Announcements

Let $X \sim \mathcal{N}(\mu, \Sigma)$ where $X \in \mathbb{R}^d$

1. Let $Y = AX + b$. For what $\widetilde{\mu}, \widetilde{\Sigma}$ is $Y \sim \mathcal{N}(\widetilde{\mu}, \widetilde{\Sigma})$

2. Suppose I can generate independent Gaussians $Z \sim \mathcal{N}(0,1)$ (e.g., `numpy.random.randn`). How can I use this to generate X?

Regularization

Machine Learning – CSE546

Kevin Jamieson

University of Washington

October 10, 2016

Regularization in Linear Regression

Recall Least Squares: $\widehat{w}_{LS} = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2$

$$= \arg\min_w (\mathbf{y} - \mathbf{X}w)^T(\mathbf{y} - \mathbf{X}w)$$

when $(\mathbf{X}^T\mathbf{X})^{-1}$ exists.... $= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

Regularization in Linear Regression

Recall Least Squares: $\widehat{w}_{LS} = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2$

$$= \arg\min_w (\mathbf{y} - \mathbf{X}w)^T(\mathbf{y} - \mathbf{X}w)$$

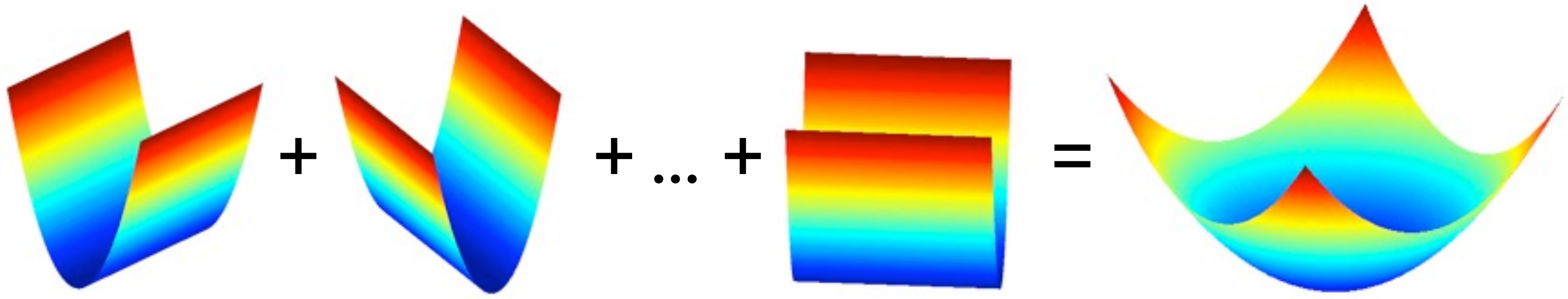
In general: $= \arg\min_w w^T(\mathbf{X}^T\mathbf{X})w - 2y^T\mathbf{X}w$

Regularization in Linear Regression

Recall Least Squares: $\widehat{w}_{LS} = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2$

$$= \arg\min_w (\mathbf{y} - \mathbf{X}w)^T(\mathbf{y} - \mathbf{X}w)$$

In general: $= \arg\min_w w^T(\mathbf{X}^T\mathbf{X})w - 2y^T\mathbf{X}w$



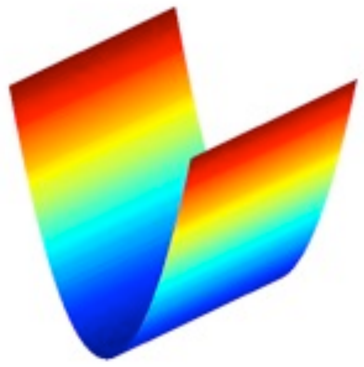
$$(y_1 - x_1^T w)^2 + (y_2 - x_2^T w)^2 + \cdots + (y_n - x_n^T w)^2 = \sum_{i=1}^n (y_i - x_i^T w)^2$$

What if $x_i \in \mathbb{R}^d$ and $d > n$?

Regularization in Linear Regression

Recall Least Squares: $\widehat{w}_{LS} = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2$

When $x_i \in \mathbb{R}^d$ and $d > n$ the objective function is flat in some directions:



Regularization in Linear Regression

Recall Least Squares: $\widehat{w}_{LS} = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2$

When $x_i \in \mathbb{R}^d$ and $d > n$ the objective function is flat in some directions:

Implies optimal solution is *underconstrained* and unstable due to lack of curvature*:*

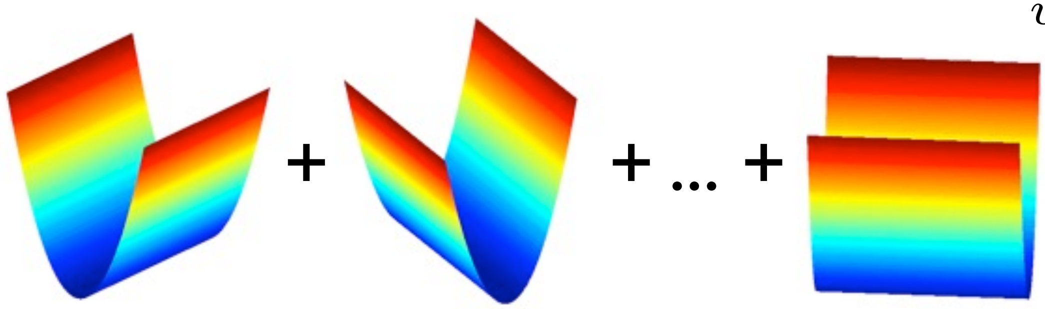
- small changes in training data result in large changes in solution
- often the *magnitudes* of *w* are "very large"

Regularization imposes "simpler" solutions by a "complexity" penalty

Ridge Regression

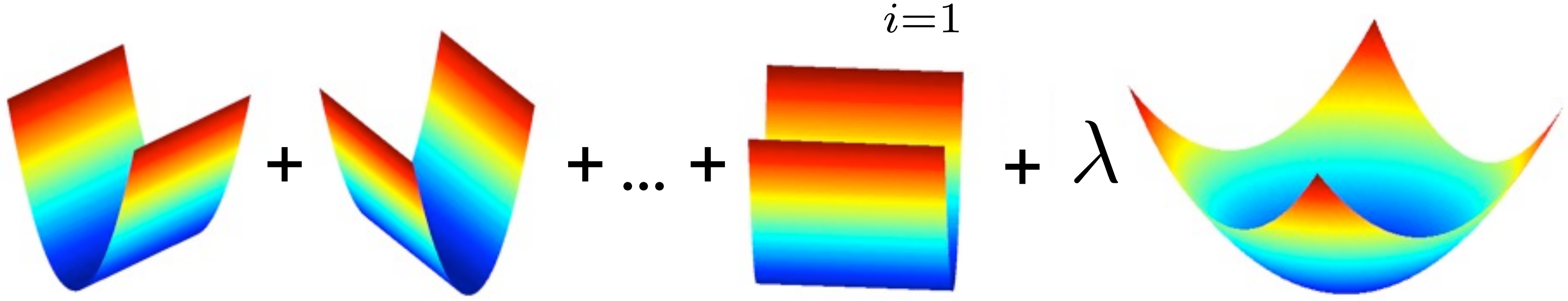
- Old Least squares objective:

$$\widehat{w}_{LS} = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2$$



- Ridge Regression objective:

$$\widehat{w}_{ridge} = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$



Minimizing the Ridge Regression Objective

$$\begin{aligned}\widehat{w}_{ridge} &= \arg\min_w \sum_{i=1}^n \big(y_i - (x_i^T w + b)\big)^2 + \lambda ||w||_2^2 \\ &= \arg\min_w ||\mathbf{y} - (\mathbf{X}w + \mathbf{1}b)||_2^2 + \lambda ||w||_2^2\end{aligned}$$

Shrinkage Properties

$$\widehat{w}_{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}$$

- If orthonormal features/basis: $\mathbf{X}^T\mathbf{X} = I$

Ridge Regression: Effect of Regularization

$$\widehat{w}_{ridge} = \arg\min_w \sum_{i=1}^n \left(y_i - (x_i^T w + b)\right)^2 + \lambda ||w||_2^2$$

- Solution is indexed by the regularization parameter λ
- Larger λ
- Smaller λ
- As λ → 0
- As λ →∞

Ridge Regression: Effect of Regularization

$$\mathcal{D} \overset{i.i.d.}{\sim} P_{XY}$$

$$\widehat{w}_{\mathcal{D},ridge}^{(\lambda)} = \arg\min_w \frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T w)^2 + \lambda ||w||_2^2$$

TRAIN error:

$$\frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T \widehat{w}_{\mathcal{D},ridge}^{(\lambda)})^2$$

TRUE error:

$$\mathbb{E}[(Y - X^T \widehat{w}_{\mathcal{D},ridge}^{(\lambda)})^2]$$

TEST error:

$$\mathcal{T} \overset{i.i.d.}{\sim} P_{XY}$$

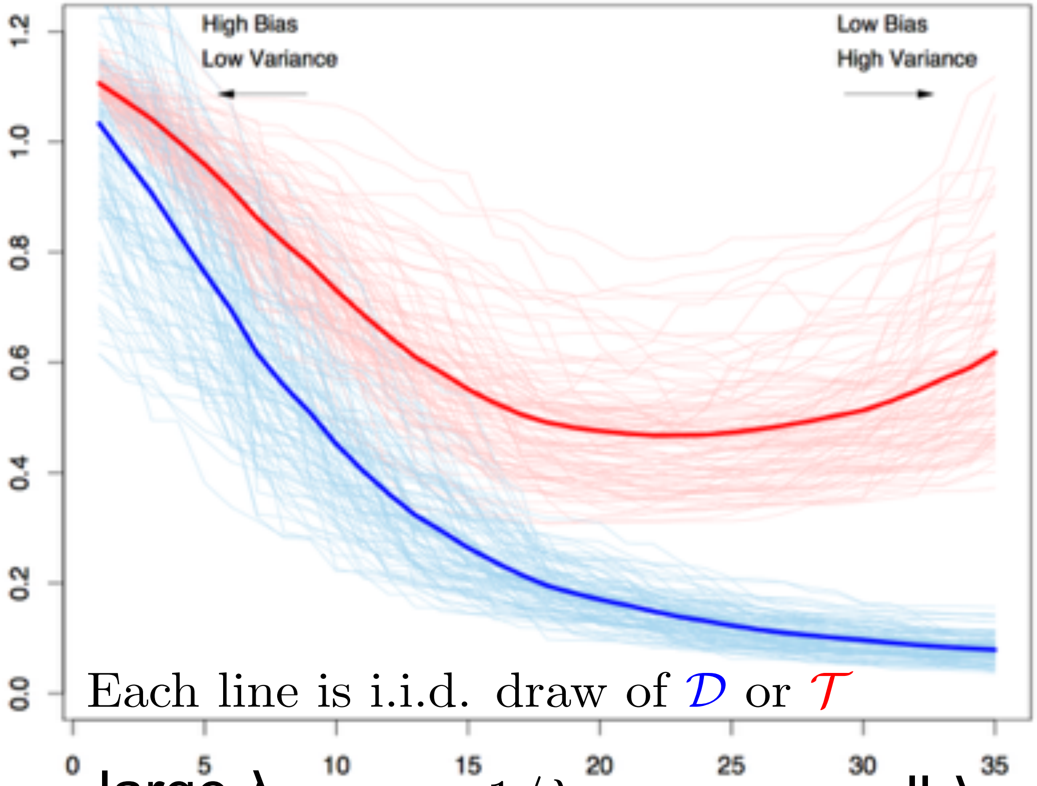
$$\frac{1}{|\mathcal{T}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T \widehat{w}_{\mathcal{D},ridge}^{(\lambda)})^2$$

$$\boxed{\text{Important: } \mathcal{D} \cap \mathcal{T} = \emptyset}$$

Ridge Regression: Effect of Regularization

$$\mathcal{D} \overset{i.i.d.}{\sim} P_{XY}$$

$$\widehat{w}^{(\lambda)}_{\mathcal{D},ridge} = \arg\min_w \frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T w)^2 + \lambda ||w||_2^2$$



Each line is i.i.d. draw of $\mathcal{D}$ or $\mathcal{T}$

large λ $\quad 1/\lambda \quad$ small λ

TRAIN error:

$$\frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T \widehat{w}^{(\lambda)}_{\mathcal{D},ridge})^2$$

TRUE error:

$$\mathbb{E}[(Y - X^T \widehat{w}^{(\lambda)}_{\mathcal{D},ridge})^2]$$

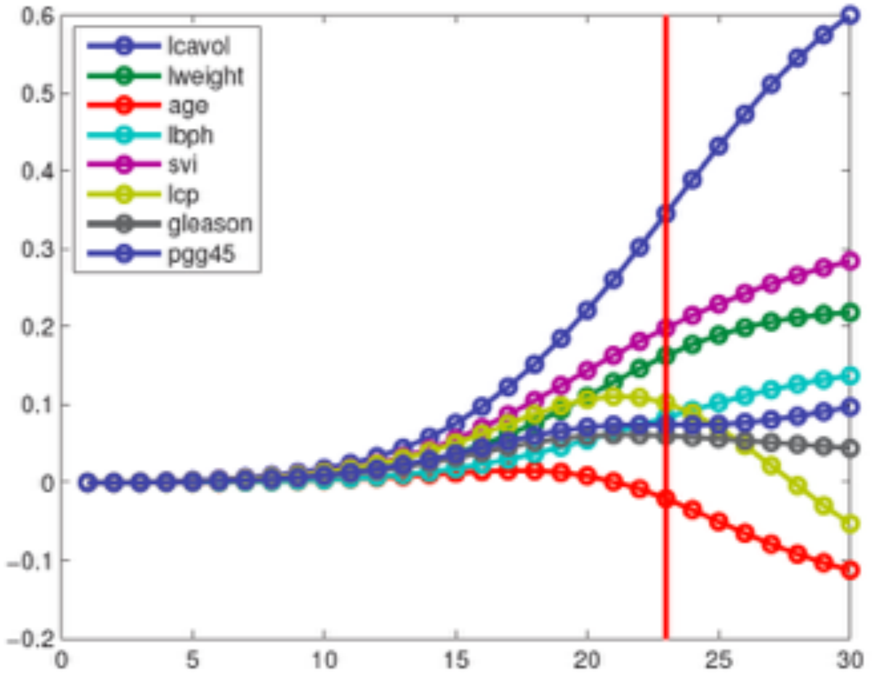
TEST error:

$$\mathcal{T} \overset{i.i.d.}{\sim} P_{XY}$$

$$\frac{1}{|\mathcal{T}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T \widehat{w}^{(\lambda)}_{\mathcal{D},ridge})^2$$

$$\boxed{\text{Important: } \mathcal{D} \cap \mathcal{T} = \emptyset}$$

Ridge Coefficient Path



From
Kevin Murphy
textbook

- Typical approach: select λ using cross validation, up next

What you need to know…

- Regularization
  - Penalizes for complex models
- Ridge regression
  - L_2 penalized least-squares regression
  - Regularization parameter trades off model complexity with training error

Cross-Validation

Machine Learning – CSE546

Kevin Jamieson

University of Washington

October 10, 2016

How… How… How????????

- *How do we pick the regularization constant λ…*
- *How do we pick the number of basis functions…*

- We could use the test data, but…

How… How… How???????

- *How do we pick the regularization constant λ…*
- *How do we pick the number of basis functions…*

- We could use the test data, but…

- Never ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever train on the test data

(LOO) Leave-one-out cross validation

- Consider a **validation set with 1 example**:
  - D – training data
  - $D\backslash j$ – training data with j th data point $(\mathbf{x}_j, \mathbf{y}_j)$ moved to validation set
- **Learn classifier $f_{D\backslash j}$ with $D\backslash j$ dataset**
- **Estimate true error** as squared error on predicting $\mathbf{y}_j$:
  - Unbiased estimate of $\text{error}_{\text{true}}(\boldsymbol{f}_{D\backslash j})$!

(LOO) Leave-one-out cross validation

- Consider a **validation set with 1 example**:
  - *D* – training data
  - *D*\j – training data with *j* th data point ($\mathbf{x}_j$,$\mathbf{y}_j$) moved to validation set
- **Learn classifier $f_{D\setminus j}$ with *D*\j dataset**
- **Estimate true error** as squared error on predicting $\mathbf{y}_\mathbf{j}$:
  - Unbiased estimate of $\text{error}_{\text{true}}(f_{D\setminus j})$!

- **LOO cross validation**: Average over all data points *j*:
  - **For each data point you leave out, learn a new classifier $f_{D\setminus j}$**
  - **Estimate error** as:

$$\text{error}_{LOO} = \frac{1}{n}\sum_{j=1}^{n}(y_j - f_{\mathcal{D}\setminus j}(x_j))^2$$

LOO cross validation is (almost) unbiased estimate of true error of h_D!

- When computing **LOOCV error, we only use *N-1* data points**
  - So it's not estimate of true error of learning with *N* data points
  - Usually pessimistic, though – learning with less data typically gives worse answer

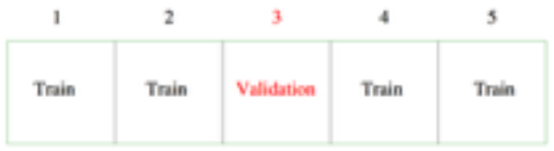
- **LOO is almost unbiased**! **Use LOO error for model selection!!!**
  - **E.g., picking λ**

Computational cost of LOO

- Suppose you have 100,000 data points
- You implemented a great version of your learning algorithm
    - Learns in only 1 second
- Computing LOO will take about 1 day!!!

Use *k*-fold cross validation

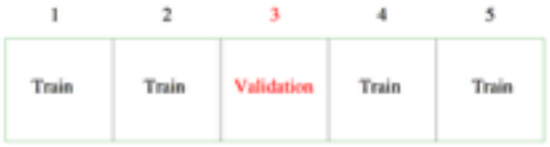
- Randomly **divide training data into *k* equal parts**
  - $D_1, \ldots, D_k$
- For each *i*
  - **Learn classifier $f_{D \backslash D_i}$ using data point not in D_i**
  - **Estimate error of $f_{D \backslash D_i}$ on validation set D_i:**



$$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j, y_j) \in \mathcal{D}_i} (y_j - f_{\mathcal{D} \backslash \mathcal{D}_i}(x_j))^2$$

Use *k*-fold cross validation

- Randomly **divide training data into *k* equal parts**
  - $D_1, \ldots, D_k$
- For each *i*
  - **Learn classifier $f_{D \backslash D_i}$ using data point not in D_i**
  - **Estimate error of $f_{D \backslash D_i}$ on validation set D_i:**



$$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j, y_j) \in \mathcal{D}_i} (y_j - f_{\mathcal{D} \backslash \mathcal{D}_i}(x_j))^2$$

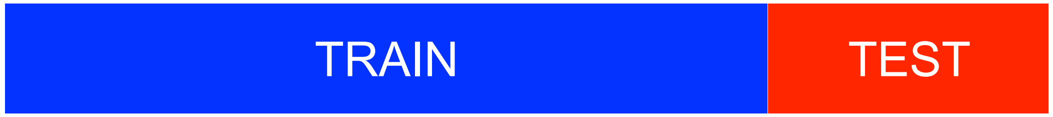
- ***k*-fold cross validation error is average** over data splits:

$$error_{k-fold} = \frac{1}{k} \sum_{i=1}^{k} error_{\mathcal{D}_i}$$

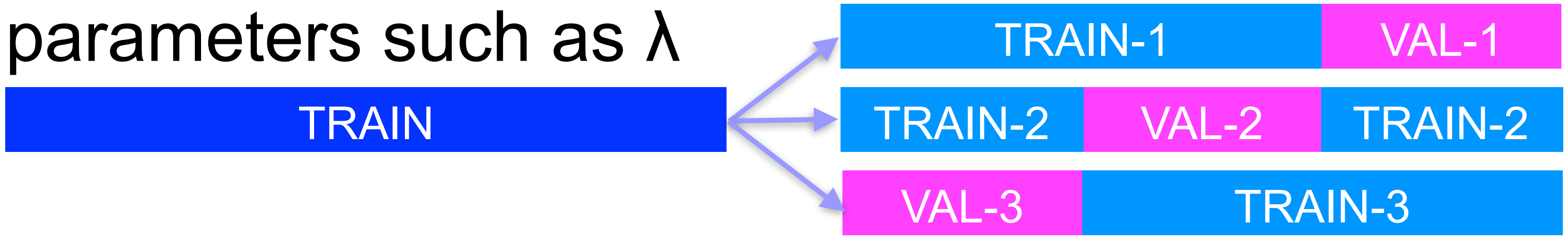
- *k*-fold cross validation properties:
  - **Much faster to compute** than LOO
  - **More (pessimistically) biased** – using much less data, only *n(k-1)/k*
  - Usually, **k = 10**

Recap

- Given a dataset, begin by splitting into



- **Model selection**: Use k-fold cross-validation on TRAIN to train predictor and choose magic parameters such as λ



- **Model assessment**: Use TEST to assess the accuracy of the model you output
  - Never ever ever ever ever train or choose parameters based on the test data

Example

- Given 10,000-dimensional data and n examples, we pick a subset of 50 dimensions that have the highest correlation with labels in the training set:

$$\text{50 indices j that have largest } \frac{|\sum_{i=1}^n x_{i,j} y_i|}{\sqrt{\sum_{i=1}^n x_{i,j}^2}}$$

- After picking our 50 features, we then use CV to train ridge regression with regularization λ
- What's wrong with this procedure?

Bootstrap

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 10, 2016

Limitations of CV

- An 80/20 split throws out a relatively large amount of data if only have, say, 20 examples.
- Test error is informative, but how accurate is this number? (e.g., 3/5 heads vs. 30/50)
- How do I get confidence intervals on statistics like the median or variance of a distribution?
- Instead of the error for the entire dataset, what if I want to study the error for a *particular example* x?

Limitations of CV

- An 80/20 split throws out a relatively large amount of data if only have, say, 20 examples.
- Test error is informative, but how accurate is this number? (e.g., 3/5 heads vs. 30/50)
- How do I get confidence intervals on statistics like the median or variance of a distribution?
- Instead of the error for the entire dataset, what if I want to study the error for a *particular example* x?

The Bootstrap: Developed by Efron in 1979.

"The most important innovation in statistics of the last 40 years"
— famous ML researcher and statistician, 2015

Bootstrap: basic idea

Given dataset drawn iid samples with CDF F_Z:

$$\mathcal{D} = \{z_1, \ldots, z_n\} \overset{i.i.d.}{\sim} F_Z$$

We compute a *statistic* of the data to get: $\widehat{\theta} = t(\mathcal{D})$

Bootstrap: basic idea

Given dataset drawn iid samples with CDF F_Z:

$$\mathcal{D} = \{z_1, \ldots, z_n\} \overset{i.i.d.}{\sim} F_Z$$

We compute a *statistic* of the data to get: $\widehat{\theta} = t(\mathcal{D})$

For b=1,…,B define the *b*th ***bootstrapped*** dataset as drawing *n* samples **with replacement** from *D*

$$\mathcal{D}^{*b} = \{z_1^{*b}, \ldots, z_n^{*b}\} \overset{i.i.d.}{\sim} \widehat{F}_{Z,n}$$

and the *b*th bootstrapped statistic as: $\theta^{*b} = t(\mathcal{D}^{*b})$

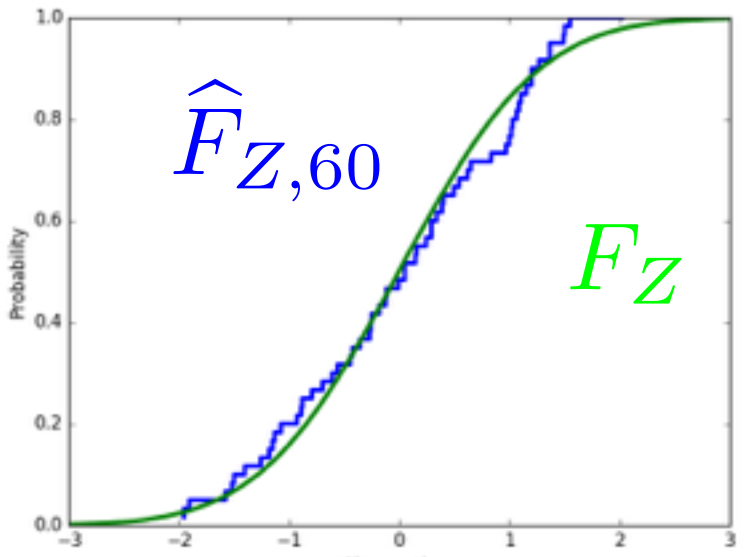
Bootstrap: basic idea

Given dataset drawn iid samples with CDF F_Z:

$$\mathcal{D} = \{z_1, \dots, z_n\} \overset{i.i.d.}{\sim} F_Z \qquad \widehat{\theta} = t(\mathcal{D})$$

For b=1,…,B, samples sampled **with replacement** from *D*

$$\mathcal{D}^{*b} = \{z_1^{*b}, \dots, z_n^{*b}\} \overset{i.i.d.}{\sim} \widehat{F}_{Z,n} \qquad \theta^{*b} = t(\mathcal{D}^{*b})$$



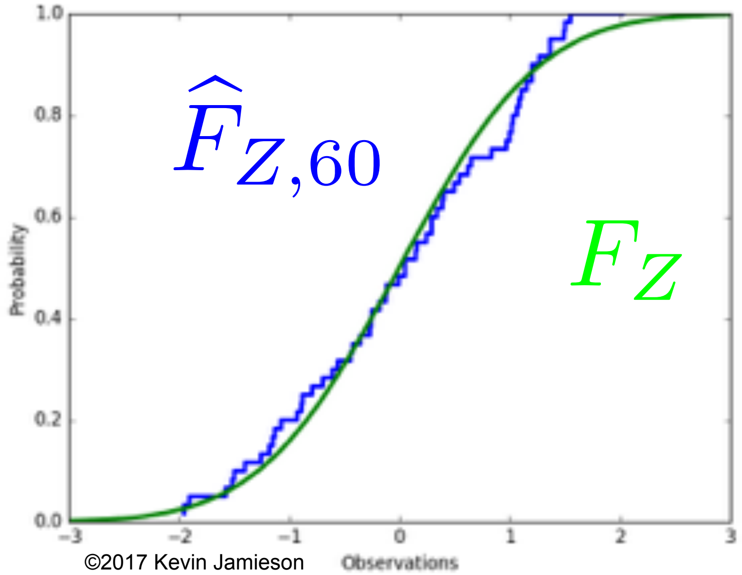
Bootstrap: basic idea

Given dataset drawn iid samples with CDF F_Z:

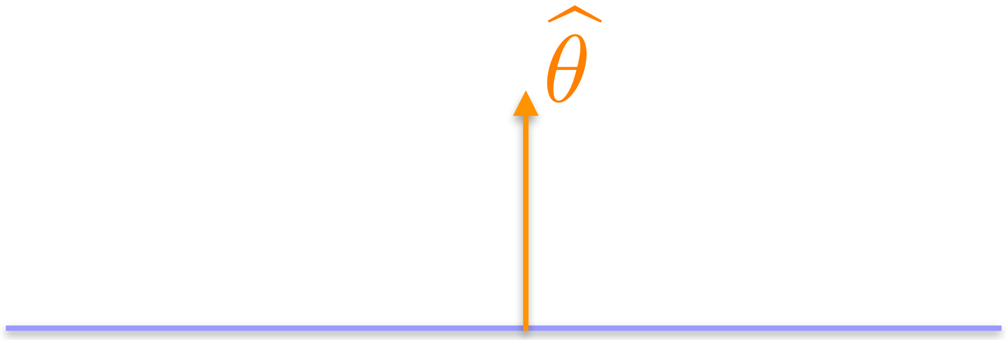
$$\mathcal{D} = \{z_1, \ldots, z_n\} \overset{i.i.d.}{\sim} F_Z \qquad \widehat{\theta} = t(\mathcal{D})$$

For b=1,…,B, samples sampled **with replacement** from *D*

$$\mathcal{D}^{*b} = \{z_1^{*b}, \ldots, z_n^{*b}\} \overset{i.i.d.}{\sim} \widehat{F}_{Z,n} \qquad \theta^{*b} = t(\mathcal{D}^{*b})$$



$$\boxed{\sup_x |\widehat{F}_n(x) - F(x)| \to 0 \quad \text{as } n \to \infty}$$

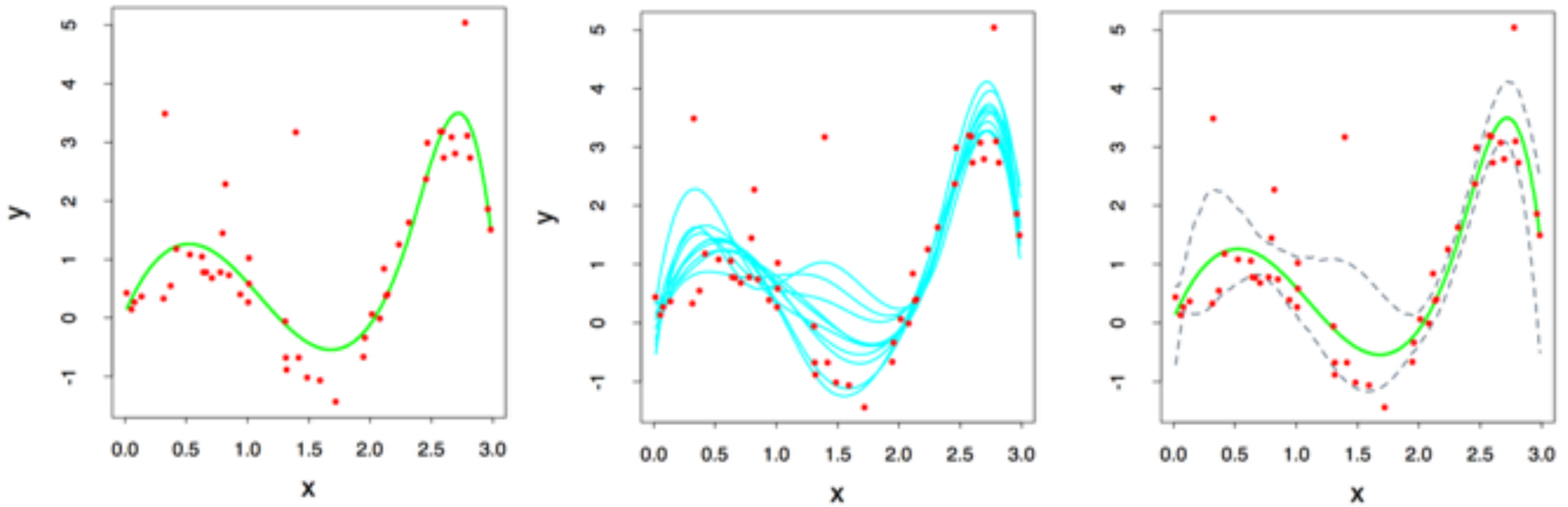


Applications

Common applications of the bootstrap:

- Estimate parameters that escape simple analysis like the variance or median of an estimate
- Confidence intervals
- Estimates of error for a particular example:

$\mathcal{D}$ $\widehat{\theta}$ θ^{*b} for $b = 1, \dots, 10$ 95% confidence interval



Figures from Hastie et al

Takeaways

Advantages:

- Bootstrap is **very** generally applicable. Build a confidence interval around ***anything***
- **Very** simple to use
- Appears to give meaningful results even when the amount of data is very small
- Very strong **asymptotic theory** (as num. examples goes to infinity)

Takeaways

Advantages:

- Bootstrap is **very** generally applicable. Build a confidence interval around ***anything***
- **Very** simple to use
- Appears to give meaningful results even when the amount of data is very small
- Very strong **asymptotic theory** (as num. examples goes to infinity)

Disadvantages

- Very few meaningful finite-sample guarantees
- Potentially **computationally intensive**
- Reliability relies on test statistic and rate of convergence of empirical CDF to true CDF, which is unknown
- Poor performance on “extreme statistics” (e.g., the max)

Not perfect, but better than nothing.

Recap

- Learning is…
  - Collect some data
    - E.g., housing info and sale price
  - Randomly split dataset into TRAIN, VAL, and TEST
    - E.g., 80%, 10%, and 10%, respectively
  - Choose a hypothesis class or model
    - E.g., linear with non-linear transformations
  - Choose a loss function
    - E.g., least squares with ridge regression penalty on TRAIN
  - Choose an optimization procedure
    - E.g., set derivative to zero to obtain estimator, cross-validation on VAL to pick num. features and amount of regularization
  - Justifying the accuracy of the estimate
    - E.g., report TEST error with Bootstrap confidence interval