"Regularization"

Machine Learning – CSE446

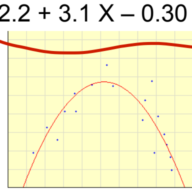
Carlos Guestrin
University of Washington

April 10, 2013

Regularization in Linear Regression

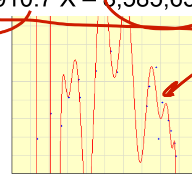
- Overfitting usually leads to very large parameter choices, e.g.:

$-2.2 + 3.1\,X - 0.30\,X^2$



$-1.1 + 4{,}700{,}910.7\,X - 8{,}585{,}638.4\,X^2 + \ldots$

penalty for large weights



overfitting

- ***Regularized*** or ***penalized*** regression aims to impose a "complexity" penalty by penalizing large weights
  - "Shrinkage" method

L_2 regularization → penalizes towards smoother functions

Ridge Regression

$\lambda > 0 \quad \|\,\|_2^2 =$

- Ameliorating issues with overfitting: penalizing large weights "regularization"
- New objective:

$$\min_{w} \underbrace{\sum_{j=1}^{N}\left(t(x_j) - \left(w_0 + \sum_{i=1}^{k} w_i h_i(x_j)\right)\right)^2}_{\text{error}_{\text{train}}} + \lambda \underbrace{\sum_{i=1}^{k} w_i^2}_{\|w_{1..k}\|_2^2}$$

RSS(w)

L_2 regularization

trade off $\text{error}_{\text{train}}$ with magnitude of coeffs.

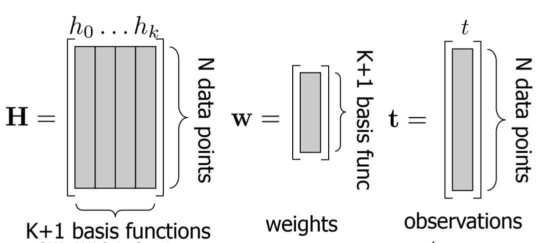
importantly, don't penalize w_0

let me free

Ridge Regression in Matrix Notation

$$\hat{\mathbf{w}}_{ridge} = \arg\min_{w} \overbrace{\sum_{j=1}^{N}\left(t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j))\right)^2}^{RSS(w)} + \overbrace{\lambda \sum_{i=1}^{k} w_i^2}^{\text{penalty}}$$

$$= \arg\min_{\mathbf{w}} \underbrace{\overbrace{(\mathbf{Hw}-\mathbf{t})^T(\mathbf{Hw}-\mathbf{t})}^{RSS(w)}}_{\text{residual error}} + \lambda\, \mathbf{w}^T I_{0+k}\, \mathbf{w}$$



$$I_{0+k} = \underbrace{\begin{pmatrix} 0 & & & 0 \\ & 1 & & \\ & & 1 & \\ 0 & & & 1 \end{pmatrix}}_{K+1}$$

$\text{length}(w) = K+1$

Minimizing the Ridge Regression Objective

given matrix A, if $\det(A)=0 \Rightarrow A^{-1}$ doesn't exist | if $\det(A)\approx 0 \Rightarrow A^{-1}$ numerically unstable

$w_0 h_0(x_j)$; $h_0(x_j)=1$

$$\hat{\mathbf{w}}_{ridge} = \arg\min_w \sum_{j=1}^{N}\left(t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j))\right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

$$= (H\mathbf{w}-\mathbf{t})^T(H\mathbf{w}-\mathbf{t}) + \lambda\, \mathbf{w}^T I_{0+k}\mathbf{w}$$

$$= w^T H^T H w + \lambda w^T I_{0+k} w - 2 w^T H^T t + t^T t = F(w)$$

$$\nabla_w F = 0 \Rightarrow 2H^T H w + 2\lambda I_{0+k} w - 2H^T t + 0 = 0$$

$$\Rightarrow 2(H^T H + \lambda I_{0+k})w = 2H^T t$$

$$\Rightarrow \hat{w}_{ridge} = (H^T H + \lambda I_{0+k})^{-1} H^T t$$

simple matrix ops !! :)

Shrinkage Properties

$$\hat{\mathbf{w}}_{ridge} = (H^T H + \lambda\, I_{0+k})^{-1} H^T \mathbf{t}$$

- If orthonormal features/basis: $H^T H = I$

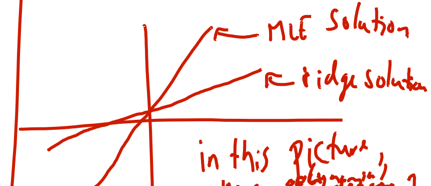
$$\hat{w}_{ridge} = (I + \lambda I_{0+k})^{-1} H^T t$$

$$\begin{pmatrix} 1 & & & 0 \\ & 1+\lambda & & \\ 0 & & 1+\lambda & \\ & & & \ddots \end{pmatrix}^{-1} H^T t$$

$$\begin{pmatrix} 1 & & 0 \\ & \frac{1}{1+\lambda} & \\ 0 & & \ddots & \frac{1}{1+\lambda} \end{pmatrix} H^T t$$

for $i>0$

$$\{\hat{w}_{ridge}\}_i = \frac{\{\hat{w}_{MLE}\}_i}{1+\lambda} \quad \} > 1$$



in this picture, fitting polynomial degree 1

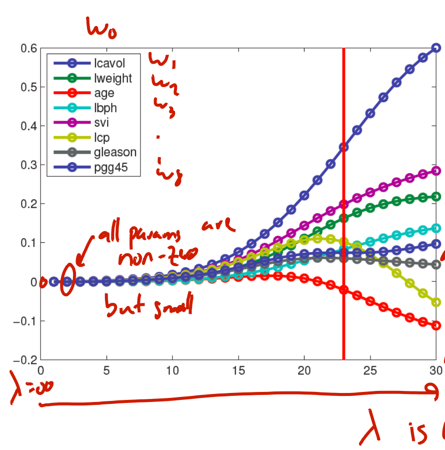
Ridge Regression: Effect of Regularization

RSS(w) regularization

$$\hat{\mathbf{w}}_{ridge} = \arg\min_w \underbrace{\sum_{j=1}^{N}\left(t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j))\right)^2}_{RSS(w)} + \underbrace{\lambda \sum_{i=1}^{k} w_i^2}_{regularization}$$

- Solution is indexed by the regularization parameter λ
- Larger λ high regularization
- Smaller λ little regularization
- As λ → 0 $\hat{w}_{ridge} \rightarrow \hat{w}_{MLE}$
- As λ →∞ $\hat{w}_{ridge} \rightarrow$ goes to a flat line, w_0 solution → average of $t(x_j)$

Ridge Coefficient Path



Some w_i large

Some smaller

$\|w\|_2^2$

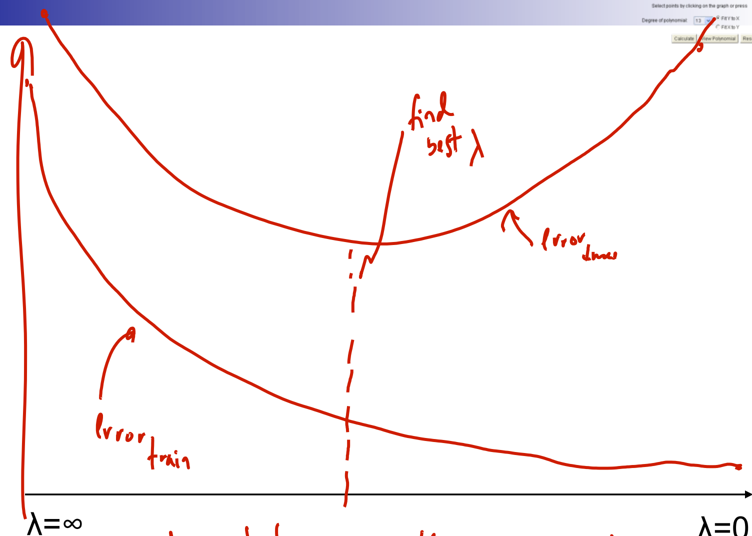
From
Kevin Murphy
textbook

- Typical approach: select λ using cross validation, more on this later in the quarter

Error as a function of regularization parameter for a fixed model complexity

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j)\right)^2$$

$$error_{true}(\mathbf{w}) = \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x})\right)^2 p(\mathbf{x})\,d\mathbf{x}$$



model selection, picking right λ

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 – CSE446
Carlos Guestrin
University of Washington

April 10, 2013

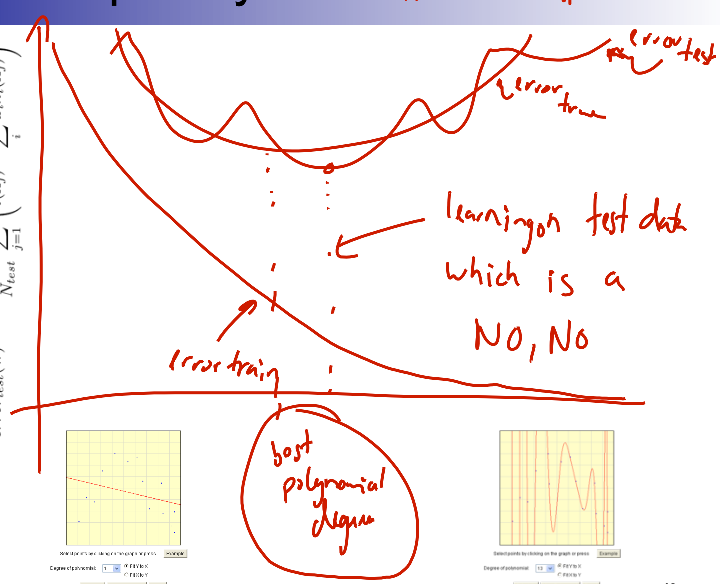
Test set error as a function of model complexity

in expectation $error_{true} = error_{test}$ ✓ has extra variability

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j)\right)^2$$

$$error_{true}(\mathbf{w}) = \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x})\right)^2 p(\mathbf{x})d\mathbf{x}$$

$$error_{test}(\mathbf{w}) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j)\right)^2$$



How… How… How???????

- How do we pick the regularization constant λ…
  - And all other constants in ML, 'cause one thing ML doesn't lack is constants to tune… ☹
- We could use the test data, but…

1. don't have enough data to set aside for picking λ

2. you can't use test data

→ overfitting to test data

(LOO) Leave-one-out cross validation

choose λ on one example

- Consider a **validation set with 1 example**:
  - *D* – training data
  - *D*\j – training data with *j* th data point moved to validation set
- **Learn ~~classifier~~ regressor $h_{D\backslash j}$ with *D*\j dataset**
- **Estimate true error** as squared error on predicting t($\mathbf{x}_j$):
  - Unbiased estimate of $error_{true}(h_{D\backslash j})$!

$$error_{true}(h_{D\backslash j}) \approx (h_{D\backslash j}(x_j) - t(x_j))^2$$

$error_{true}(h_{D\backslash j})$

$= E[(h_{D\backslash j} - t)^2]$

$= E[(h_{D\backslash j}(x_j) - t(x_j))^2]$

  - Seems really bad estimator, but wait!
- **LOO cross validation**: Average over all data points *j*:
  - **For each data point you leave out, learn a new classifier $h_{D\backslash j}$**
  - **Estimate error** as:

$$error_{LOO} = \frac{1}{N}\sum_{j=1}^{N}\left(t(\mathbf{x}_j) - h_{\mathcal{D}\backslash j}(\mathbf{x}_j)\right)^2$$

$\approx E[(t - h_{N-1})^2]$ ← estimate what I can learn with N−1 points

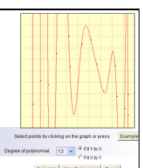
LOO cross validation is (almost) unbiased estimate of true error of h_D!

- 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**!

$$error_{true}(h_D) \approx E[\,error_{LOO}(h_D)\,]$$

- **Great news!**
  - **Use LOO error for model selection!!!**
  - **E.g., picking λ**

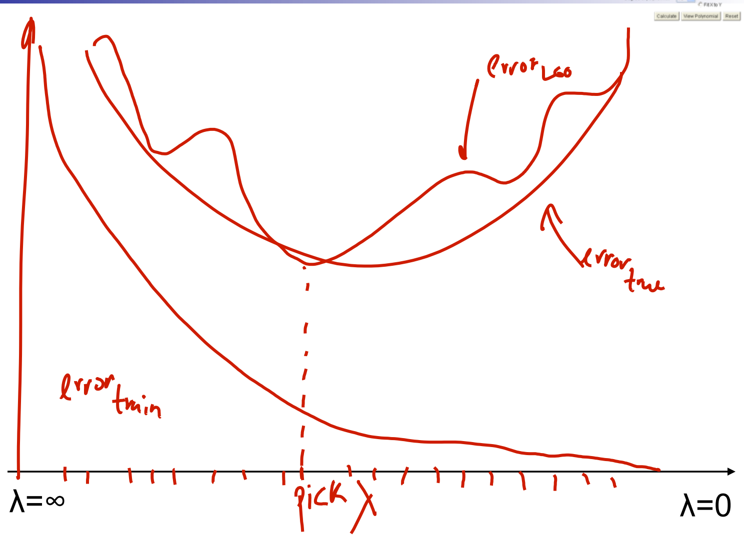
Using LOO to Pick λ



$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

$$error_{true}(\mathbf{w}) = \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 p(\mathbf{x}) d\mathbf{x}$$

$$error_{LOO} = \frac{1}{N} \sum_{j=1}^{N} \left(t(\mathbf{x}_j) - h_{\mathcal{D}\setminus j}(\mathbf{x}_j) \right)^2$$



Using LOO error for model selection

$$error_{train}(\mathbf{w}) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2$$

$$error_{true}(\mathbf{w}) = \int_{\mathbf{x}} \left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x}) \right)^2 p(\mathbf{x}) d\mathbf{x}$$

$$error_{LOO} = \frac{1}{N} \sum_{j=1}^{N} \left(t(\mathbf{x}_j) - h_{\mathcal{D}\backslash j}(\mathbf{x}_j) \right)^2$$

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!!!
  - If you have to do for each choice of basis functions, it will take fooooooreeeve'!!!
- Solution 1: Preferred, but not usually possible
  - Find a cool trick to compute LOO (e.g., see homework)

in closed form

Solution 2 to complexity of computing LOO: (More typical) **Use *k*-fold cross validation**

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

$$error_{\mathcal{D}_i} = \frac{k}{N} \sum_{\mathbf{x}_j \in \mathcal{D}_i} \left(t(\mathbf{x}_j) - h_{\mathcal{D}\backslash\mathcal{D}_i}(\mathbf{x}_j)\right)^2$$

- ***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 *m(k-1)/k*
  - Usually, **k = 10** ☺

What you need to know…

- 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
- Use cross-validation to choose magic parameters such as λ
- Leave-one-out is the best you can do, but sometimes too slow
  - In that case, use k-fold cross-validation