UCI University of California, Irvine

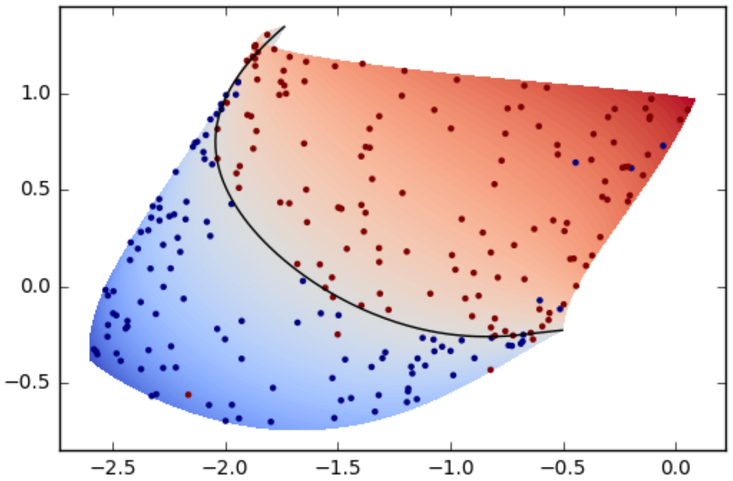
CS 273A: Machine Learning

Winter 2021

Lecture 6: Regularization

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



Logistics

assignments

- Assignment 2 due next Tuesday, Jan 26

project

- Project guidelines on Canvas:
  - https://canvas.eee.uci.edu/courses/34497/pages/projects
- Team rosters due next Thursday, Jan 28 **on Canvas**

Today's lecture

Polynomial regression

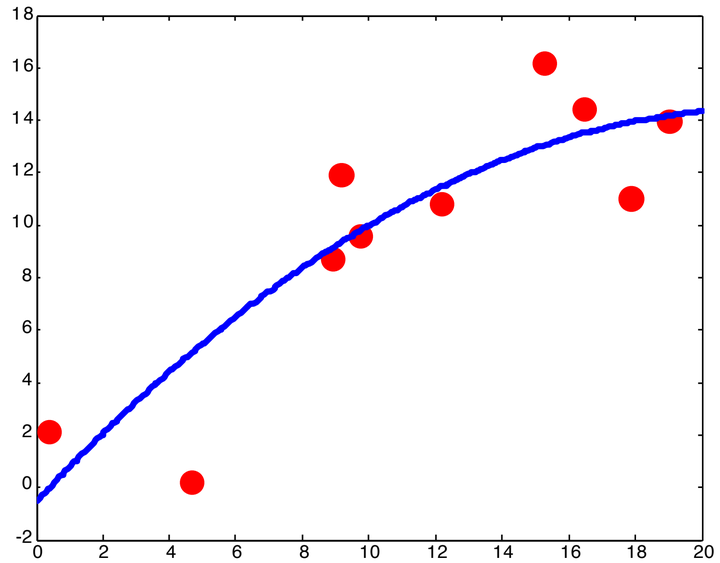
Inductive bias and regularization

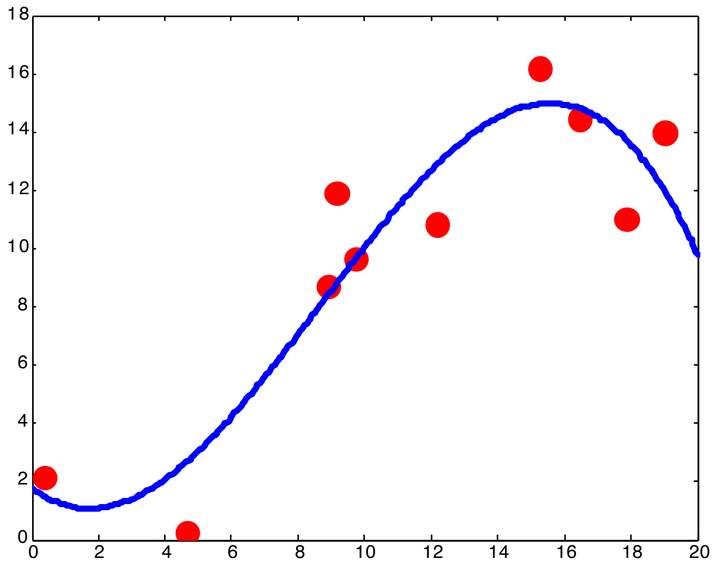
Cross-validation

Linear classification

Polynomial regression

- Some data cannot be explained by linear regression
  - A higher-order polynomial may be a better fit





Polynomial regression

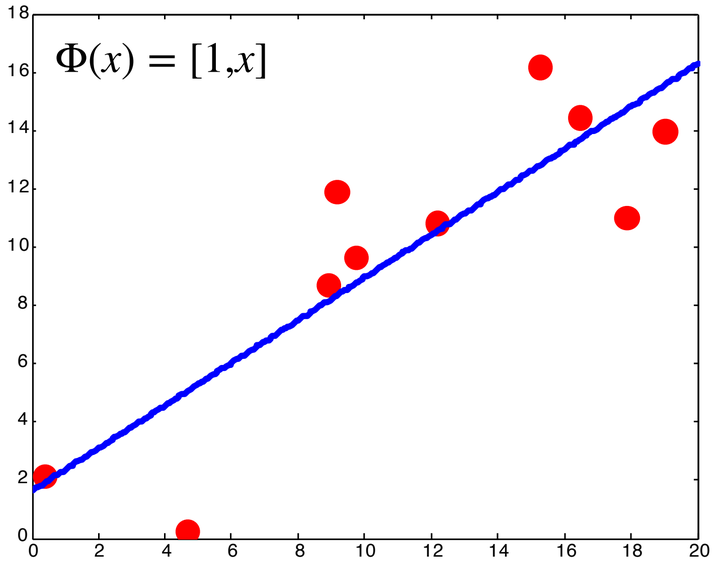
- Consider a polynomial in a single feature x

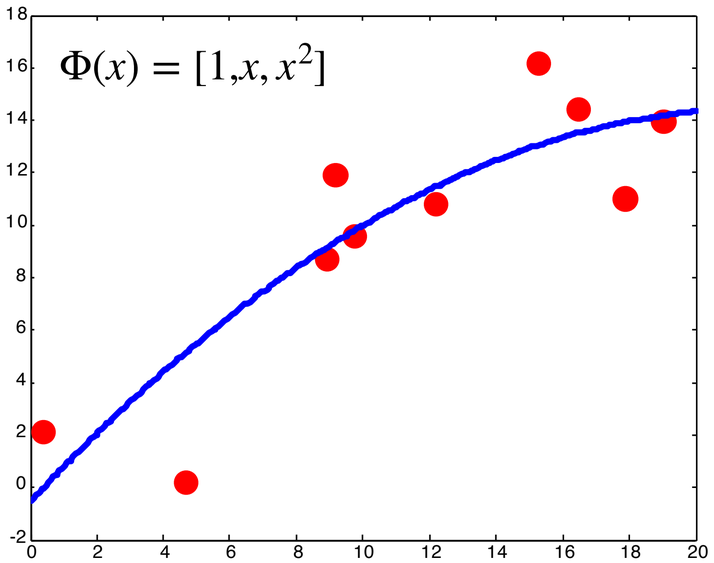
$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \cdots$$

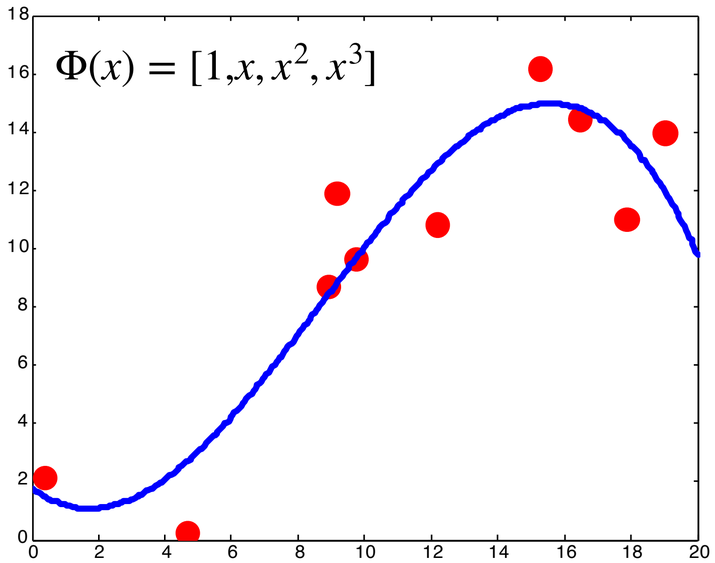
- Can we reduce this to something we already know?
  - Think of higher-order terms $x^2, x^3, \ldots$ as new features
- $\mathscr{D} = \{(x^{(j)}, y^{(j)})\} \implies \{([x^{(j)}, (x^{(j)})^2, (x^{(j)})^3, \ldots], y^{(j)})\}$
  - Denote $\Phi(x) = [x, x^2, x^3, \ldots]$
- Perform linear regression with $\hat{y} = \theta^\intercal \Phi(x)$

Polynomial regression

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





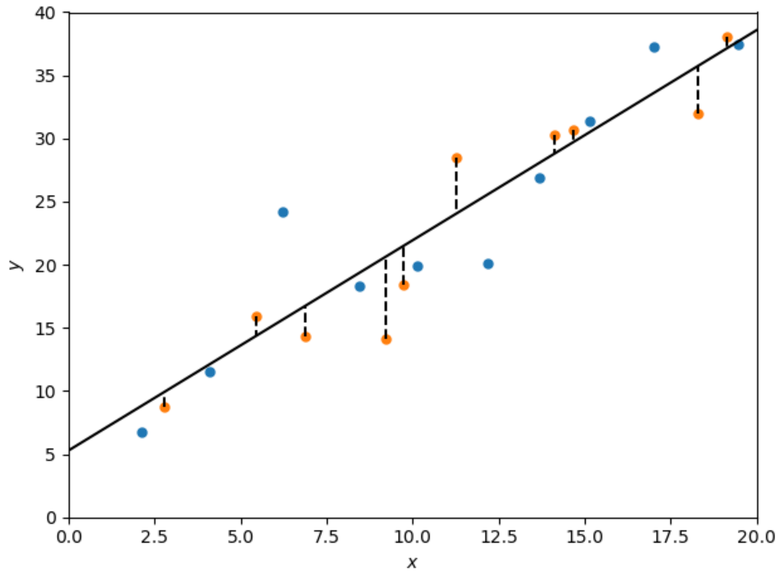


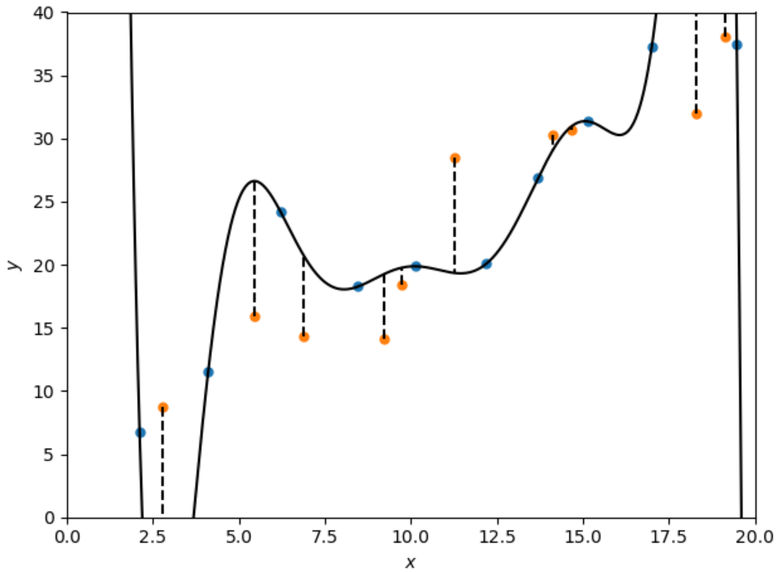
Feature expansion

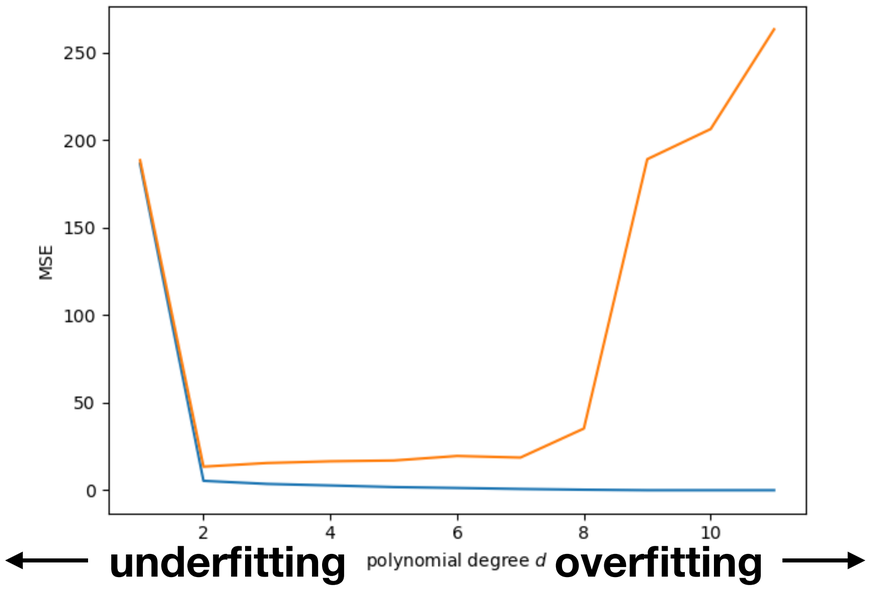
- In principle, can use any features we think are useful
- Instead of collecting more information per data point
  - apply nonlinear transformation to x to get more “linear explainability” of y
- More examples:
  - Cross-terms between features: $x_i x_j$, $x_i x_j x_k$, ...
  - Trigonometric functions: $sin(\omega x + \phi)$
  - Others: $\frac{1}{x}$, $\sqrt{x}$, ...
- Linear regression = linear in θ, the features can be as complex as we want

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







Today's lecture

Polynomial regression

Inductive bias and regularization

Cross-validation

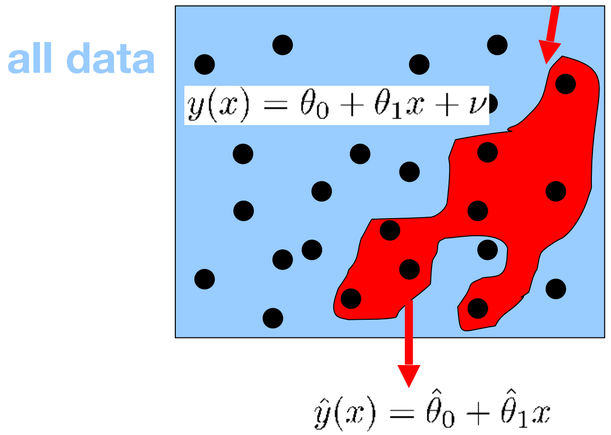
Linear classification

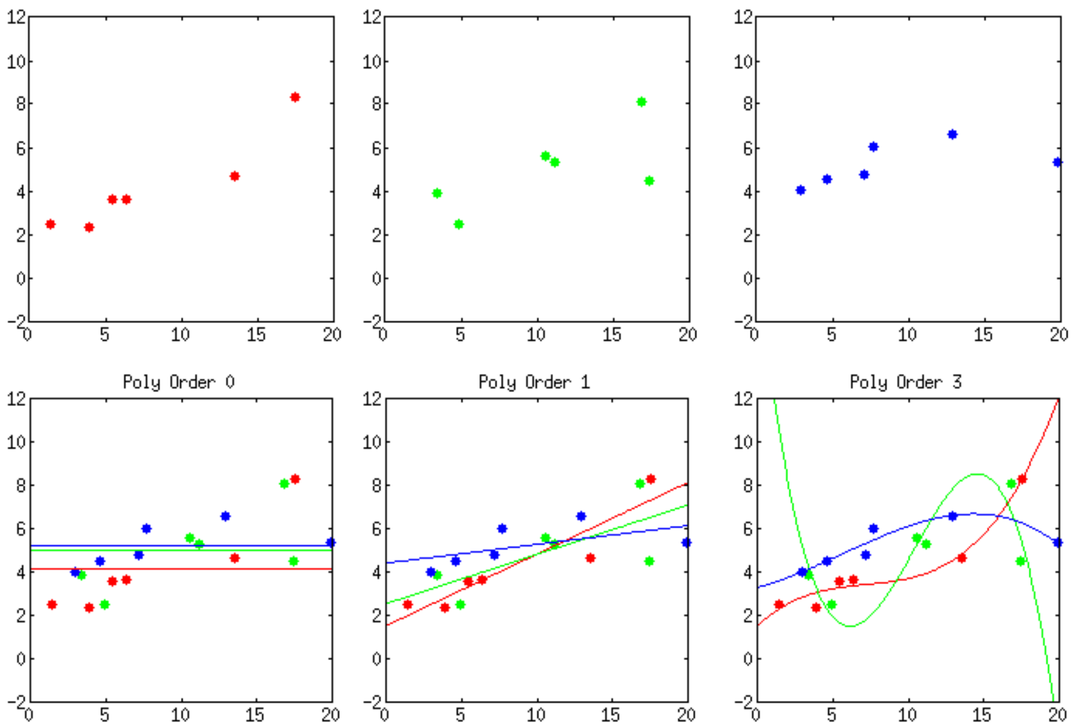
Inductive bias

- Inductive bias = assumptions we make to generalize to data we haven't seen
  - 10 data points suggest 9-degree polynomial, but we're “biased” towards linear
  - Examples: polynomials, smooth functions, neural network architecture, etc.
- Without any assumptions, there is no generalization
  - No Free Lunch Theorem = “Anything is possible” in the test data
- Occam's razor: prefer simpler explanations of the data

Bias vs. variance

- Imagine 3 universes → 3 datasets
- A simple model:
  - Poor prediction (on average across universes)
    - High bias
  - Doesn't vary much between universes
    - Low variance
- A complex model:
    - Low bias
    - High variance





Analyzing learning algorithms

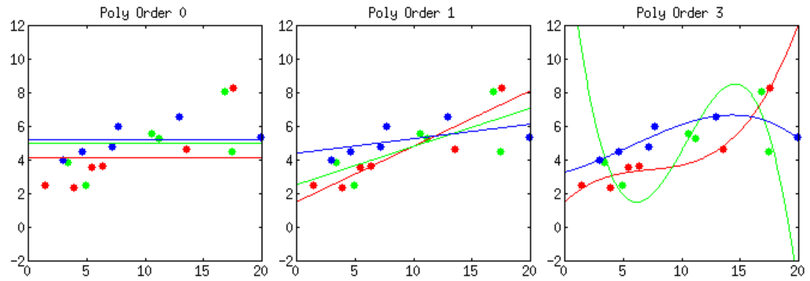
- Learning algorithm (incl. model class): $\mathscr{A} : \mathscr{D} \to \theta$
- How good is a model?
  - Test loss: $\mathscr{L}_\theta = \mathbb{E}_{x,y\sim p}[\ell(y, \hat{y}_\theta(x))]$
- How good is an algorithm?
  - Expected test loss over datasets: $\mathbb{E}_{\mathscr{D}}[\mathscr{L}_{\theta(\mathscr{D})}]$
  - We can estimate it with multiple datasets
  - We can analyze it theoretically if we make some assumptions

Bias–variance tradeoff

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

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

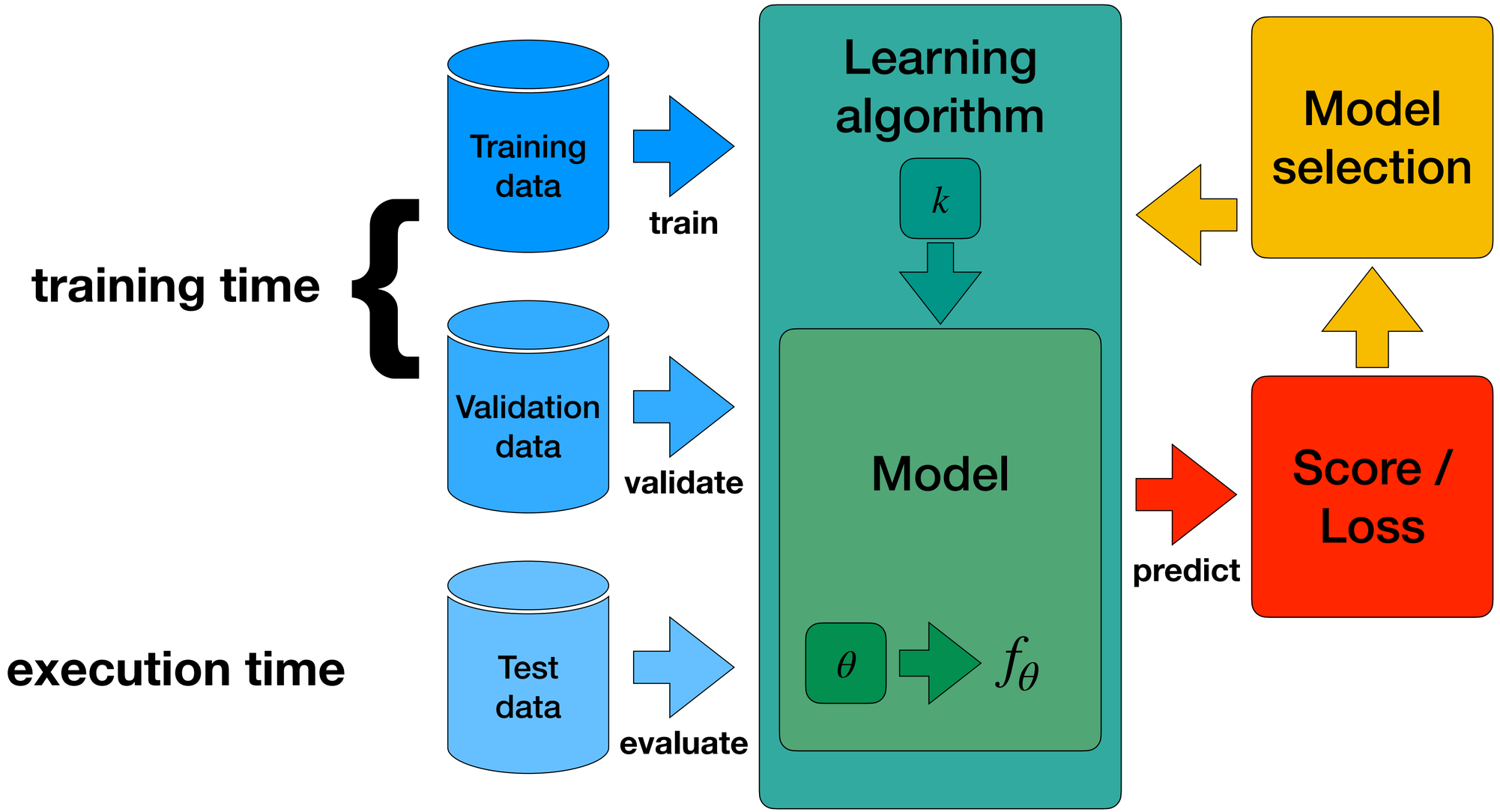
- Both components contribute equally to the quality of our algorithm
  - We can generally improve one at the expense of the other
    - Bias generally decreases with complexity
    - Variance generally increases with complexity



Variance and overfitting

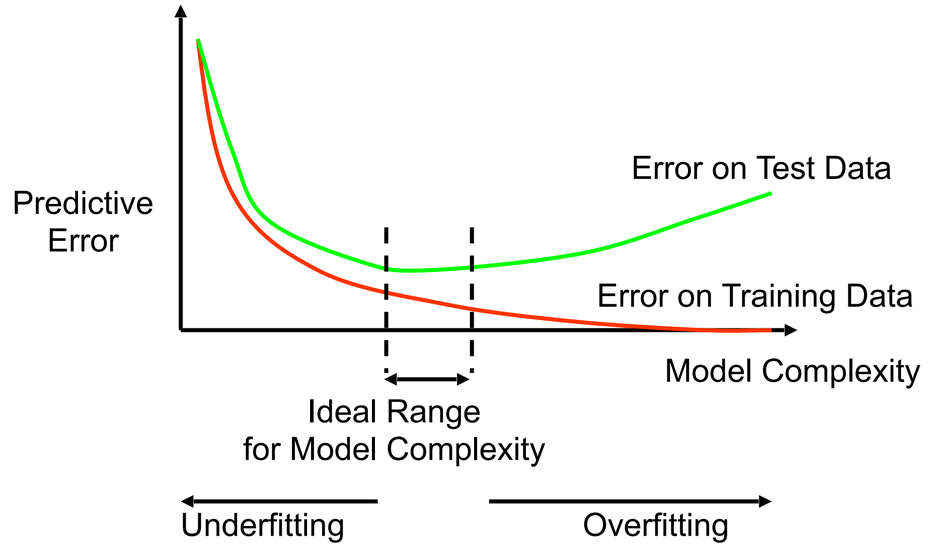
- Prediction that varies much with dataset = overfits to noise in training data
  - Rather than fitting the trend in the underlying distribution
  - Will perform poorly on test data
- How to select model complexity?
  - Model selection via validation

Model selection



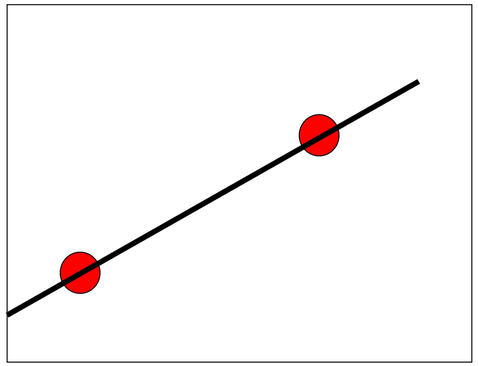
How to control model complexity?

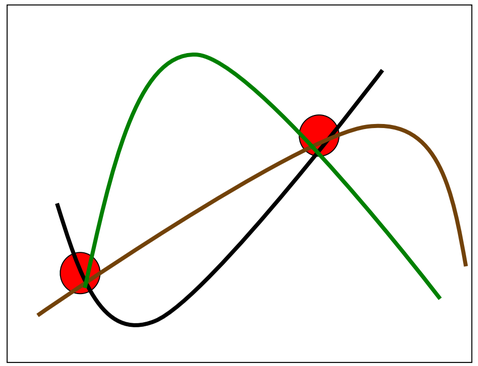
- To increase model complexity:
  - Add features, parameters
  - More on this later
- To decrease model complexity:
  - Remove features (feature selection)
  - Perform a part of training that attends less to noise (e.g. early stopping)
  - Regularization (up next)



Example: quadratic regression

- One linear model best fits two data points
- But infinitely many quadratic ones do
  - How to choose among them?
- For polynomials: reduce degree
- Generally: regularize
  - Add constraint / loss term to reduce sensitivity to noisy data
- Example: $\min_{\theta} \mathscr{L}_{\theta}$ s.t. $\|\theta\| \leq C$
  - Equivalently: $\min_{\theta} \mathscr{L}_{\theta} + \alpha\|\theta\|^2$





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 and Bayesian prediction

- Assume the data was generated using this process:
  - Parameter vector θ was sampled from a Gaussian: $\theta \sim \mathcal{N}(0, \alpha^{-1} I)$
  - Features X were sampled "somehow" (it won't matter)
  - Labels y are linear in X, but with Gaussian noise: $y = \theta^\intercal X + \epsilon \qquad \epsilon \sim \mathcal{N}(0, I)$
- What is the joint distribution $p(\theta, X, y)$?
  - $p(\theta, X, y) = p(\theta) p(X) p(y \mid \theta, X) = \mathcal{N}(\theta; 0, \alpha^{-1} I) p(X) \mathcal{N}(y - \theta^\intercal X; 0, I)$
  - $\log p(\theta, X, y) = \log p(X) - \frac{1}{2} \alpha^2 \|\theta\|^2 - \frac{1}{2} \|y - \theta^\intercal X\|^2 + \text{const}$
  - $p(\theta \mid X, y) = \mathcal{N}(\theta; y X^\intercal (X X^\intercal + \alpha I)^{-1}, (X X^\intercal + \alpha I)^{-1})$

MAP θ

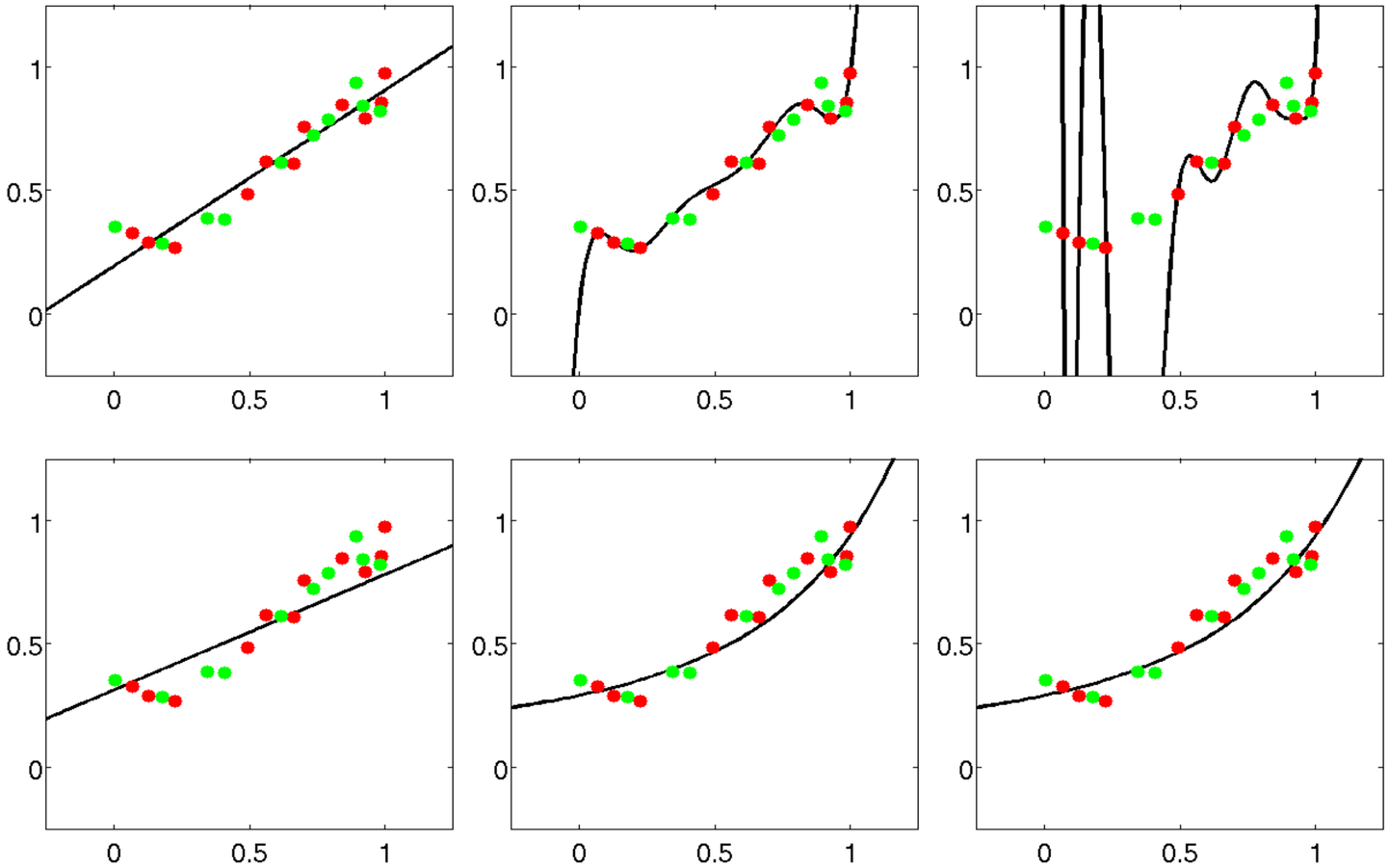
Regularization

- Comparing unregularized and regularized regression:

- $\alpha = 0$

  (Unreg.)

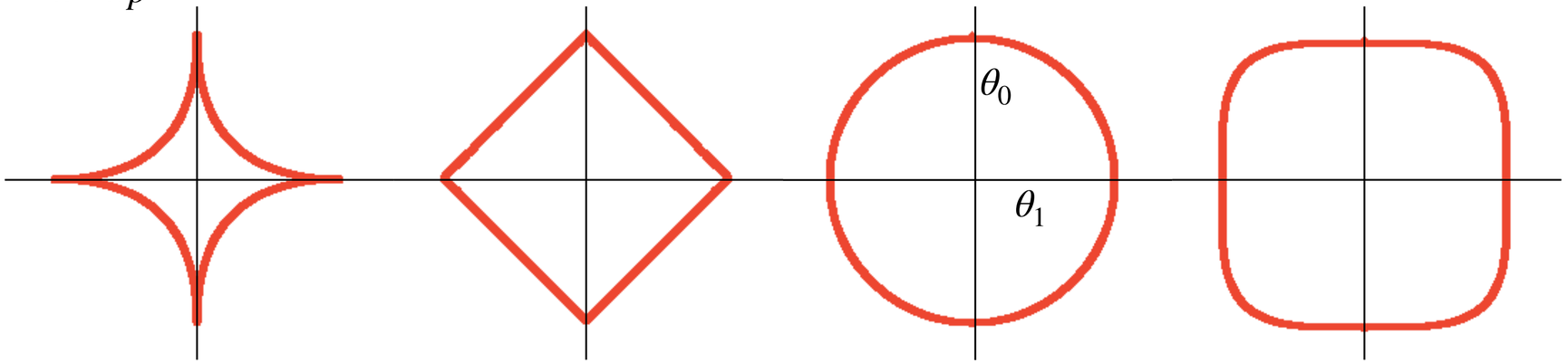
- $\alpha = 1$



L_p regularization

- Other popular regularizers are L_p norm: $\|\theta\|_p = \left(\sum_i |\theta_i|^p \right)^{\frac{1}{p}}$

- Isosurfaces: ($\|\theta\|_p = \text{const}$)

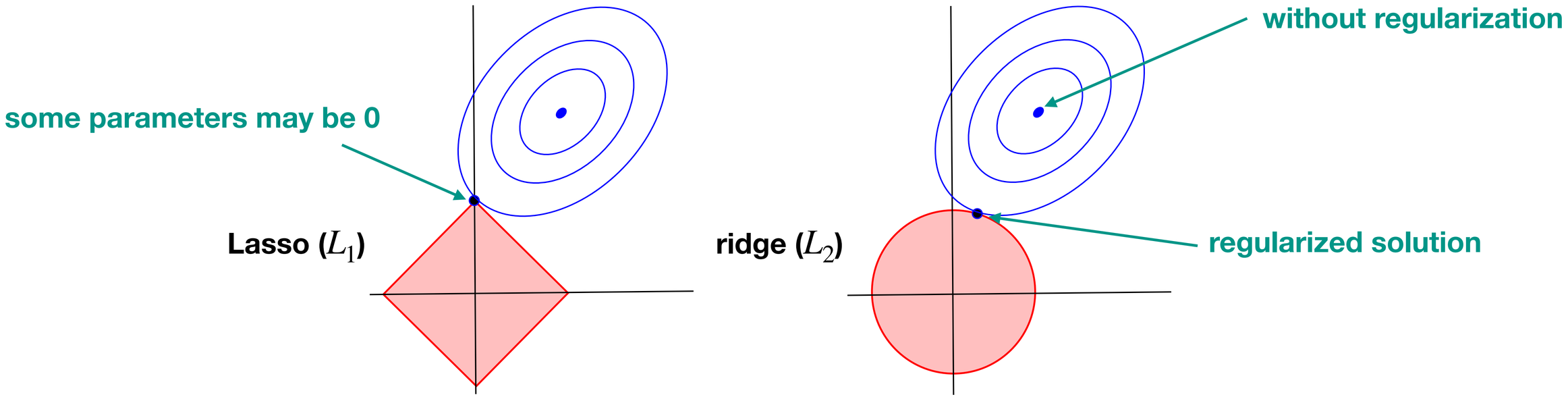


- $L_0 = \lim_{p \to 0} L_p$: number of nonzero parameters, natural notion of model complexity

- $L_\infty = \lim_{p \to \infty} L_p$: maximum parameter value

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



Today's lecture

Polynomial regression

Inductive bias and regularization

Cross-validation

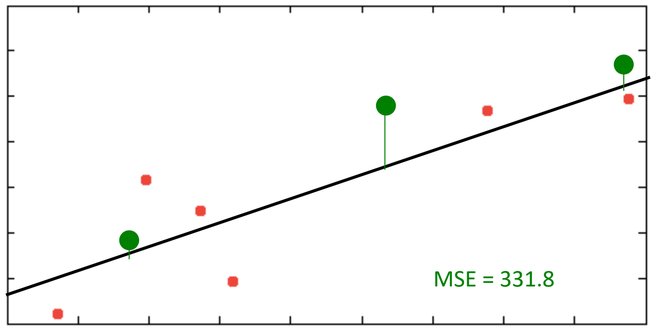
Linear classification

Validation

- To select model class / model hyper-parameters ϕ (e.g. polynomial degree)
  - Train models on training dataset: $\theta = \mathcal{A}_\phi(\mathcal{D}_{\text{training}})$
  - Evaluate models on validation dataset: $\mathcal{L} = \mathbb{E}_{x,y \sim \mathcal{D}_{\text{validation}}}[\ell_\theta(x, y)]$
- What if we don't get a validation set?
  - Split training set into training + validation

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?



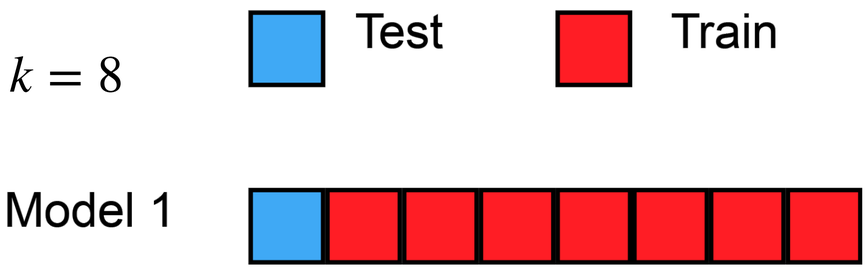
training data

validation data

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

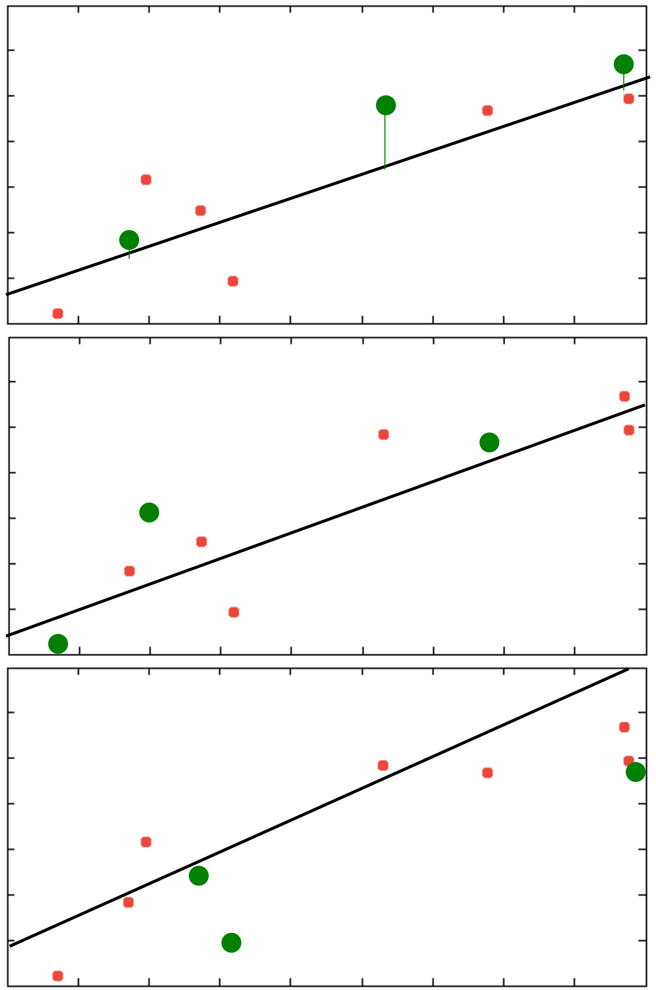
k-fold cross-validation method

- Randomly split the data into k disjoint sets
- For each of the k sets:
  - Hold it, train on the other $k-1$ sets
  - Validate on the held-out set
- Use average validation loss to select model hyper-parameters ϕ
- Train with selected ϕ on full data



k-fold cross-validation method

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



Split 1:
MSE = 331.8

Split 2:
MSE = 361.2

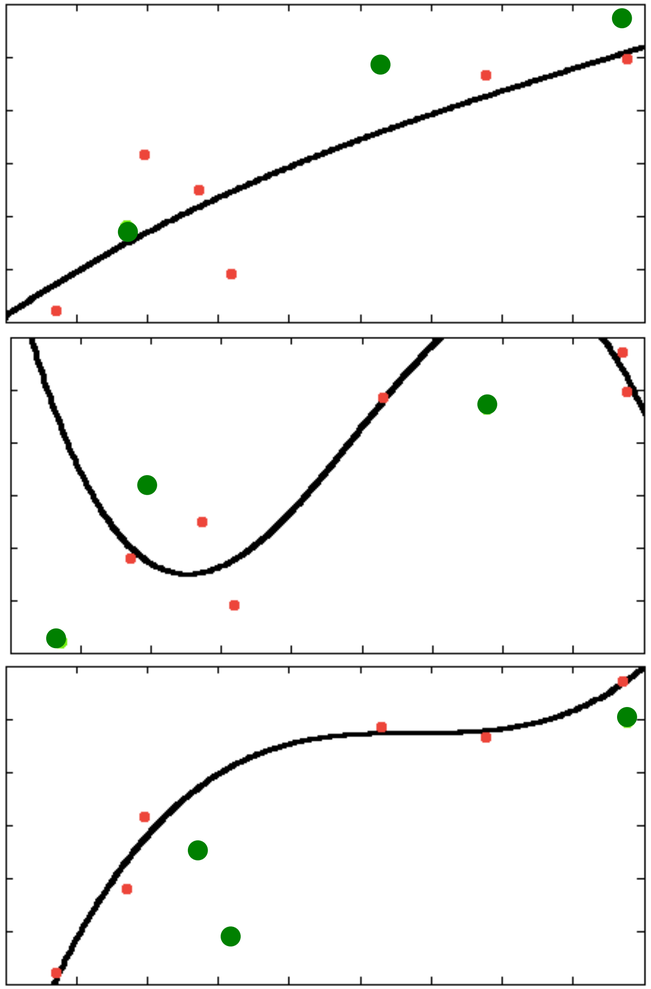
Split 3:
MSE = 669.8

3-Fold X-Val MSE
= 464.1

$x^{(i)}$	$y^{(i)}$
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

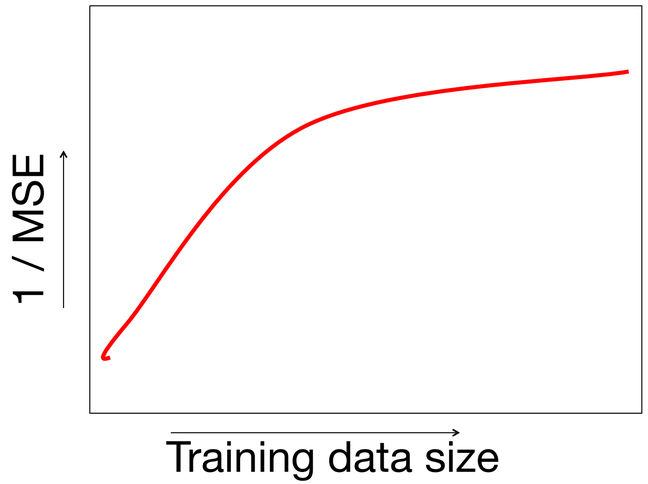
$x^{(i)}$	$y^{(i)}$
88	79
32	-2
27	30
68	73
7	-16
20	43
53	77
17	16
87	94

Cross-validation: considerations

- Trade off model training time with loss estimation accuracy
- Single held-out set: train on $m' < m$ data points, estimate loss on the rest
  - m must be large enough for both training and validation
  - We have an estimate of the final model performance
- k-fold XVal: split data into k disjoint sets, train on all but one used for validation
  - Computationally more expensive: training k models
  - Each validated model may be worse: trained on $m - \frac{m}{k}$ data points
  - But: estimate loss on more data, output model trained on all data
- LOO XVal: train on all but one data point, validate it, average this over all data points

Learning curves

- Plot performance (higher = better) as a function of training size
  - Assess impact of fewer data on performance
  - E.g., MSE0 – MSE for regression, or 1 – error rate for classification
- Performance (properly measured) should increase with training size
  - Should improve quickly when data is scarce, saturate when there's "enough"
  - May need to average over multiple experiments / trials / runs



Today's lecture

Polynomial regression

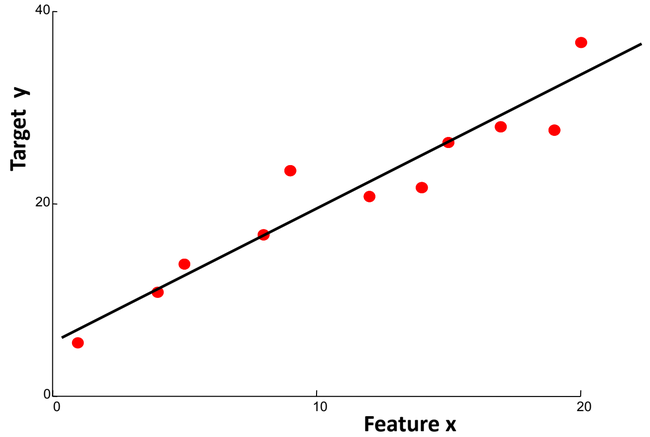
Inductive bias and regularization

Cross-validation

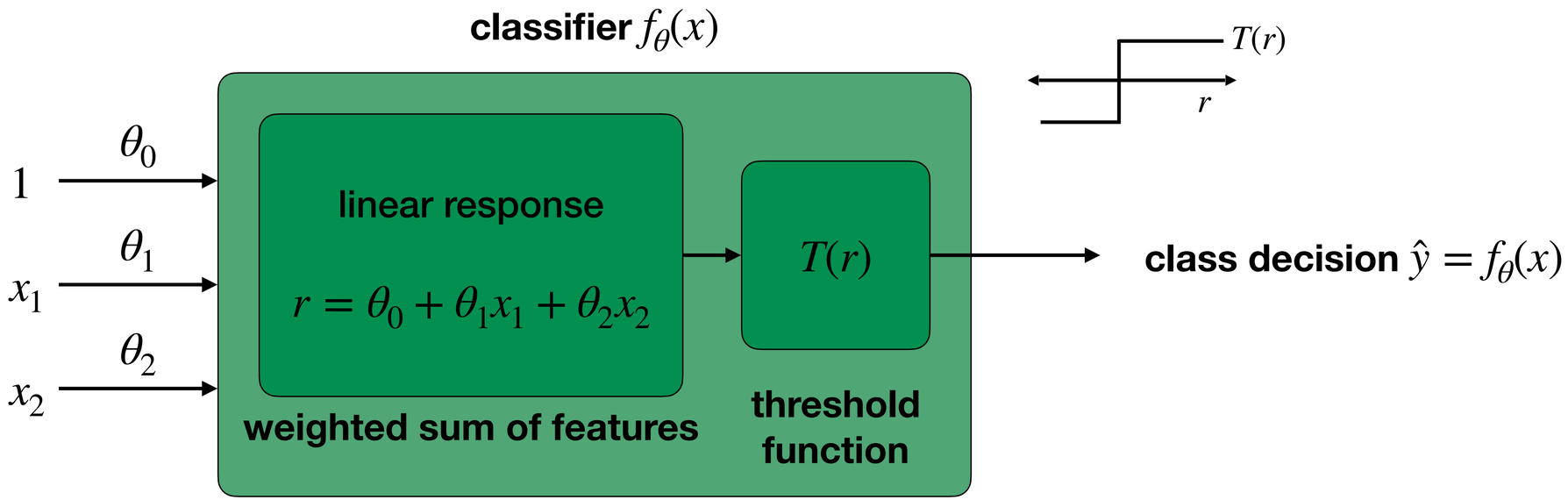
Linear classification

Linear regression vs. classification

- Regression:
  - Continuous target y
  - Predictor $\hat{y} = \theta^\intercal x$
- Classification:
  - Discrete label y
  - Classifier $\hat{y} = ?$



Perceptron

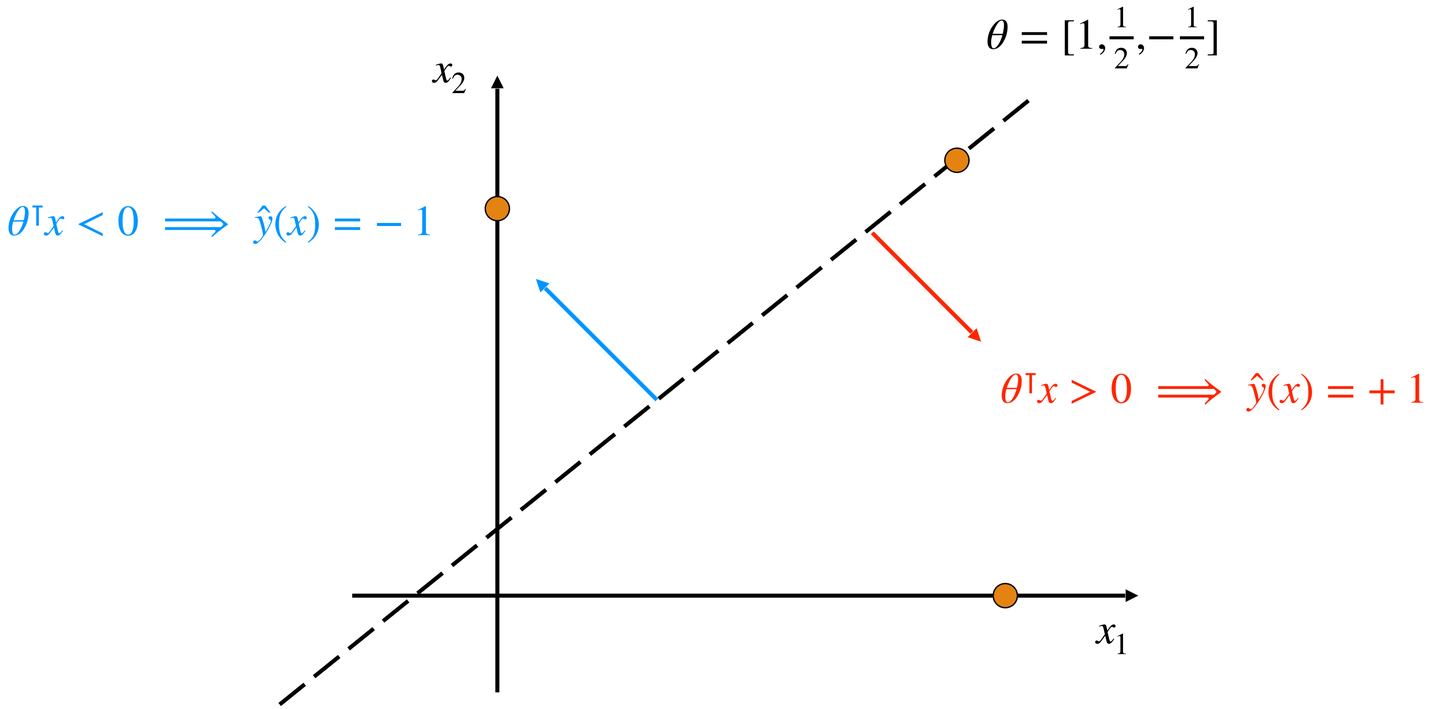



```
r = theta.T @ X       # compute linear response

y_hat = (r > 0)        # predict class 1 vs. 0

y_hat = 2*(r > 0) - 1  # predict class 1 vs. -1
```

Perceptron

- Perceptron = linear classifier
  - Parameters θ = weights (also denoted w)
  - Response = weighted sum of the features $r = \theta^\intercal x$
  - Prediction = thresholded response $\hat{y}(x) = T(r) = T(\theta^\intercal x)$
  - Decision function: $\hat{y}(x) = \begin{cases} +1 & \text{if } \theta^\intercal x > 0 \\ -1 & \text{otherwise} \end{cases}$ (for $T(r) = \text{sign}(r)$)
- Perceptron: a simple (vastly inaccurate) model of human neurons
  - Weights = “synapses”
  - Prediction = “neural firing”

Example


$\theta = [1, \frac{1}{2}, -\frac{1}{2}]$
x_2
$\theta^\intercal x < 0 \implies \hat{y}(x) = -1$
$\theta^\intercal x > 0 \implies \hat{y}(x) = +1$
x_1

Logistics

assignments

- Assignment 2 due next Tuesday, Jan 26

project

- Project guidelines on Canvas:
  - https://canvas.eee.uci.edu/courses/34497/pages/projects
- Team rosters due next Thursday, Jan 28 **on Canvas**