Warm up

Fix any $a, b, c > 0$.

1. What is the $x \in \mathbb{R}$ that minimizes $ax^2 + bx + c$

2. What is the $x \in \mathbb{R}$ that minimizes $\max\{-ax + b, cx\}$

Overfitting

Machine Learning – CSE546
Kevin Jamieson
University of Washington

Oct 9, 2018

Bias-Variance Tradeoff

- Choice of hypothesis class introduces learning bias
  - More complex class → less bias
  - More complex class → more variance
- But in practice??
- Before we saw how increasing the feature space can increase the complexity of the learned estimator:

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \subset \ldots$$

$$\widehat{f}_{\mathcal{D}}^{(k)} = \arg \min_{f \in \mathcal{F}_k} \frac{1}{|\mathcal{D}|} \sum_{(x_i, y_i) \in \mathcal{D}} (y_i - f(x_i))^2$$

Complexity grows as k grows

Training set error as a function of model complexity

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \subset \dots \qquad \mathcal{D} \overset{i.i.d.}{\sim} P_{XY}$$

$$\widehat{f}_{\mathcal{D}}^{(k)} = \arg\min_{f \in \mathcal{F}_k} \frac{1}{|\mathcal{D}|} \sum_{(x_i, y_i) \in \mathcal{D}} (y_i - f(x_i))^2$$

TRAIN error:

$$\frac{1}{|\mathcal{D}|} \sum_{(x_i, y_i) \in \mathcal{D}} (y_i - \widehat{f}_{\mathcal{D}}^{(k)}(x_i))^2$$

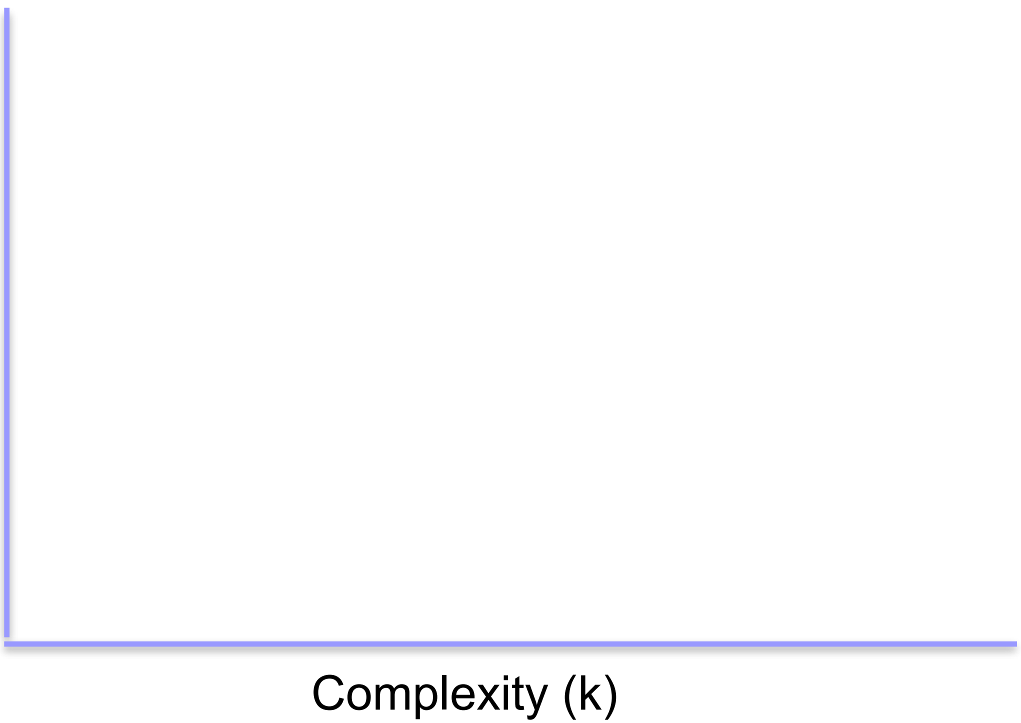
TRUE error:

$$\mathbb{E}_{XY}[(Y - \widehat{f}_{\mathcal{D}}^{(k)}(X))^2]$$

Training set error as a function of model complexity

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \subset \dots \quad \mathcal{D} \overset{i.i.d.}{\sim} P_{XY}$$

$$\widehat{f}_{\mathcal{D}}^{(k)} = \arg\min_{f \in \mathcal{F}_k} \frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - f(x_i))^2$$



TRAIN error:

$$\frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - \widehat{f}_{\mathcal{D}}^{(k)}(x_i))^2$$

TRUE error:

$$\mathbb{E}_{XY}[(Y - \widehat{f}_{\mathcal{D}}^{(k)}(X))^2]$$

TEST error:

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

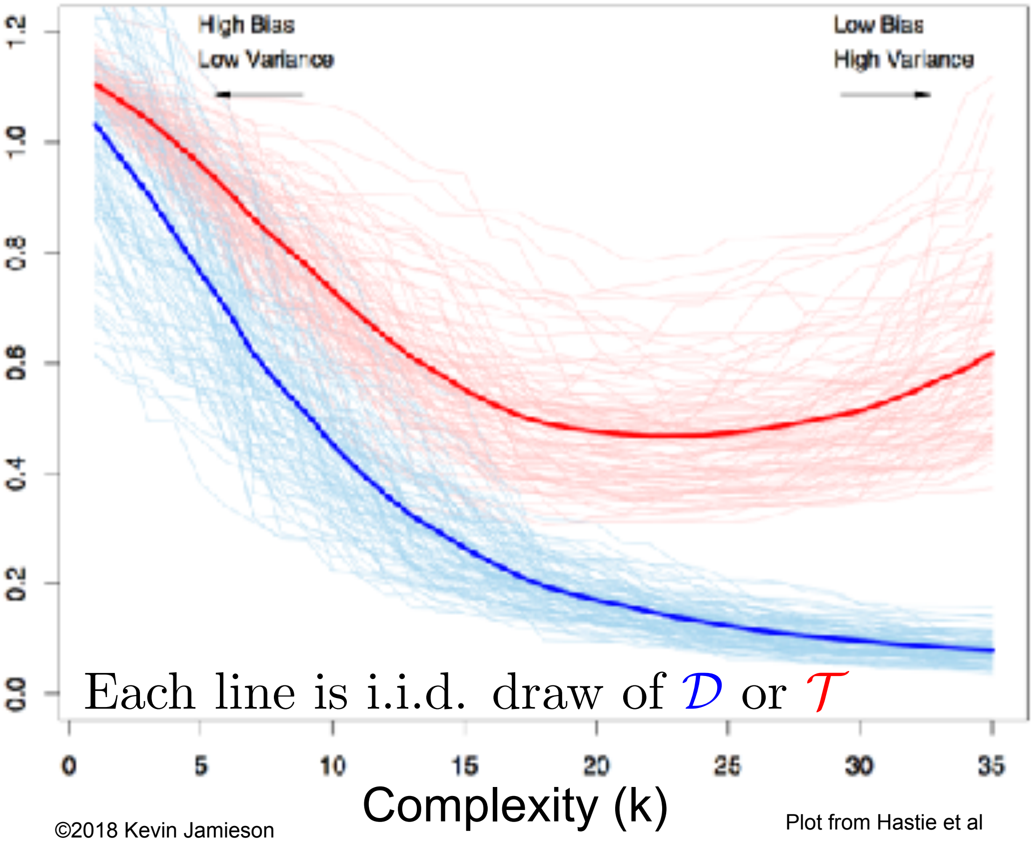
$$\frac{1}{|\mathcal{T}|} \sum_{(x_i,y_i)\in\mathcal{T}} (y_i - \widehat{f}_{\mathcal{D}}^{(k)}(x_i))^2$$

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

Training set error as a function of model complexity

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \subset \ldots \qquad \mathcal{D} \overset{i.i.d.}{\sim} P_{XY}$$

$$\widehat{f}_{\mathcal{D}}^{(k)} = \arg\min_{f \in \mathcal{F}_k} \frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - f(x_i))^2$$



Plot from Hastie et al

TRAIN error:

$$\frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - \widehat{f}_{\mathcal{D}}^{(k)}(x_i))^2$$

TRUE error:

$$\mathbb{E}_{XY}[(Y - \widehat{f}_{\mathcal{D}}^{(k)}(X))^2]$$

TEST error:

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

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

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

Training set error as a function of model complexity

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \subset \dots \qquad \mathcal{D} \overset{i.i.d.}{\sim} P_{XY}$$

$$\widehat{f}_{\mathcal{D}}^{(k)} = \arg\min_{f \in \mathcal{F}_k} \frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - f(x_i))^2$$

TRAIN error:

$$\frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - \widehat{f}_{\mathcal{D}}^{(k)}(x_i))^2$$

TRUE error:

$$\mathbb{E}_{XY}[(Y - \widehat{f}_{\mathcal{D}}^{(k)}(X))^2]$$

TEST error:

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

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

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

TRAIN error is **optimistically biased** because it is evaluated on the data it trained on. **TEST error** is **unbiased** only if *T* is never used to train the model or even pick the complexity k.

Test set error

- Given a dataset, **randomly** split it into two parts:
  - Training data: $\mathcal{D}$
  - Test data: $\mathcal{T}$

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

- Use training data to learn predictor
  - e.g., $\frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - \widehat{f}_{\mathcal{D}}^{(k)}(x_i))^2$
  - use training data to pick complexity k
- Use test data to report predicted performance

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

Regularization

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 9, 2016

Regularization in Linear Regression

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

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:

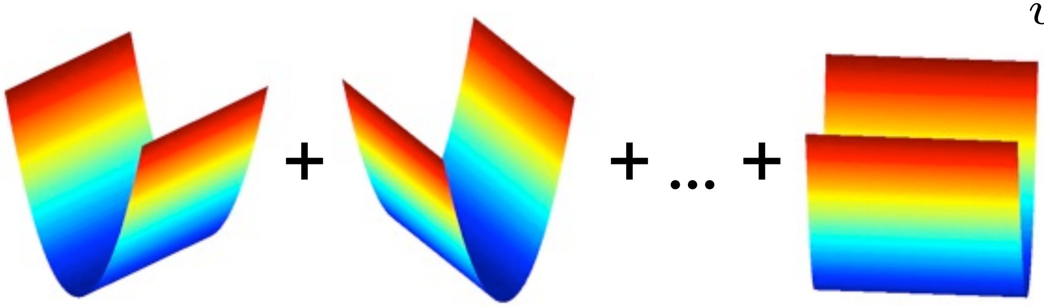
- small changes in training data result in large changes in solution
- often the *magnitudes* of *w* are "very large"

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

Ridge Regression

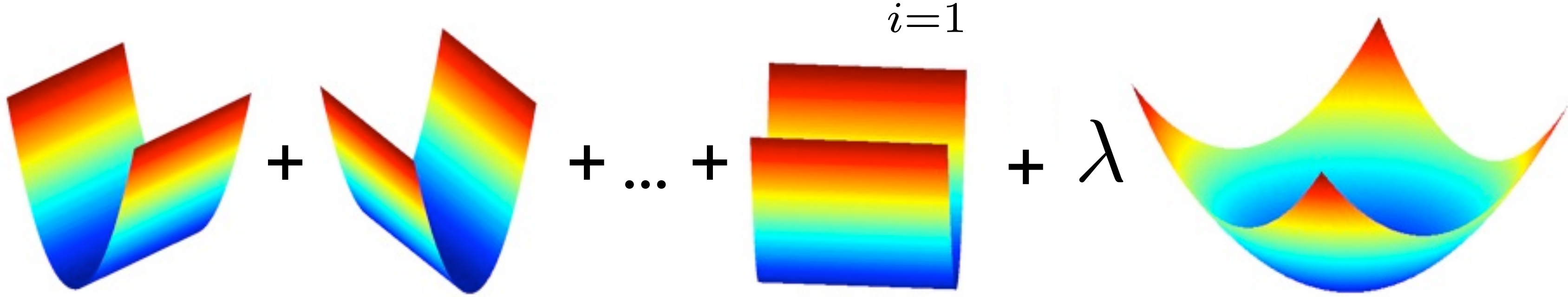
- Old Least squares objective:

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



- Ridge Regression objective:

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



Shrinkage Properties

$\epsilon \sim \mathcal{N}(0, \sigma^2 I)$

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

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

- ***Assume:*** $\mathbf{X}^T\mathbf{X} = nI$ ***and*** $\mathbf{y} = \mathbf{X}w + \boldsymbol{\epsilon}$

$$\begin{aligned}\widehat{w}_{ridge} &= (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T(\mathbf{X}w + \boldsymbol{\epsilon}) \\ &= \frac{n}{n+\lambda} w + \frac{1}{n+\lambda}\mathbf{X}^T\boldsymbol{\epsilon}\end{aligned}$$

$$\mathbb{E}\|\widehat{w}_{ridge} - w\|^2 = \frac{\lambda^2}{(n+\lambda)^2}\|w\|^2 + \frac{dn\sigma^2}{(n+\lambda)^2}$$

$$\lambda^* = \frac{d\sigma^2}{\|w\|^2}$$

Ridge Regression: Effect of Regularization

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

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

Ridge Regression: Effect of Regularization

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

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

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:

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

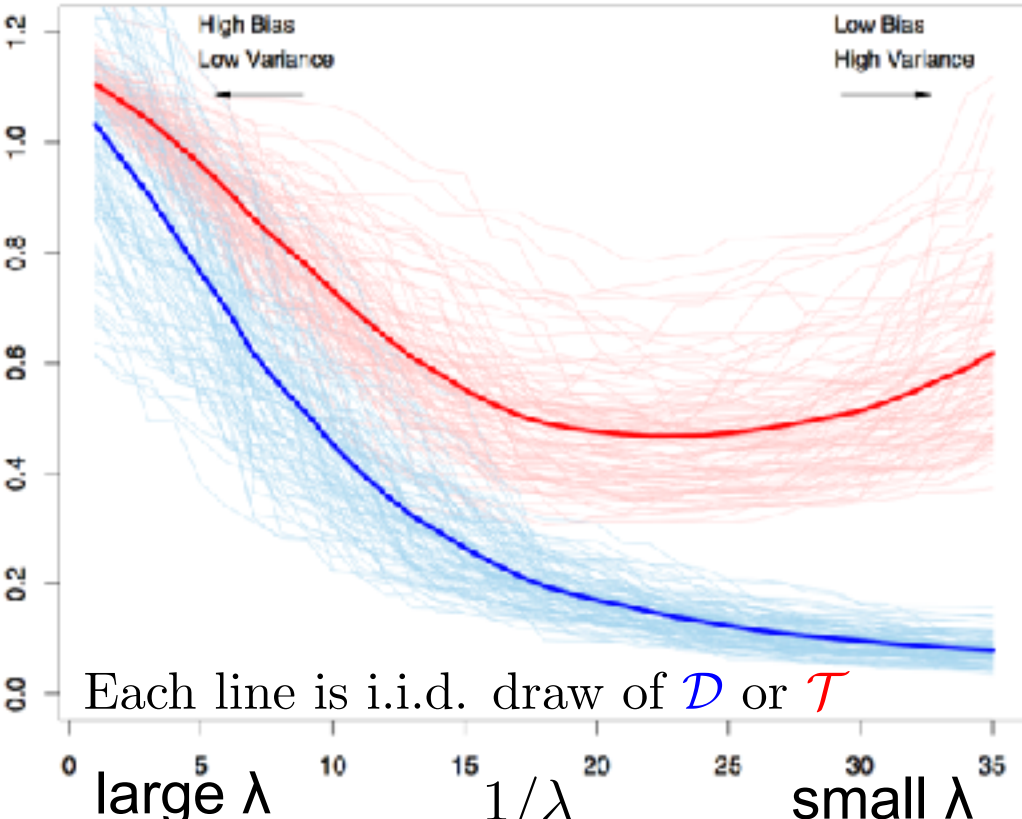
$$\frac{1}{|\mathcal{T}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T \widehat{w}^{(\lambda)}_{\mathcal{D},ridge})^2$$

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

Ridge 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:

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

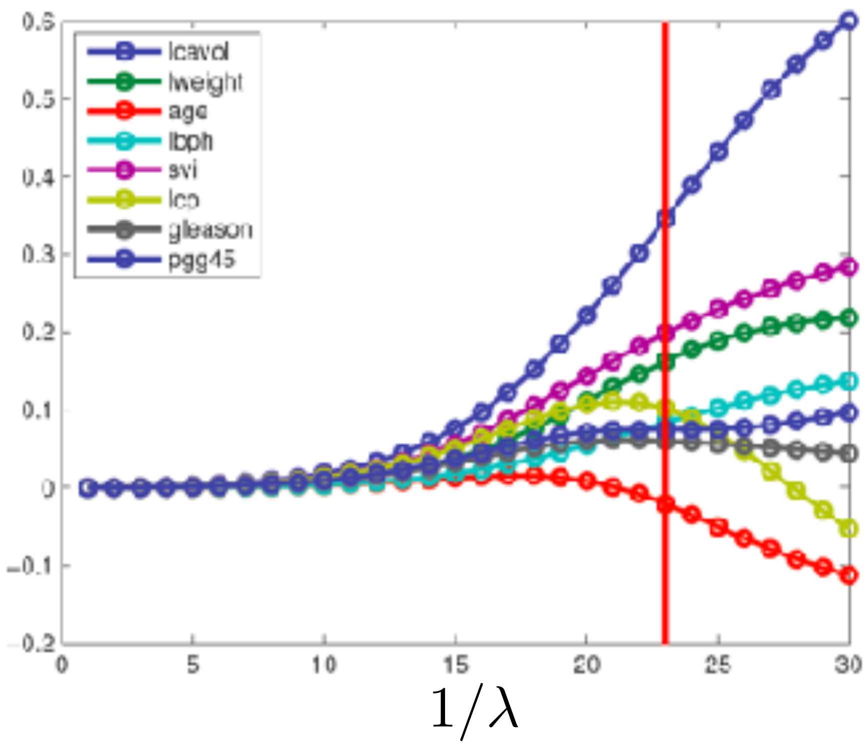
TEST error:

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

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

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

Ridge 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 9, 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 $\mathbf{y}_\mathbf{j}$:
  - Unbiased estimate of $\text{error}_{true}(f_{D\backslash j})$!

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

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

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:

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?

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

Simple Variable Selection LASSO: Sparse Regression

Machine Learning – CSE546

Kevin Jamieson
University of Washington

October 9, 2016

Sparsity

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

- Vector **w** is sparse, if many entries are zero

- Very useful for many tasks, e.g.,
  - **Efficiency**: If size(**w**) = 100 Billion, each prediction is expensive:
    - If part of an online system, too slow
    - If **w** is sparse, prediction computation only depends on number of non-zeros

Sparsity

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

- Vector **w** is sparse, if many entries are zero

- Very useful for many tasks, e.g.,
  - **Efficiency**: If size(**w**) = 100 Billion, each prediction is expensive:
    - If part of an online system, too slow
    - If **w** is sparse, prediction computation only depends on number of non-zeros
  - **Interpretability**: What are the relevant dimension to make a prediction?
    - E.g., what are the parts of the brain associated with particular words?

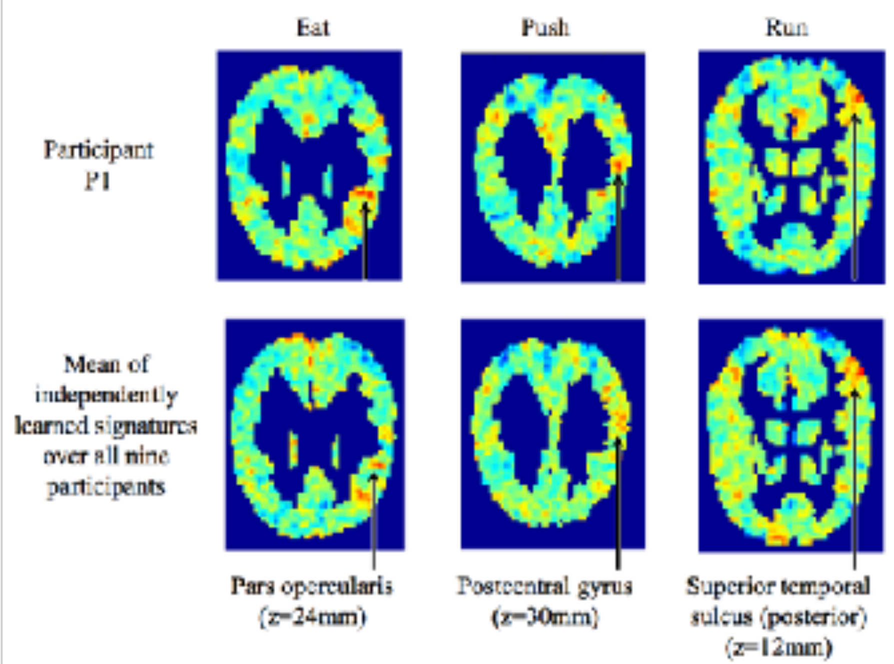


Figure from Tom Mitchell

Sparsity

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

- Vector **w** is sparse, if many entries are zero

- Very useful for many tasks, e.g.,
  - **Efficiency**: If size(**w**) = 100 Billion, each prediction is expensive:
    - If part of an online system, too slow
    - If **w** is sparse, prediction computation only depends on number of non-zeros
  - **Interpretability**: What are the relevant dimension to make a prediction?
    - E.g., what are the parts of the brain associated with particular words?

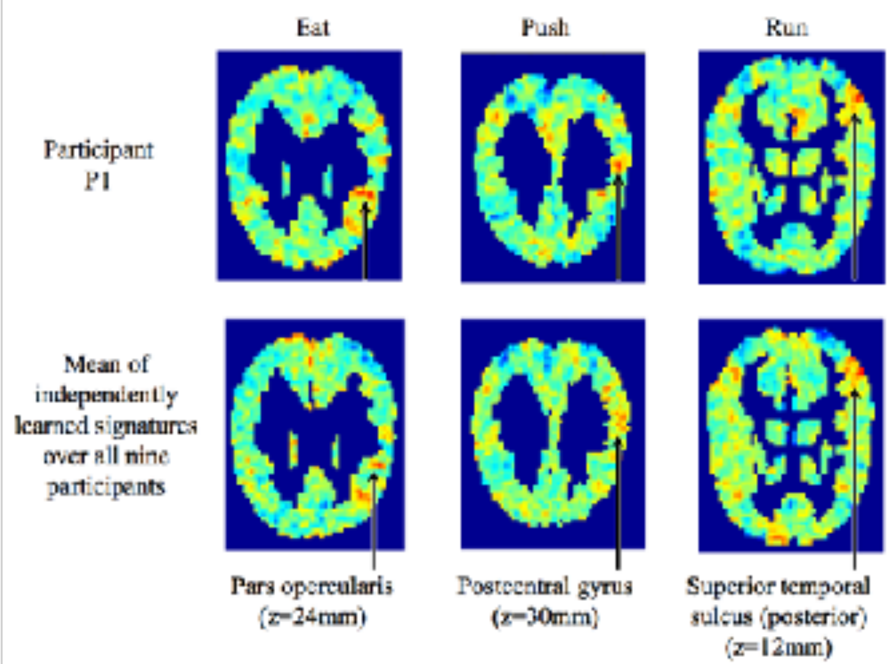


Figure from Tom Mitchell

- How do we find "best" subset among all possible?

Greedy model selection algorithm

- Pick a dictionary of features
  - e.g., cosines of random inner products
- Greedy heuristic:
  - Start from empty (or simple) set of features $F_0 = \emptyset$
  - Run learning algorithm for current set of features F_t
    - Obtain weights for these features
  - Select **next best feature $h_i(x)^*$**
    - e.g., $h_j(x)$ that results in lowest training error learner when using $F_t + \{h_j(x)^*\}$
  - $F_{t+1} \leftarrow F_t + \{h_i(x)^*\}$
  - Recurse

Greedy model selection

- Applicable in many other settings:
  - Considered later in the course:
    - Logistic regression: Selecting features (basis functions)
    - Naïve Bayes: Selecting (independent) features $P(X_i|Y)$
    - Decision trees: Selecting leaves to expand
- Only a heuristic!
  - **Finding the best set of k features is computationally intractable!**
  - Sometimes you can prove something strong about it…
-

When do we stop???

- Greedy heuristic:
  - …
  - Select **next best feature X_i^***
    - E.g. $h_j(x)$ that results in lowest training error learner when using $F_t + \{h_j(x)^*\}$
  - Recurse

When do you stop???

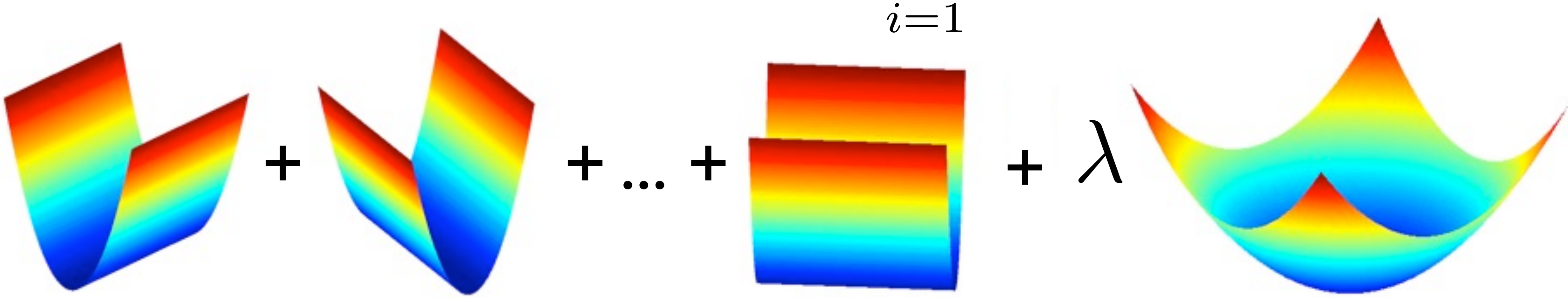
- When training error is low enough?
- When test set error is low enough?
- Using cross validation?

Is there a more principled approach?

Recall Ridge Regression

- Ridge Regression objective:

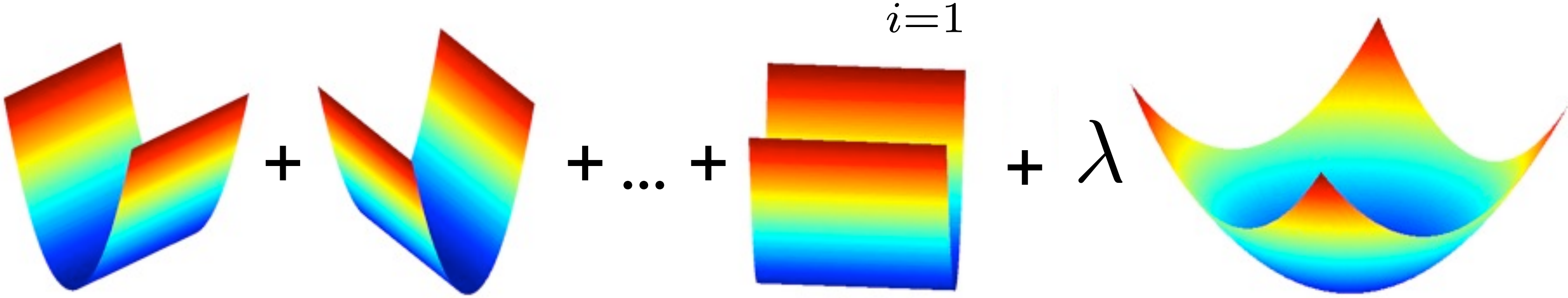
$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$



Ridge vs. Lasso Regression

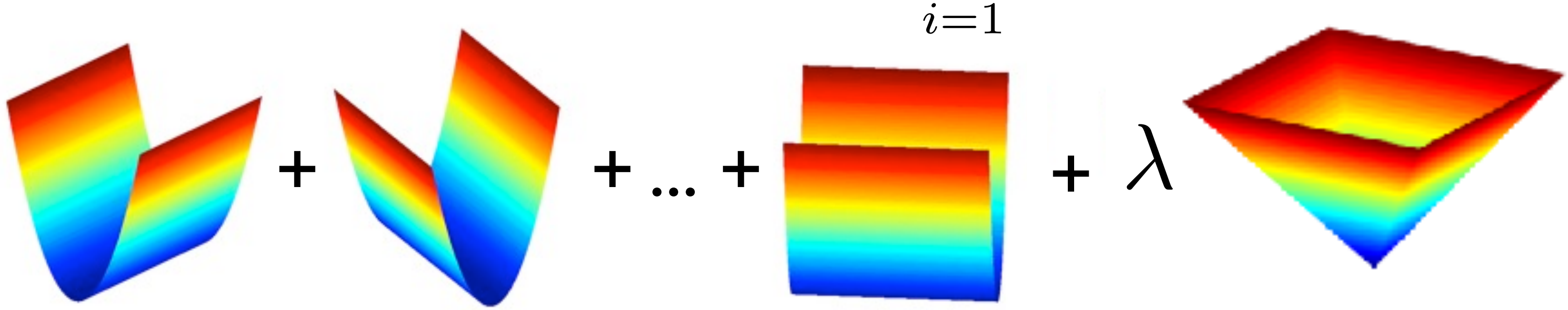
- Ridge Regression objective:

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



- Lasso objective:

$$\widehat{w}_{lasso} = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_1$$



Penalized Least Squares

$$\text{Ridge} : r(w) = ||w||_2^2 \qquad \text{Lasso} : r(w) = ||w||_1$$

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

Penalized Least Squares

$$\text{Ridge}: r(w) = ||w||_2^2 \qquad \text{Lasso}: r(w) = ||w||_1$$

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

For any $\lambda \geq 0$ for which $\widehat{w}_r$ achieves the minimum, there exists a $\nu \geq 0$ such that

$$\widehat{w}_r = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 \qquad \text{subject to } r(w) \leq \nu$$

Penalized Least Squares

$$\text{Ridge} : r(w) = ||w||_2^2 \qquad \text{Lasso} : r(w) = ||w||_1$$

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

For any $\lambda \geq 0$ for which $\widehat{w}_r$ achieves the minimum, there exists a $\nu \geq 0$ such that

$$\widehat{w}_r = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 \qquad \text{subject to } r(w) \leq \nu$$