Do not distribute course material

You may not and may not allow others to reproduce or distribute lecture notes and course materials publicly whether or not a fee is charged.

https://www.cs.princeton.edu/courses/archive/spr09/cos513/scribe/lecture10.pdf

http://www.stat.cmu.edu/~ryantibs/advmethods/notes/highdim.pdf

http://timroughgarden.org/s17/l/l6.pdf

Regularization - Preventing overfitting

As much art as science…

How can we reduce the out of sample error by preferring some solutions in our hypothesis class

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y})^2 = \frac{1}{N}RSS(\mathbf{w})$$

https://www.cs.princeton.edu/courses/archive/spr09/cos513/scribe/lecture10.pdf

http://www.stat.cmu.edu/~ryantibs/advmethods/notes/highdim.pdf

http://timroughgarden.org/s17/l/l6.pdf

Regularization Preventing overfitting

A way to prefer some model/hypothesis over others in our class based on some idea of what is the preferred model

ence…

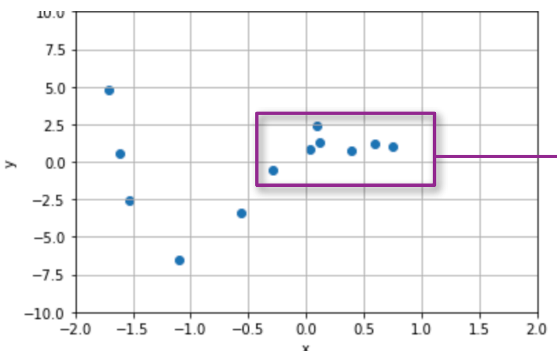
How can we reduce the out of sample error by preferring some solutions in our hypothesis class

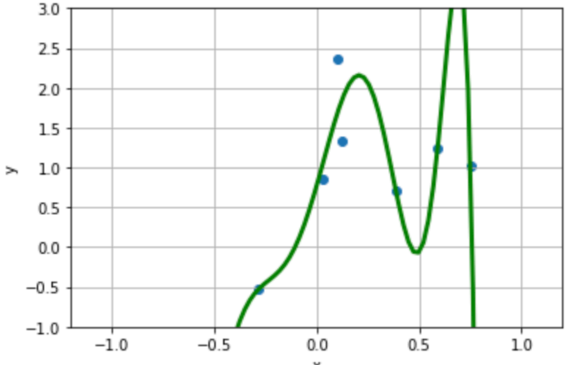
$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y})^2 = \frac{1}{N}RSS(\mathbf{w})$$

Learning objectives

- Understand regularizer can be used to decrease overfitting
- Understand how to create an objective function that prefers functions with smaller coefficients (simpler models) by adding a L1 or L2 penalty term
- Understand why we don't regularize the bias term
- How the L1 or L2 penalty term affects bias and variance
- How to use model selection to tune the hyper-parameter λ
- L1 regularization can produce feature selection
- Understand that feature scaling should be performed in most cases

Example:





$\mathbf{w}^T$=[0.8 9 9 -76 -176 208 551

Poor Generalization!

If $w_j = 551$ then a small change to the value of the jth feature makes a huge change in $\hat{y}$

Observations:
Notice that the amount of overfitting depended on the order of the model and how many examples we have.
Our hypothesis that overfit had large coefficients. How could we keep the coefficients small?

We will need to balance between how well we fit the data (the in sample error) and how much we restrict the size of our coefficients (that we are using to prevent overfitting)

$$\underbrace{E_{\text{in}}(\mathbf{w})}_{\text{fit}} + \underbrace{\text{penalty for large } \mathbf{w}}_{\text{restrict the size of our coefficients}}$$

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y})^2 = \frac{1}{N}RSS(\mathbf{w})$$

Lasso - *L*east *A*bsolute *S*election and *S*hrinkage *O*perator

PERFORM VARIABLE SELECTION

MORE EFFICIENT TO HAVE LESS FEATURES

INTERPRETABILITY

Approach taken from https://courses.cs.washington.edu/courses/cse446/17wi/slides/lasso-annotated.pdf

Too many features!

In some datasets, only a subset of the features contribute to the answer

Removing features reduces variance

Removing features makes understanding the coefficients easier

How could we choose which features to use?

- Run the algorithm on all possible subsets of the features
- Remove features one by one are rerun the algorithm - seeing if it get worse
- Start with one feature and slowly add new features if “they help”

We can use LASSO regularization to reduce the number of features d = # features. Note we don't want to restrict w_0. Many approaches to this issue are possible. We will leave w_0 out of the penalty term. Note: Scaling the features is suggested

Lasso Regression

If λ=0 then $\mathbf{w}_{\text{lasso}} = \mathbf{w}_{\text{lin}}$

$\mathbf{w}_{\text{lasso}}$ = the best parameters for LASSO

$\mathbf{w}_{\text{lin}}$ = the best parameters for least squares cost

If $\lambda$is very large then

$$w_i \sim 0 \text{ for all } i > 0$$

If λ is a constant then

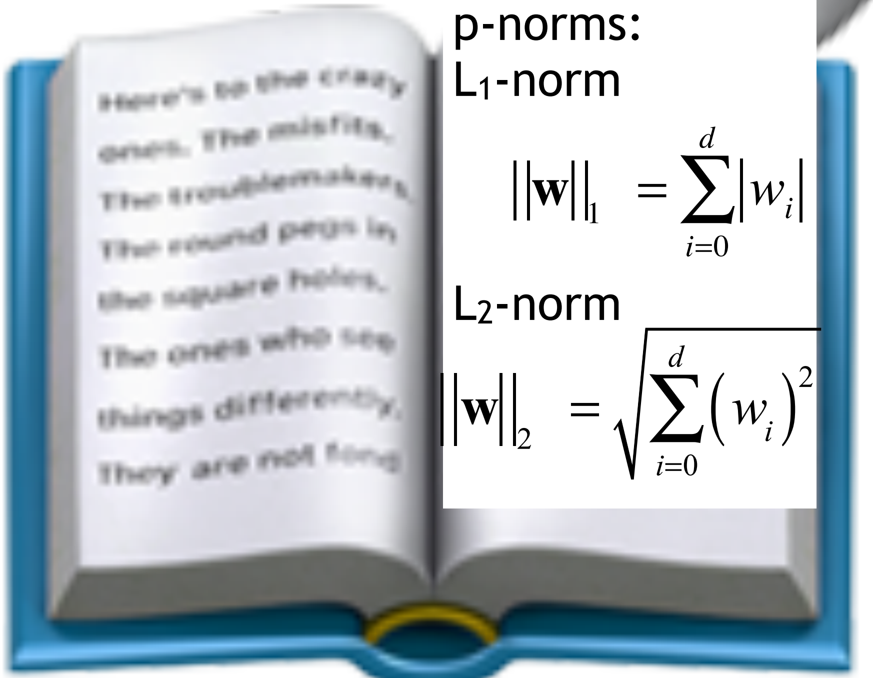
$$0 \le \|w_{\text{lasso}}\|_1 \le \|w_{\text{lin}}\|_1$$

- Tuning parameter λ to balance fit and number of parameters

$$\mathrm{E}_{\text{aug}}(\mathbf{w}) = \mathrm{E}_{\text{in}}(\mathbf{w}) + \text{penalty for complex models}$$

$$\mathrm{E}_{\text{lasso}}(\mathbf{w}) = \mathrm{E}_{\text{in}}(\mathbf{w}) + \lambda\left(|\cancel{w_0}| + |w_1| + |w_2| + \cdots + |w_d|\right)$$

- λ controls the model complexity
  - Large λ
    - high bias, low variance
  - small λ
    - low bias, high variance



p-norms:

L_1-norm

$$\|\mathbf{w}\|_1 = \sum_{i=0}^{d} |w_i|$$

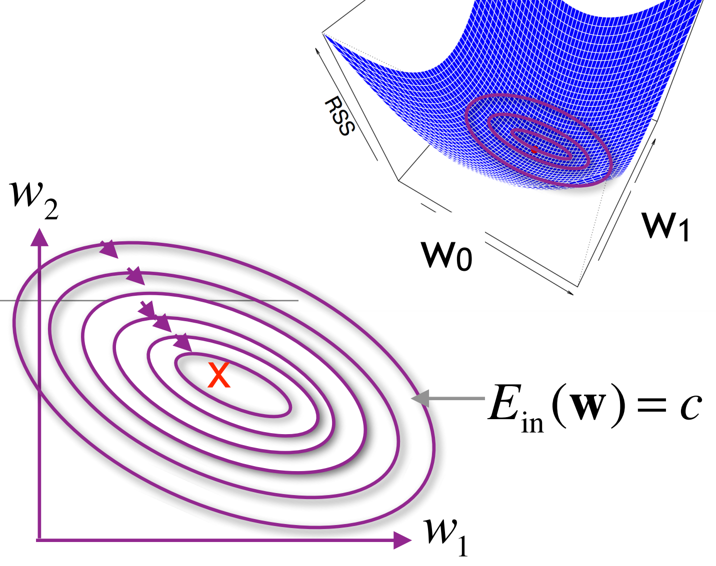
L_2-norm

$$\|\mathbf{w}\|_2 = \sqrt{\sum_{i=0}^{d} (w_i)^2}$$

Geometric Intuition

❑Looking at the contour plot of RSS

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2 = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \mathbf{w}^T\mathbf{x})^2$$

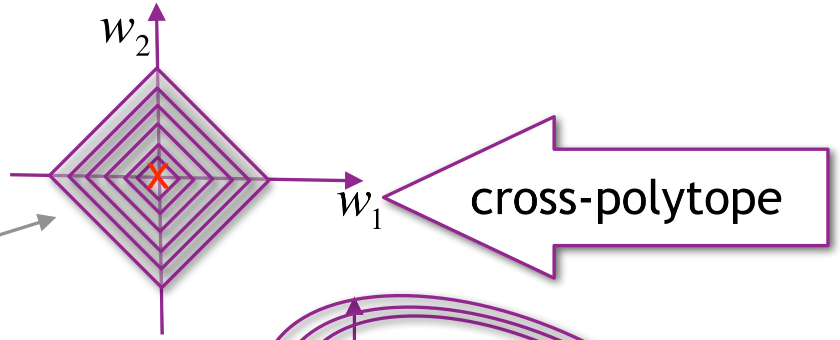


❑Looking at the contour plot of the L_1 norm

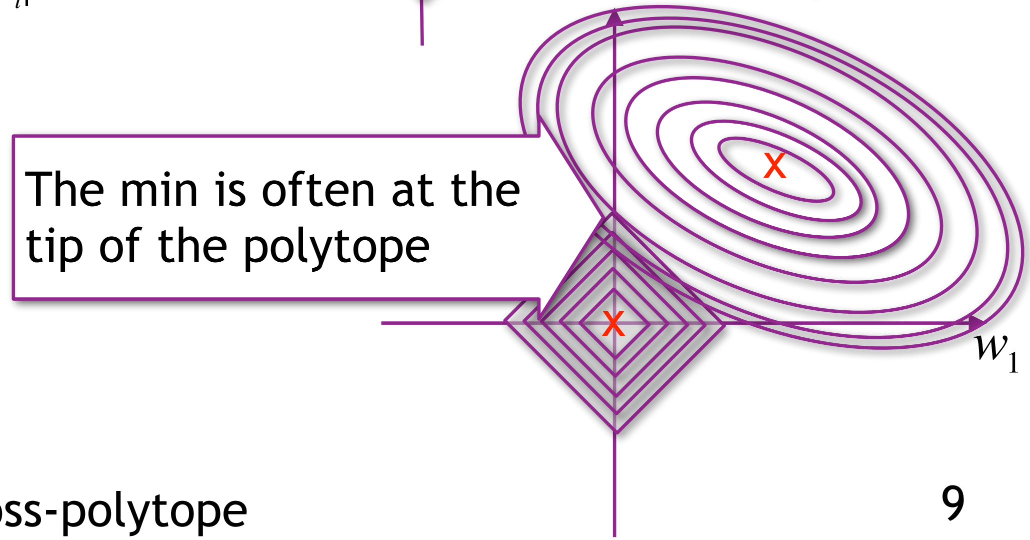
$$\|\mathbf{w}_{1:d}\|_1 = \sum_{i=1}^{d} |w_i|$$

Level curve/contour line/Isoline

$$\|\mathbf{w}_{1:d}\|_1 = \sum_{i=1}^{d} |w_i| = s$$



❑Looking at $\mathrm{E}_{\text{lasso}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \lambda\|\mathbf{w}_{1:d}\|_1$



https://en.wikipedia.org/wiki/Cross-polytope

Uh Oh, no closed form solution for minimizing

$$\mathrm{E}_{\text{lasso}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \lambda \|\mathbf{w}_{1:d}\|_1$$

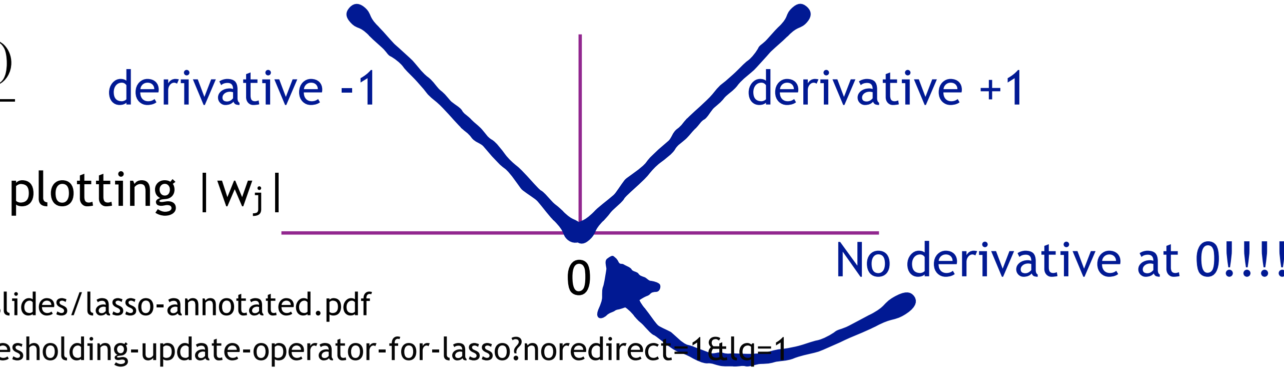
- $E_{lasso}(\mathbf{w})$ is convex, but we cannot take the derivative and set it to zero to find the optimal **w**
- It is possible to optimize the j[th] coefficient while the others remain fixed

$$w_j^* = \arg\min_z E_{\text{lasso}}(\mathbf{w} + z\mathbf{e_j})$$

$\mathbf{e_j}$ is the jth unit vector

$$\frac{\partial E_{\text{lasso}}(\mathbf{w})}{\partial w_j} = \frac{2}{N}\sum_{i=1}^{N}(y_i - (w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_j x_j^{(i)} + \cdots + w_d x_d^{(i)}))(-x_j^{(i)}) + \lambda \frac{\partial \|\mathbf{w}_{1:d}\|_1}{\partial w_j}$$

$$\frac{\partial \lambda \|\mathbf{w}_{1:d}\|_1}{\partial w_j} = \lambda \frac{\partial \|\mathbf{w}_{1:d}\|_1}{\partial w_j} = \lambda \frac{\partial(|w_1| + \cdots + |w_d|)}{\partial w_j} = \lambda \frac{\partial(|w_j|)}{\partial w_j}$$



Approach taken from:https://courses.cs.washington.edu/courses/cse546/14au/slides/lasso-annotated.pdf
https://stats.stackexchange.com/questions/123672/coordinate-descent-soft-thresholding-update-operator-for-lasso?noredirect=1&lq=1
https://support.sas.com/resources/papers/proceedings15/3297-2015.pdf
http://www.math.mcgill.ca/yyang/regression/extra/lasso.pdf
https://en.wikipedia.org/wiki/Lasso_(statistics)
http://myweb.uiowa.edu/pbreheny/7600/s16/notes/2-15.pdf

Ridge and LASSO

Overview

Suggested to **scale** features before performing ridge regression or lasso regression.

Ridge regression has a **closed form solution**.
Ridge regression shrinks coefficients towards zero.

Lasso does not have a closed form solution.
Lasso performs **feature selection** as well as **parameter estimation**.

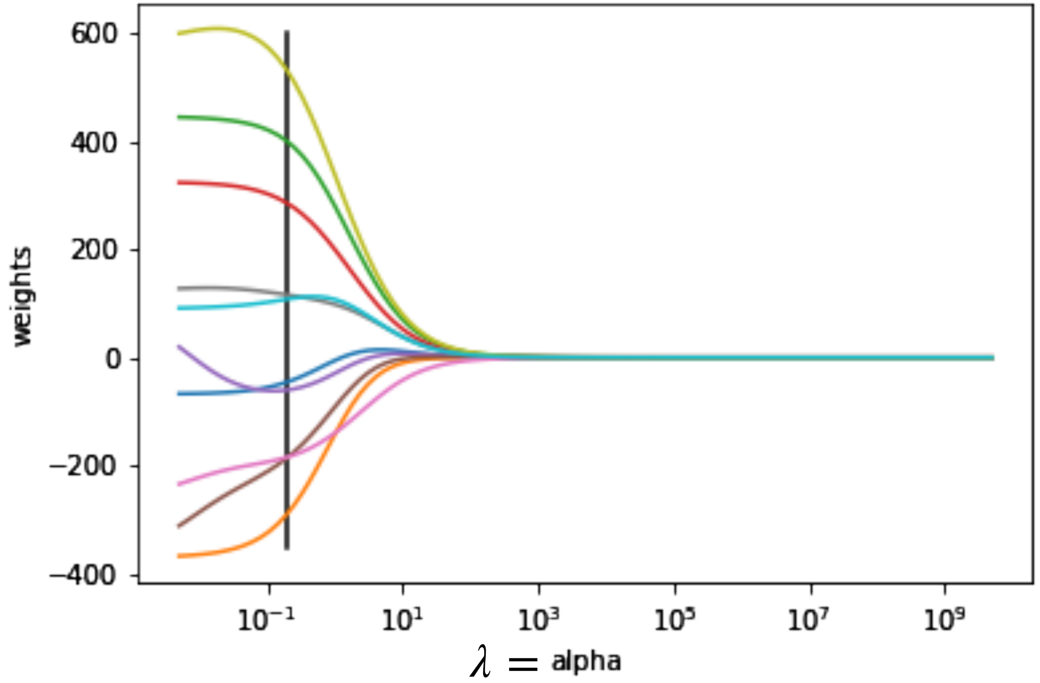
For both lasso and ridge regression, the tuning parameter λ **controls the strength of the regularization**.

Both ridge and lasso **increase bias, but decrease variance**

How changes in λ affect the optimal coefficients

RIDGE

$$E_{ridge}(\mathbf{w}) = E_{in}(\mathbf{w}) + \lambda(w_1^2 + w_2^2 + w_3^2 + \cdots + w_d^2)$$

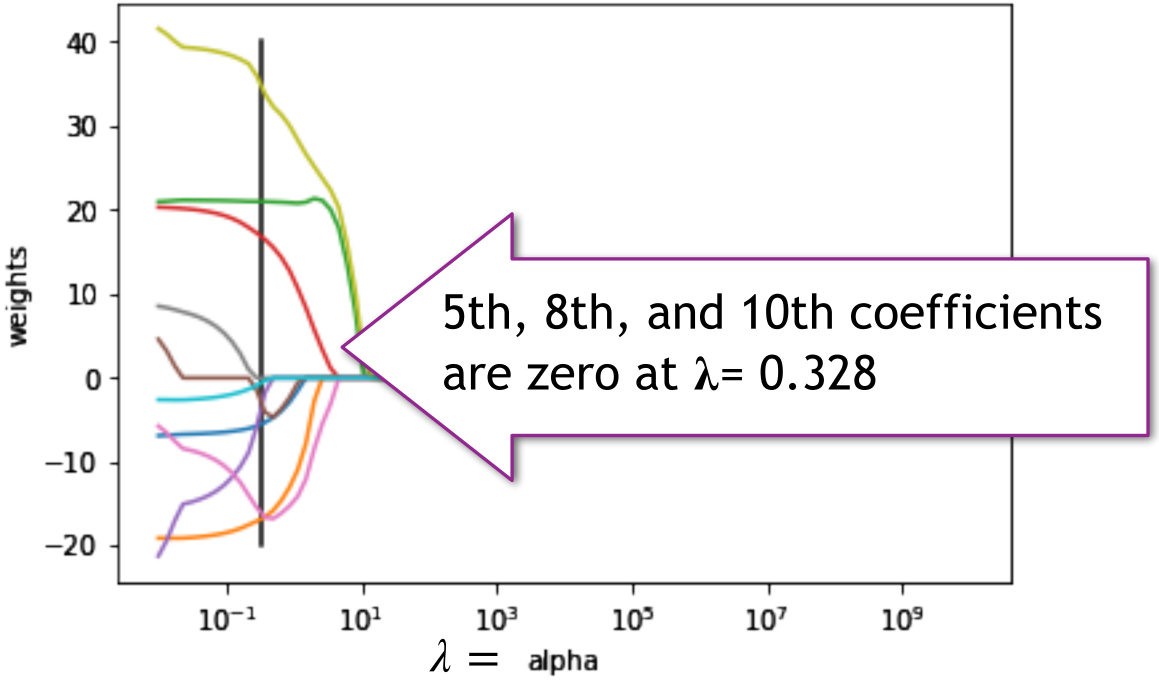



```
The best λ = alpha: 0.188
```

$$E_{ridge}(\mathbf{w}) = E_{in}(\mathbf{w}) + 0.188(w_1^2 + w_2^2 + w_3^2 + \cdots + w_d^2)$$

LASSO

$$E_{lasso}(\mathbf{w}) = E_{in}(\mathbf{w}) + \lambda\left(|w_1| + |w_2| + |w_3| + \cdots + |w_d|\right)$$




```
The best λ = alpha:  0.328
```

$$E_{lasso}(\mathbf{w}) = E_{in}(\mathbf{w}) + 0.328\left(|w_1| + |w_2| + |w_3| + \cdots + |w_d|\right)$$

Standardization in Scikit-Learn

Many algorithm (such as ridge regression and lasso regression) require the data to be standardized t
work correctly.

We want to apply the same scaling we did on the test data to future examples…

```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train) # zero mean and unit variance
# Then later when want to use our classifier on new data, we scale the new data the same way we scaled the training
X_test_scaled = scaler.transform(X_test)
```

More information from http://scikit-learn.org/stable/modules/preprocessing.html
Many other variations of scaling can be found here

Ridge Regression in Scikit-Learn

auto is default

solver : {*‘auto’, ‘cholesky’, ‘sag”,…}*

closed form solution we proved

stochastic gradient descent (typically faster when when X is large). Needs scaled data!

```
from sklearn.linear_model import Ridge
import numpy as np

clf = Ridge(alpha=1.0, solver = ‘cholesky’)
clf.fit(X, y)
```

```
print(clf.coef_)
print(clf.intercept_)
```

Some the methods

fit(X, y)	Fit Ridge regression model.
predict(X)	Predict class labels for samples in X.
score(X, y[, sample_weight])	Return the coefficient of determination of the prediction

Information from http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

Lasso Regression in Scikit-Learn

```
from sklearn import linear_model
clf = linear_model.Lasso(alpha=0.1)
clf.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])

print(clf.coef_)
print(clf.intercept_)
```

[0.85   0.]

0.150

Some of the methods

fit(X, y)	Fit Lasso regression model.
predict(X)	Predict class labels for samples in X.
score(X, y[, sample_weight])	Return the coefficient of determination of the prediction

Information from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html