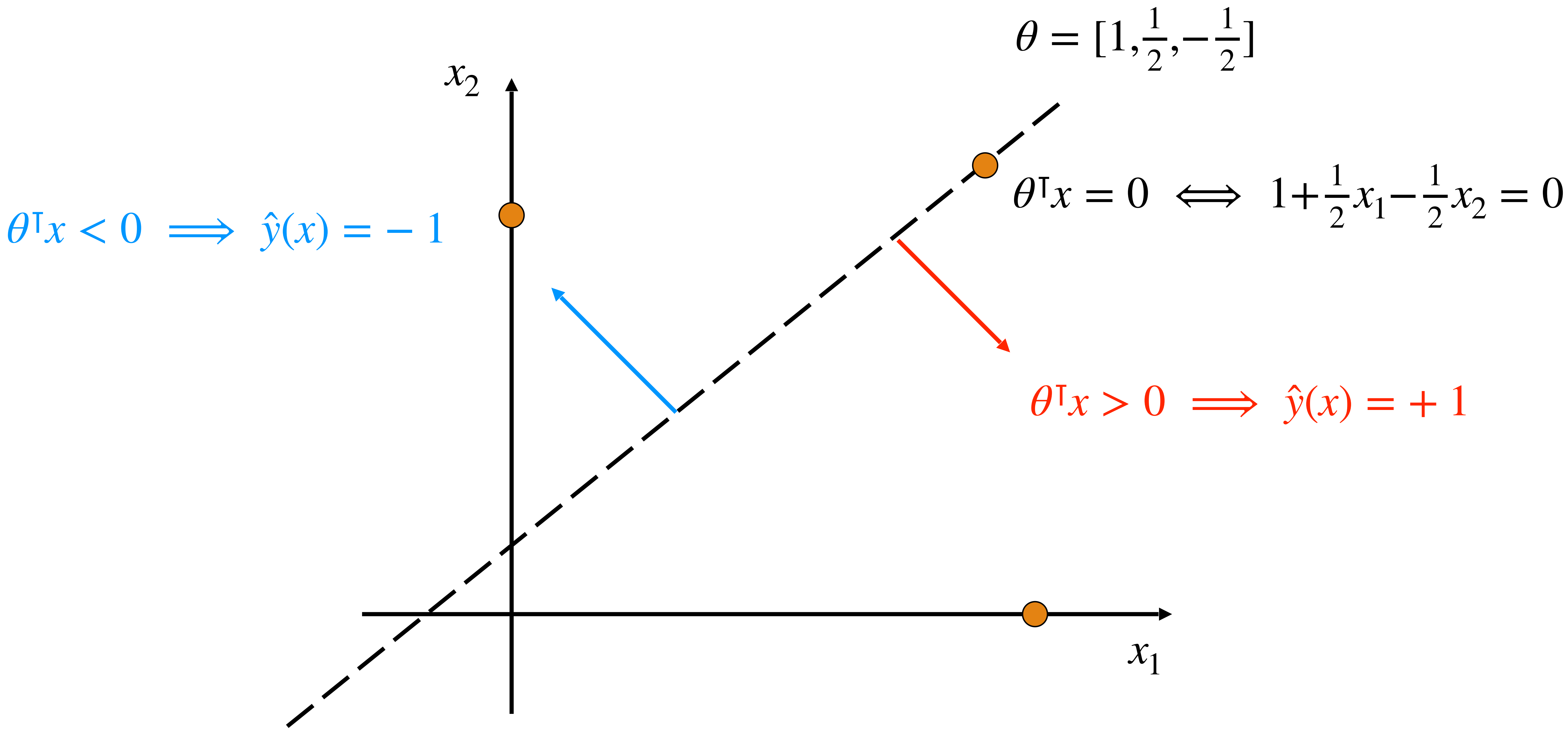
$x^{(i)}$	$y^{(i)}$
88	79
32	-2
27	30
68	73
7	-16
20	43
53	77
17	16
87	94

Perceptron




```
r = theta.T @ X         # compute linear response
y_hat = (r > 0)         # predict class 1 vs. 0
y_hat = 2*(r > 0) - 1   # predict class 1 vs. -1
```

Example

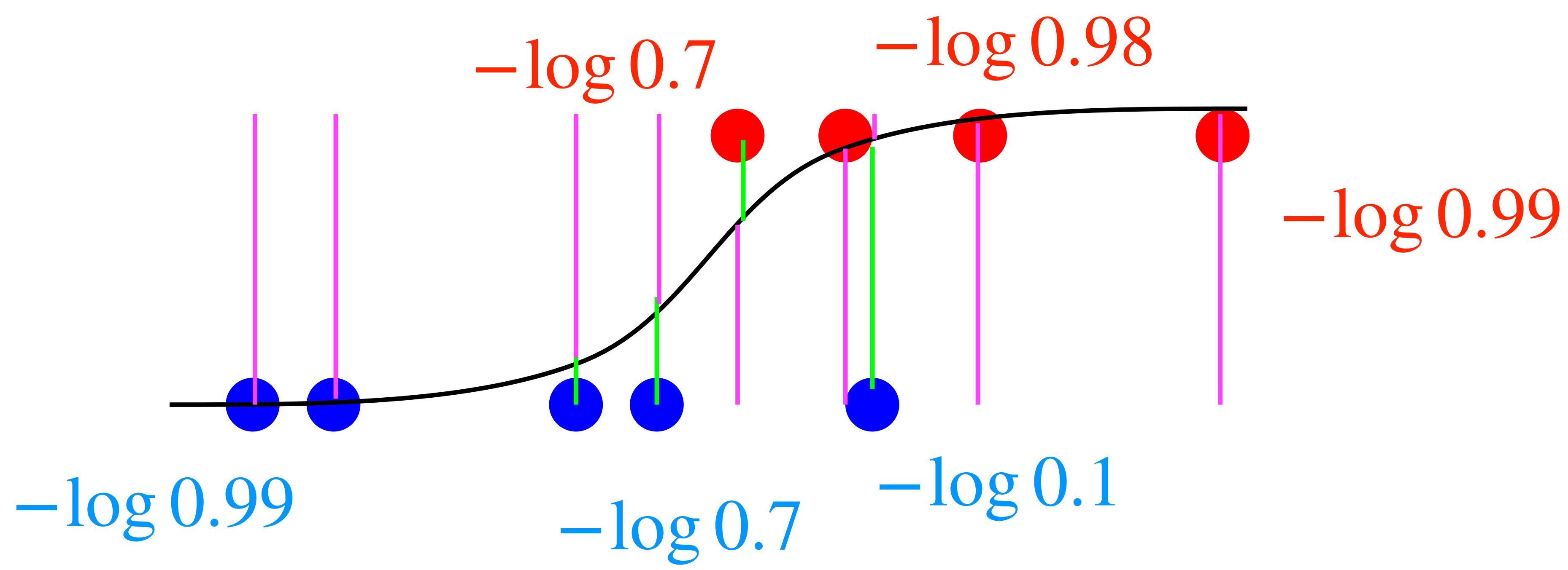


Adapted from Padhraic Smyth

Logistic Regression

- Can we turn a linear response into a probability? Sigmoid! $\sigma : \mathbb{R} \to [0,1]$
- Think of $\sigma(\theta^\intercal x) = p_\theta(y = 1 \mid x)$
- Negative Log-Likelihood (NLL) loss:

$$\mathscr{L}_\theta(x, y) = -\log p_\theta(y \mid x) = -\underbrace{y \log \sigma(\theta^\intercal x)}_{\text{for } y=1} - \underbrace{(1-y)\log(1-\sigma(\theta^\intercal x))}_{\text{for } y=0}$$



Logistic Regression: gradient

- Logistic NLL loss: $\mathscr{L}_\theta(x,y) = -y\log\sigma(\theta^\intercal x) - (1-y)\log(1-\sigma(\theta^\intercal x))$

- Gradient:

$$-\nabla_\theta \mathscr{L}_\theta(x,y) = \left(y\frac{\sigma'(\theta^\intercal x)}{\sigma(\theta^\intercal x)} - (1-y)\frac{\sigma'(\theta^\intercal x)}{1-\sigma(\theta^\intercal x)}\right)x$$

$$= (\underbrace{y\,(1-\sigma(\theta^\intercal x))}_{\text{error for } y=1} - (1-y)\underbrace{\sigma(\theta^\intercal x)}_{\text{error for } y=0})x$$

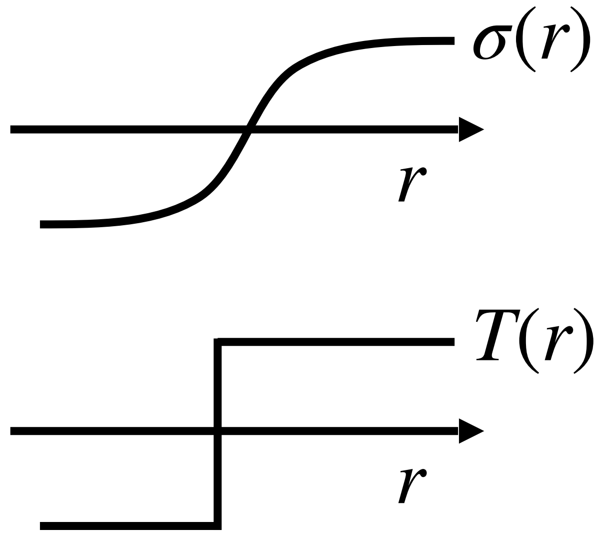
$$= (y - p_\theta(y=1|x))x$$

error for $y = 1$

error for $y = 0$

but update toward $-x$

- Compare:



  - Perceptron: $(y-\hat{y})x$ ⟵ constant error (± 2), insensitive to margin
  - Logistic MSE: $-\nabla_\theta \mathscr{L}_\theta(x,y) = 2(y-\sigma(\theta^\intercal x))\sigma'(\theta^\intercal x)x$ ⟵ 0 gradient for bad mistakes

Multi-class linear models

- More generally: add features — can even depend on y!

$$f_\theta(x) = \arg\max_y \theta^\intercal \Phi(x, y)$$

- Example: $y \in \{1,2,\ldots,C\}$
  - $\Phi(x, y) = [0 \;\; 0 \;\; \cdots \;\; x \;\; \cdots \;\; 0] =$ one-hot$(y) \otimes x$
  - $\theta = [\theta_1 \;\; \cdots \;\; \theta_C]$

$$\implies f_\theta(x) = \arg\max_c \theta_c^\intercal x$$ ← **largest linear response**

Multi-class perceptron algorithm

- While not done:
  - For each data point $(x, y) \in \mathscr{D}$:
    - Predict: $\hat{y} = \arg\max_c \theta_c^\intercal x$
    - Increase response for true class: $\theta_y \leftarrow \theta_y + \alpha x$
    - Decrease response for predicted class: $\theta_{\hat{y}} \leftarrow \theta_{\hat{y}} - \alpha x$
- More generally:
  - Predict: $\hat{y} = \arg\max_y \theta^\intercal \Phi(x, y)$
  - Update: $\theta \leftarrow \theta + \alpha(\Phi(x, y) - \Phi(x, \hat{y}))$

Multilogit Regression

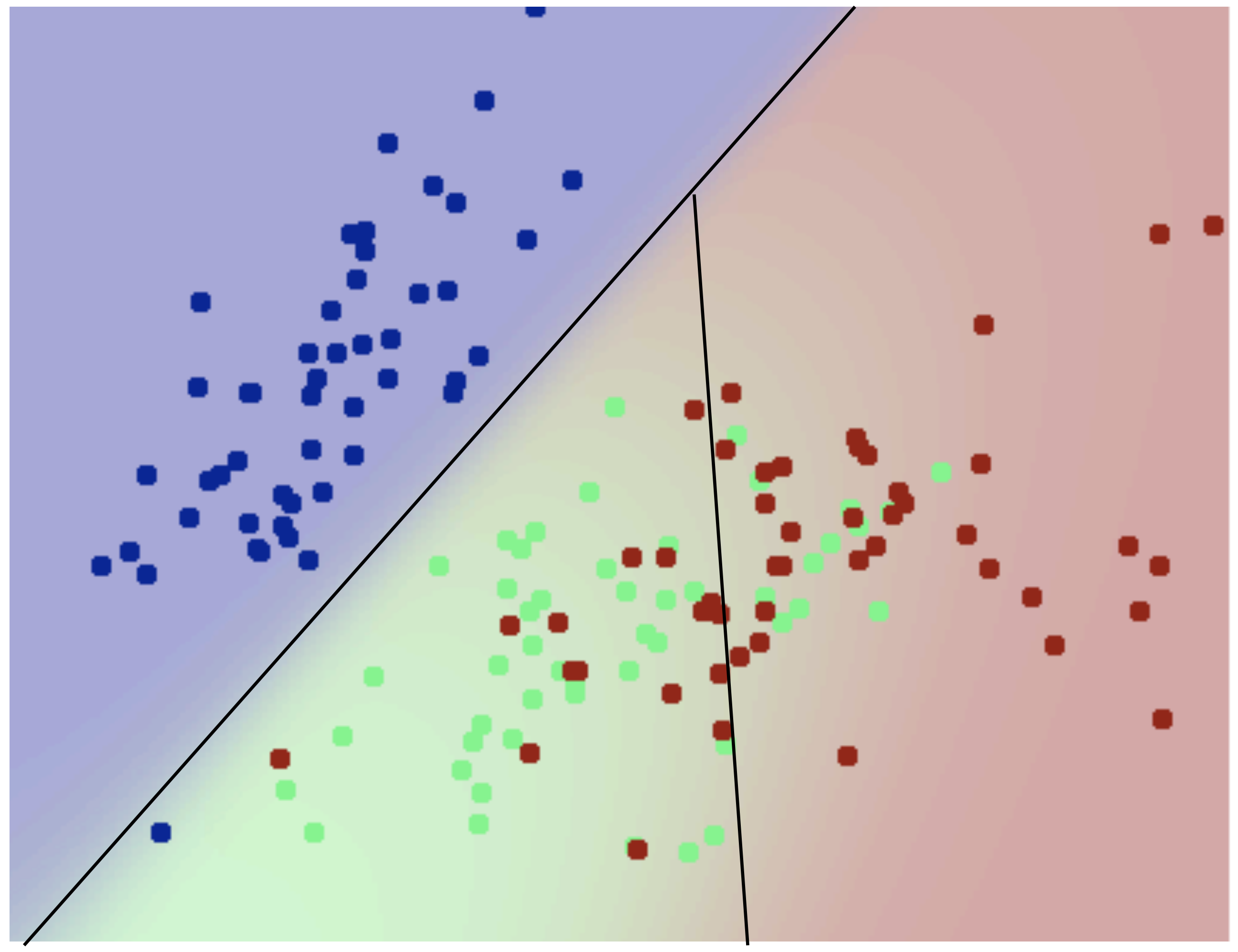
- Define multi-class probabilities: $p_\theta(y|x) = \dfrac{\exp(\theta_y^\intercal x)}{\sum_c \exp(\theta_c^\intercal x)} = \operatorname{soft\,max}_c \theta_c^\intercal x \Big|_y$

  "logit" for c

  - For binary y: $$p_\theta(y=1|x) = \frac{\exp(\theta_1^\intercal x)}{\exp(\theta_1^\intercal x) + \exp(\theta_2^\intercal x)} = \frac{1}{1+\exp((\theta_2-\theta_1)^\intercal x)} = \sigma((\theta_1-\theta_2)^\intercal x)$$

    Logistic Regression with $\theta = \theta_1 - \theta_2$

- Benefits:
  - Probabilistic predictions: knows its confidence
  - Linear decision boundary: $\arg\max_y \exp(\theta_y^\intercal x) = \arg\max_y \theta_y^\intercal x$
  - NLL is convex



Learning Decision Trees

- Start from empty decision tree
- Split on max-info-gain feature x_i
  - $\arg\max_i \mathbb{I}[x_i; y \,|\, b] = \arg\max_i \mathbb{H}[y \,|\, b] - \mathbb{H}[y \,|\, b, x_i]$
- Repeat for each sub-tree, until:
  - Entropy = 0 (all y are the same)
  - No more features
  - Information gain very small?
- Label leaf with majority y

Entropy reduction

- Select feature that most decreases uncertainty

- Entropy of y in branch b (before the next split):

$$\mathbb{H}[y \mid b] = -\sum_c p(y=c \mid b) \log p(y=c \mid b)$$

$$= -\tfrac{5}{8}\log\tfrac{5}{8} - \tfrac{3}{8}\log\tfrac{3}{8} = 0.66$$

- Entropy after splitting by x_1:

$$\mathbb{H}[y \mid b, x_1] = \mathbb{E}_{x_1 \mid b}[\mathbb{H}[y \mid b, x_1]] = -\sum_v p(x_1 = v \mid b) \sum_c p(y=c \mid b, x_1 = v) \log p(y = c \mid b, x_1 = v)$$

$$= -\tfrac{4}{8}(\tfrac{4}{4}\log\tfrac{4}{4} + \tfrac{0}{4}\log\tfrac{0}{4}) - \tfrac{4}{8}(\tfrac{1}{4}\log\tfrac{1}{4} + \tfrac{3}{4}\log\tfrac{3}{4}) = 0.28$$

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

Information gain

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

- Information gain = reduction in entropy from conditioning y on x_1
    - The amount of new information that x_1 has on y

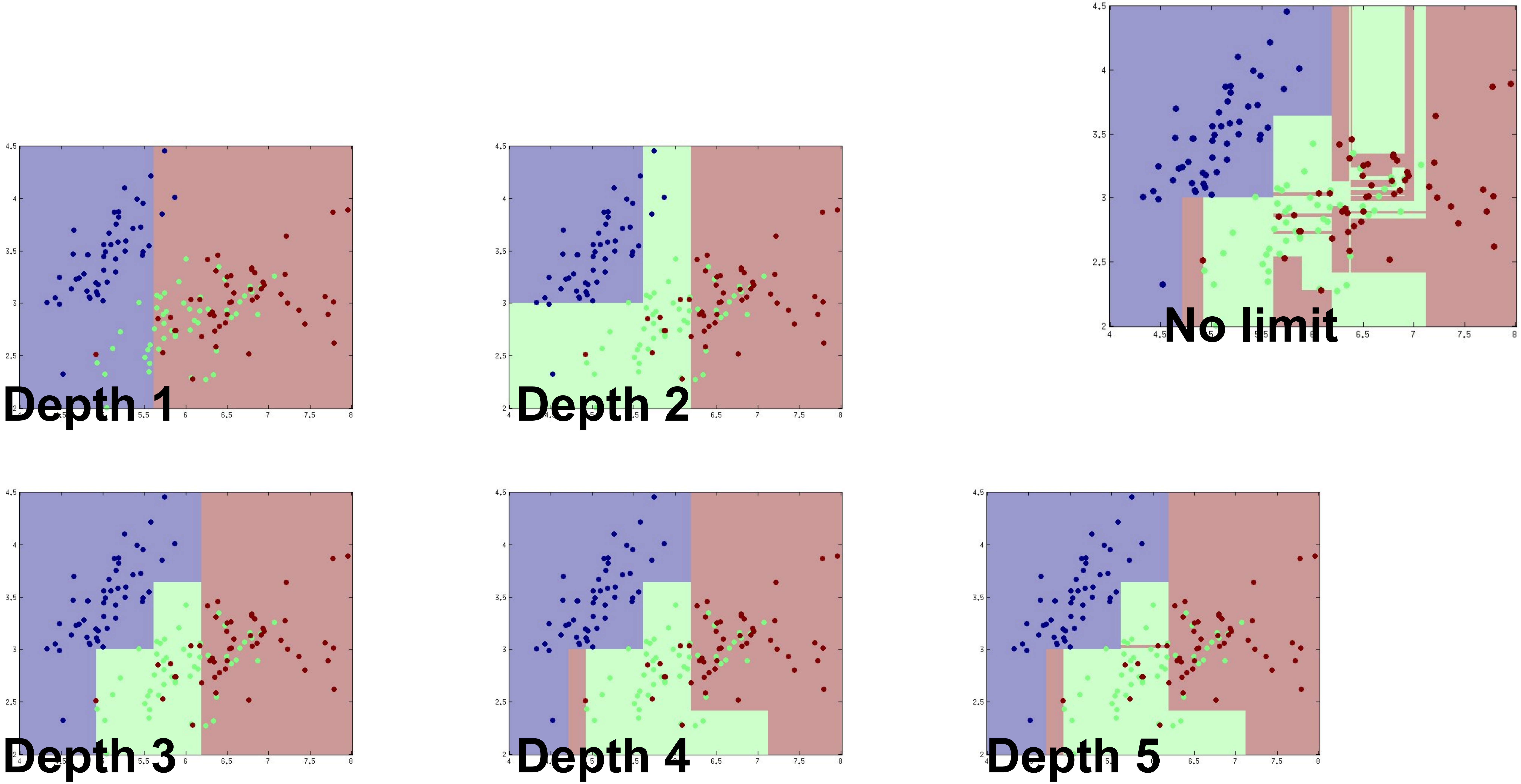
$$\mathbb{I}[x_1; y \mid b] = \mathbb{H}[y \mid b] - \mathbb{H}[y \mid b, x_1] = 0.66 - 0.28 = 0.38$$

$$\mathbb{I}[x_2; y \mid b] = 0.66 - 0.63 = 0.03$$

select x_1 for Decision Tree

- Information gain is always non-negative
    - By convexity of the entropy

Controlling complexity

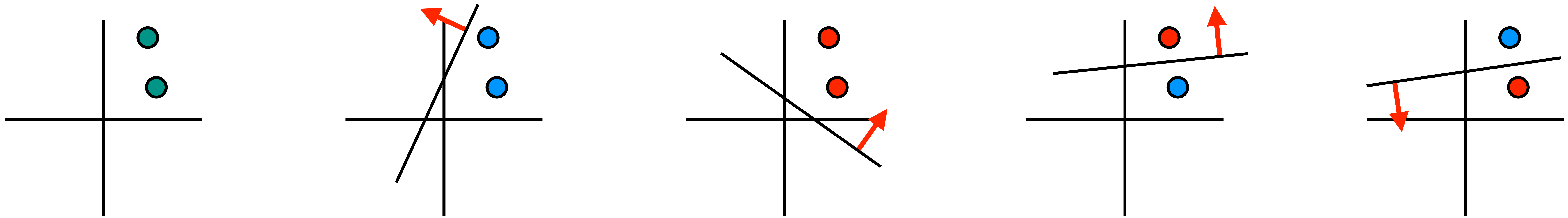


Shattering

- Separability / realizability: there's a model that classifies all points correctly
- Shattering: the points are separable regardless of their labels
  - Our model class can shatter points $x^{(1)}, \ldots, x^{(h)}$

    if for any labeling $y^{(1)}, \ldots, y^{(h)}$

    there exists a model that classifies all of them correctly
- Example: can $f_\theta(x) = \text{sign}(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ shatter these points?

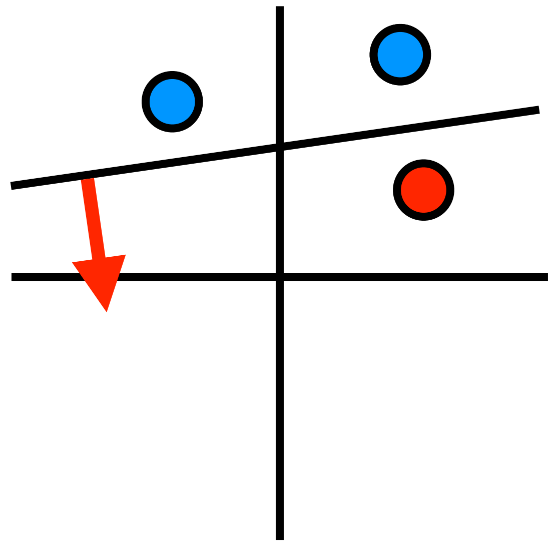


Vapnik–Chervonenkis (VC) dimension

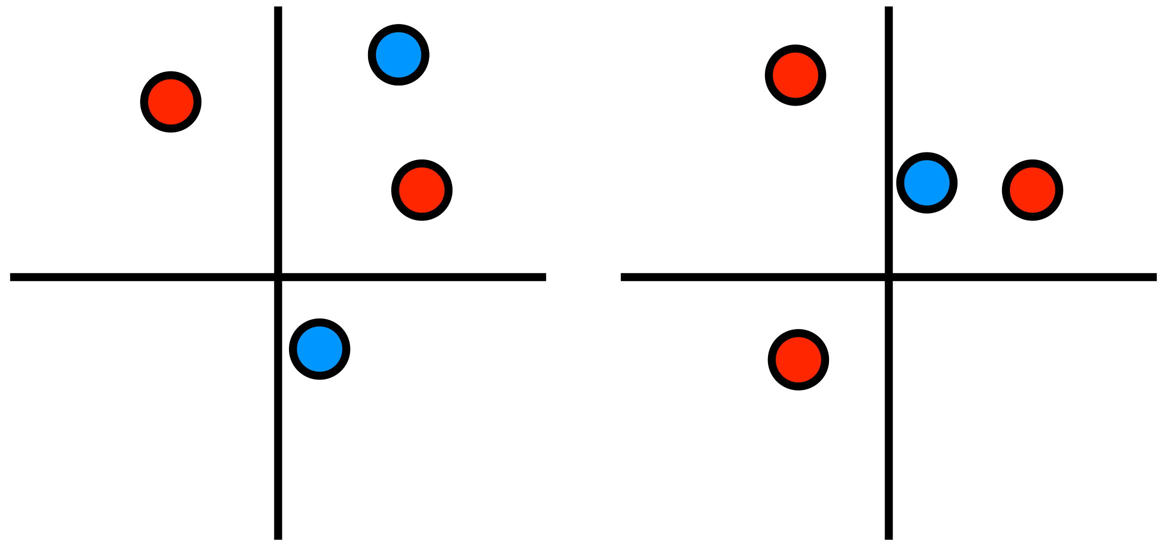
- VC dimension: maximum number H of points that can be shattered by a class
- A game:
  - Fix a model class $f_\theta : x \to y \quad \theta \in \Theta$
  - Player 1: choose h points $x^{(1)}, \dots, x^{(h)}$
  - Player 2: choose labels $y^{(1)}, \dots, y^{(h)}$
  - Player 1: choose model θ
  - Are all $y^{(j)} = f_\theta(x^{(j)})$? $\implies$ Player 1 wins $\quad \exists x^{(1)}, \dots, x^{(h)} : \ \forall y^{(1)}, \dots, y^{(h)} : \ \exists \theta : \ \forall j : \ y^{(j)} = f_\theta(x^{(j)})$
- $h \leq H \implies$ Player 1 can win, otherwise cannot win

VC dimension: example (2)

- Example: $f_\theta(x) = \text{sign}(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
  - We can place 3 points and shatter them



  - We can prevent shattering <u>any</u> 4 points:
    - If they form a convex shape, alternate labels
    - Otherwise, label differently the point in the triangle



  - $H = 3$
- Linear classifiers (perceptrons) of d features have VC-dim $d + 1$
  - But VC-dim is generally not #parameters

Model selection with VC-dim

- Using validation / cross-validation:
  - Estimate loss on held out set
  - Use validation loss to select model

- Using VC dimension:
  - Use generalization bound to select model
  - Structural Risk Minimization (SRM)
  - Bound not tight, much too conservative