Unit 5
LASSO Regularization and Feature Selection

EL-GY 9143: INTRODUCTION TO MACHINE LEARNING

PROF. PEI LIU

Learning Objectives

- ❑ Describe model selection and identify when it may be needed
- ❑ Mathematically describe linear regression with regularization
- ❑ Select regularizers to impose constraints such as sparsity
- ❑ Compute an L1-regularized estimate (LASSO) using sklearn tools
- ❑ Compute the optimal regularization level using cross validation
- ❑ Interpret results from a LASSO path
- ❑ Set regularizer based on a probabilistic prior
- ❑ Perform other feature selection methods

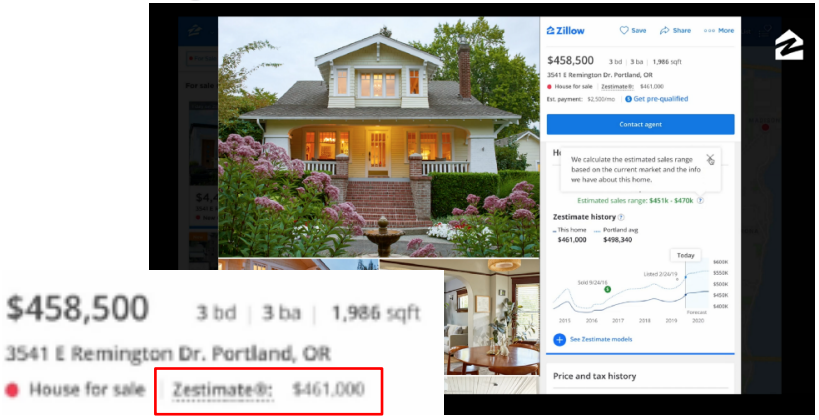
Outline

- Motivating Example: Feature selection in predicting housing prices
- Model selection and regularization
- Housing prices prediction with LASSO
- Probabilistic interpretation
- Other model selection examples
- Other model selection methods
- In-Class Exercise: Audio Pitch Detection

Predicting Housing Prices

AI, MACHINE LEARNING & RESEARCH

Introducing a new and improved Zestimate algorithm



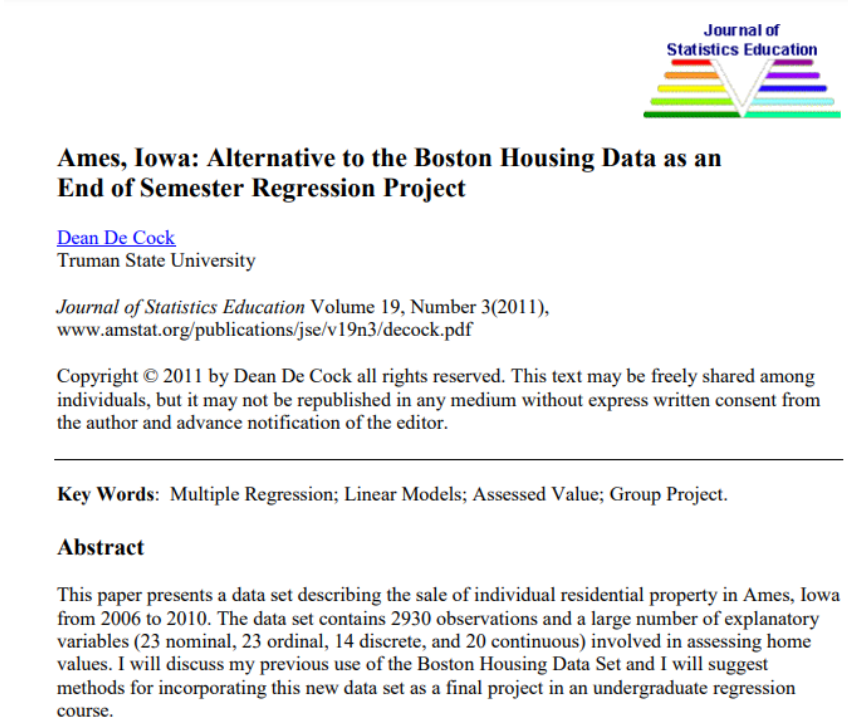
https://www.zillow.com/tech/introducing-a-new-and-improved-zestimate-algorithm/

- Many services now predict house prices
- Data science enters real estate!
- Many possible variables:
  - Square meters
  - Condition
  - Zip code
  - Education quality
  - …
- What variables *really* determine the price?

Zillow

StreetEasy

Ames, Iowa Dataset



Journal of Statistics Education

Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project

Dean De Cock
Truman State University

Journal of Statistics Education Volume 19, Number 3(2011),
www.amstat.org/publications/jse/v19n3/decock.pdf

Key Words: Multiple Regression; Linear Models; Assessed Value; Group Project.

Abstract

This paper presents a data set describing the sale of individual residential property in Ames, Iowa from 2006 to 2010. The data set contains 2930 observations and a large number of explanatory variables (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) involved in assessing home values. I will discuss my previous use of the Boston Housing Data Set and I will suggest methods for incorporating this new data set as a final project in an undergraduate regression course.

- Ames, Iowa Dataset
  - Sales from 2006 to 2010
  - From Dean De Cock
- Alternative to Boston Housing dataset
- Many more variables to explore
  - Approximately 81 variables
  - 2930 samples

Loading the Dataset

```
df = pd.read_csv('housing_train.csv')
df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

5 rows × 81 columns

- Issues:
  - Many different types of data: Discrete and continuous
  - Missing values (NaN)

Data Cleaning

```
1  nsamp, natt = df.shape
2  print('Number samples = %d' % nsamp)
3  print('Number attributes per sample = %d' % natt)
```

```
Number samples = 1460
Number attributes per sample = 81
```

- Original data

```
1  df = df.dropna(axis=1)
2
3  nsamp, natt = df.shape
4  print('Number samples = %d' % nsamp)
5  print('Number attributes per sample = %d' % natt)
6
```

- Remove columns with NaN values
  - Could use more sophisticated methods

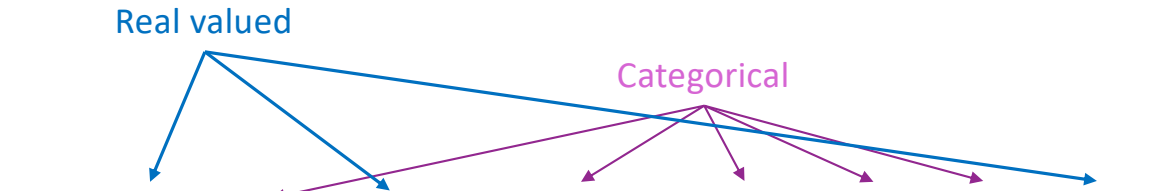
```
1  df = df.loc[df['SaleCondition'] == 'Normal']
2
3  nsamp, natt = df.shape
4  print('Number samples = %d' % nsamp)
5  print('Number attributes per sample = %d' % natt)
```

```
Number samples = 1198
Number attributes per sample = 62
```

- Keep only normal sales
  - Recommended in De Cock paper
  - Makes fitting much easier

Categorical Variables

- Data has many categorical variables
- Need to code the categorical variables to numerical values



	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0

Splitting the Variables

- First, split the variables into categorical and real

```
# Remove the ID, month sold and sales price (it is the target)
ignore_vars = ['Id', 'MoSold', 'SalePrice']

# Find real and categorical variables
cols = df.columns
cat_vars = []
real_vars = []

for col in cols:

    if not (col in ignore_vars):
        if df.dtypes[col]  == 'object':
            cat_vars.append(col)
        else:
            real_vars.append(col)
```

```
Categorical variables = ['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'N
eighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd',
'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType
2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageType', 'GarageFin
ish', 'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'SaleCondition']

Real variables = ['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnr
Area', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFul
lBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt',
'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal',
'YrSold']
```

One Hot Coding

Original

	MSZoning	Street	LotShape	LandContour	Utilities	LotConfig
0	RL	Pave	Reg	Lvl	AllPub	Inside
1	RL	Pave	Reg	Lvl	AllPub	FR2
2	RL	Pave	IR1	Lvl	AllPub	Inside
4	RL	Pave	IR1	Lvl	AllPub	FR2
5	RL	Pave	IR1	Lvl	AllPub	Inside

One Hot coded

	MSZoning_FV	MSZoning_RH	MSZoning_RL	MSZoning_RM	Street_Pave
0	0	0	1	0	1
1	0	0	1	0	1
2	0	0	1	0	1
4	0	0	1	0	1
5	0	0	1	0	1

- Use pandas get_dummies
- Replaces categorical variables with one-hot coded values
  - Ex: MSZoning
  - Becomes MSZoning_FV, MSZoning_RH, …

```
# Get the dataframes with real and categorical variables
df_real = df[real_vars]
df_cat = df[cat_vars]

# One-hot encode the categorical variables
df_cat_enc = pd.get_dummies(df_cat, drop_first=True)
```

Scaling Data

- Split data into training and test
- Scale data
  - Remove mean and variance
- Needed to compare coefficients
  - Ensures that all variables have same range
- Note: The scaling transform is
  - Fit on the training data
  - Performed on training and test

```
from sklearn.model_selection import train_test_split

Xtr, Xts, ytr, yts = train_test_split(X,y,test_size=0.3)
```

```
from sklearn.preprocessing import StandardScaler

# Create the scaler objects
xscal = StandardScaler()
yscal = StandardScaler()

# Fit and transform the training data
Xtr1 = xscal.fit_transform(Xtr)
ytr1 = yscal.fit_transform(ytr[:,None])

# Transform the test data
Xts1 = xscal.transform(Xts)
yts1 = yscal.transform(yts[:,None])
```

First Try: Linear Regression

```python
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.metrics import r2_score

# Fit
reg_ls = LinearRegression()
reg_ls.fit(Xtr1, ytr1)

# Training R^2
yhat1_tr = reg_ls.predict(Xtr1)
rsq_tr = r2_score(yhat1_tr, ytr1)
print('Training R^2 = %12.4e' % rsq_tr)

# Test R^2
yhat1_ts = reg_ls.predict(Xts1)
rsq_ts = r2_score(yts1, yhat1_ts)
print('Test R^2     = %12.4e' % rsq_ts)
```

```
Training R^2 =   9.3726e-01
Test R^2     =  -1.0430e+20
```

- Simple idea:
  - Use linear regression over features
- Fits the training data very well!
  - R^2 ≈ 0.937
- But, completely fails on the test data
  - $R^2 > 10^{20}$

Conditioning

- What went wrong?
- Recall LS solution is: $\hat{\beta} = (A^T A)^{-1} A^T y$
- Matrix $A^T A$ may be ill-conditioned
  - Eigenvalues close to zero
  - Inverse blows up
- With ill-conditioned data:
  - Training error is fine
  - But the test error blows up
- Overfits data

```python
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.metrics import r2_score

# Fit
reg_ls = LinearRegression()
reg_ls.fit(Xtr1, ytr1)

# Training R^2
yhat1_tr = reg_ls.predict(Xtr1)
rsq_tr = r2_score(yhat1_tr, ytr1)
print('Training R^2 = %12.4e' % rsq_tr)

# Test R^2
yhat1_ts = reg_ls.predict(Xts1)
rsq_ts = r2_score(yts1, yhat1_ts)
print('Test R^2     = %12.4e' % rsq_ts)
```

```
Training R^2 =   9.3726e-01
Test R^2     =  -1.0430e+20
```

Improving Conditioning via Ridge Regression

- Standard LS solution: $\hat{\beta} = (A^T A)^{-1} A^T y$
- Ridge Regression: Add a conditioning term:

$$\hat{\beta} = (A^T A + cI)^{-1} A^T y$$

  - c is a small positive value.
  - Makes inverse well-behaved
  - We will see this technique more later

- Get good test R^2

```
reg_ls = Ridge(alpha=1e-5)
reg_ls.fit(Xtr1, ytr1)
yhat1 = reg_ls.predict(Xts1)
rsq = r2_score(yts1, yhat1)
print('Test R^2      = %12.4e' % rsq)
```

```
Test R^2      = 0.904567
```

What Components Matter?

- Simple idea: Look at large coefficients
- We see variables that we may expect:
  - Square footage
  - Quality
  - Zoning
- But there are some issues
  - Some variables seem highly correlated
  - Ex: GrLivArea and 2ndFlrSF
  - KitchenQual and OverallQual
  - Do we need both?

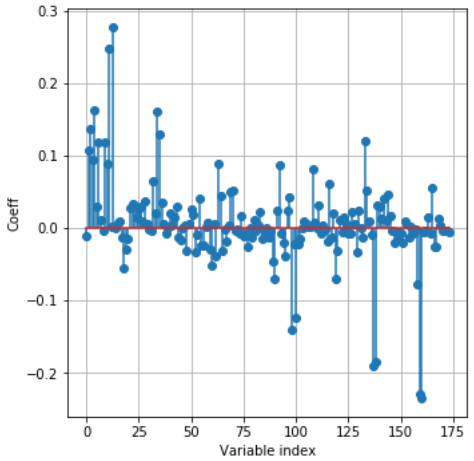
```
coeff_ls = reg_ls.coef_.ravel()
nprint = 10
I = np.argsort(np.abs(coeff_ls))
I = np.flipud(I)
for i in range(nprint):
    j = I[i]
    print('%20s %f' % (xnames[j], coeff_ls[j]) )
```

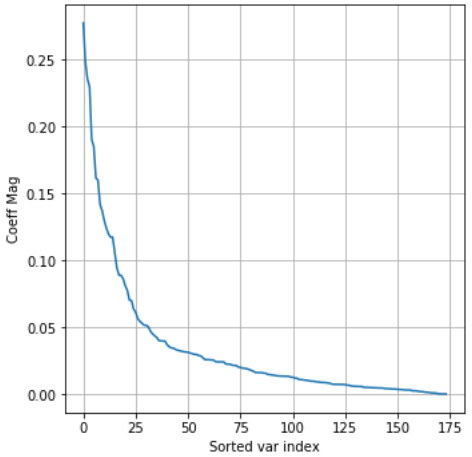
```
        GrLivArea 0.277146
         2ndFlrSF 0.248113
   KitchenQual_TA -0.235208
   KitchenQual_Gd -0.229472
     ExterQual_Gd -0.190125
     ExterQual_TA -0.184943
        YearBuilt 0.161462
      MSZoning_RL 0.159591
  RoofStyle_Gable -0.141669
      OverallQual 0.136913
```

What Components Do *Not* Matter?

- All coefficients are far from zero
- Very few coefficients that can be removed
- Does this mean all variables matter?

- Model or feature selection problem:
  - *How do we find the variables that matter?*





Outline

- ❑Motivating Example: Feature selection in predicting housing prices
- ❑Model selection and regularization
- ❑Housing prices prediction with LASSO
- ❑Probabilistic interpretation
- ❑Other model selection examples
- ❑Other model selection methods
- ❑In-Class Exercise: Audio Pitch Detection

Model Selection via Sparsity

	MSZoning_FV	MSZoning_RH	MSZoning_RL	MSZoning_RM	Street_Pave	LotShape_IR2	LotShape_IR3	LotShape_Reg	LandContour_HLS
0	0	0	1	0	1	0	0	1	0
1	0	0	1	0	1	0	0	1	0
2	0	0	1	0	1	0	0	0	0
4	0	0	1	0	1	0	0	0	0
5	0	0	1	0	1	0	0	0	0

174 variables after one-hot coding

- Model selection problem: Need to identify the parameters that *really* matter
  - Help interpret results
  - Improves generalization error (less parameters)
- Idea: Fit model under sparsity constraint:
  - Linear model: $\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$
  - Feature x_j is ignored if $\beta_j = 0$
  - Try to force most $\beta_j = 0 \Rightarrow$ Model only uses a few of the variables

Regularized LS Estimation

- Regularization: General method for finding constrained solutions
  - E.g. solutions that are sparse
- Standard least squares estimation (from Lecture 3):

$$\hat{\beta} = \arg\min_{\beta} MSE(\beta), \qquad MSE(\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Regularized estimator:

$$\hat{\beta} = \arg\min_{\beta} J(\beta), \qquad J(\beta) = MSE(\beta) + \phi(\beta)$$

  - $MSE(\beta)$ = mean-squared prediction error from before
  - $\phi(\beta)$ = regularizing function.
- Concept: Regularizer penalizes β that are "unlikely"
  - Constrains estimate to smaller set of parameters

Two Common Regularizers

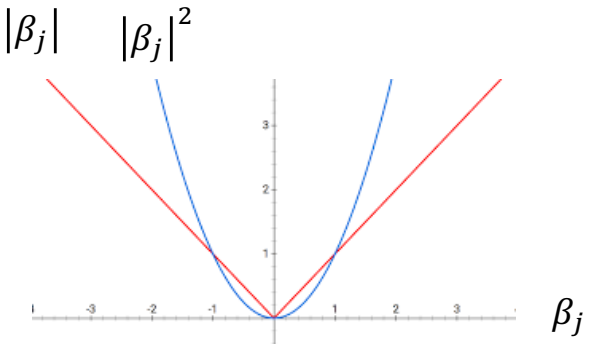
- Ridge regression (called L2)

$$\phi(\beta) = \frac{\alpha}{n}\sum_{j=1}^{d}|\beta_j|^2$$

- LASSO regression (called L1)

$$\phi(\beta) = 2\alpha\sum_{j=1}^{d}|\beta_j|$$

- Coefficient $\alpha > 0$ determines regularization level
  - Higher $\alpha \Rightarrow$ Higher level of reguarlization, more constrainted
  - Will show how to select α later via cross-validation
  - Scaling factors adjust to match sklearn convention
- Both penalize large β_j: Tries to make β_j small
  - Will see that L1 also promotes sparsity
- Convention: Do not include intercept term β_0
  - In general, no reason to make this term small



L1 and L2 Norm

- Ridge and LASSO Regularization can be written with norms

- Ridge cost function:

$$J(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{d}\left|\beta_j\right|^2 = \|\boldsymbol{y} - \boldsymbol{A\beta}\|^2 + \alpha\|\boldsymbol{\beta}\|_2^2$$

- LASSO cost function:

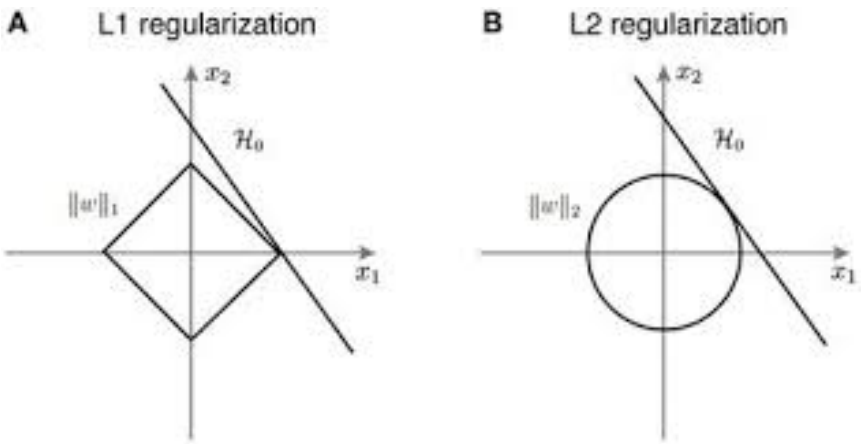
$$J(\boldsymbol{\beta}) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{d}\left|\beta_j\right| = \frac{1}{2n}\|\boldsymbol{y} - \boldsymbol{A\beta}\|^2 + \alpha\|\boldsymbol{\beta}\|_1$$

  - $\|\boldsymbol{\beta}\|_1$ = L1 norm (pronounced ell-1)

Ridge vs LASSO

- L2 tends to lead to many "small" coefficients
  - But solutions are not exactly zero
  - Not ideal for feature selection

- L1 tends to lead to more sparse solutions
  - Several coefficients are zero



Solving Ridge Regression

- Ridge regression problem: Find β to minimize

$$J(\boldsymbol{\beta}) = \|\boldsymbol{y} - A\boldsymbol{\beta}\|^2 + \alpha\|\boldsymbol{\beta}\|^2$$

- Solution for given regularization level

$$\boldsymbol{\beta}_{ridge} = (\boldsymbol{A}^T\boldsymbol{A} + \alpha\boldsymbol{I})^{-1}\boldsymbol{A}^T\boldsymbol{y}$$

  - Set gradient = 0
  - See homework

- Sklearn function for ridge regression:
  - http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

Solving LASSO Regression

- LASSO cost function:

$$J(\boldsymbol{\beta}) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha\sum_{j=1}^{d}|\beta_j| = \frac{1}{2n}\|\boldsymbol{y} - \boldsymbol{A\beta}\|^2 + \alpha\|\boldsymbol{\beta}\|_1$$

- Because derivative of $|\beta_j|$ is not continuous, there is no closed-form solution.
- Many methods to solve iteratively
  - Least angle regression (LAR), coordinate descent, ADMM
  - However, the cost function is convex ⇒ no local minima [See Unit 7]
  - Beyond the scope of this class
  - See textbook [Hastie2008] for LAR method

Data Scaling

- Scaling: Whenever using regularization:
  - Scale each feature and the target to have zero mean and unit variance (or STD)
  - $x_{ij} \rightarrow (x_{i,j} - \bar{x}_j)/\mathrm{STD}(x_{ij})$
  - $y_i \rightarrow (y_i - \bar{y})/\mathrm{STD}(y_i)$
- After predictor for the scaled data are determined:
  - Derive the equivalent predictor on the original data (HW!)
- Motivation:
  - Without scaling, the regularization level depends on the data range
  - With mean removal, we do not need the intercept term β_0
  - So that the regularization term is simply a L2 or L1 norm of coefficient vector

Selecting the Regularization Level

- How do we select regularization level α?
  - Higher α ⇒ More constrained / simpler model
  - Lower α ⇒ More complex model
- Similar to inverse of model order
- Find α via cross-validation

Pseudo-code

Split in training X_{tr}, y_{tr} and test X_{ts}, y_{ts}.

For α in α_{test}:

- $\hat{\beta} = fit(X_{tr}, y_{tr}, \alpha)$ // Fit on training data
- $\hat{y}_{ts} = predict(X_{ts})$ // Predict on test data
- $S[\alpha] = score(y_{ts}, \hat{y}_{ts})$ // Score on test data

$\hat{\alpha} = argmax\ S[\alpha]$ // Select α with highest test score

Summary

Method	Regularizer	Effect on parameters	Solution for Fitting
None	$\phi(\boldsymbol{\beta}) = 0$	Leaves parameters unconstrained	$\widehat{\boldsymbol{\beta}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\,\boldsymbol{A}^T\boldsymbol{y}$
Ridge	$\phi(\boldsymbol{\beta}) = \frac{\alpha}{n}\|\boldsymbol{\beta}\|_2^2$	Makes parameters small Close to zero	$\widehat{\boldsymbol{\beta}} = (\boldsymbol{A}^T\boldsymbol{A} + \alpha\,\boldsymbol{I})^{-1}\,\boldsymbol{A}^T\boldsymbol{y}$
LASSO	$\phi(\boldsymbol{\beta}) = 2\alpha\|\boldsymbol{\beta}\|_1$	Makes parameters sparse. Many coefficients exactly zero	No analytic solution. Need to run an optimizer

- Regularized least squares

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\beta} J(\boldsymbol{\beta}), \qquad J(\boldsymbol{\beta}) = \frac{1}{n}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\beta}\|^2 + \phi(\boldsymbol{\beta})$$

- Whatever you choose for the regularizer:
  - Scale data before training
  - Select regularization level with cross-validation

In-Class Exercise

Question

We wish to fit a linear model of the form,

$$\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p,$$

for a problem with $p = 100$ variables. Suggest regularizers $\phi(\boldsymbol{\beta})$ to impose the following constraints:

(a) All the coefficients $\beta_j, j = 1, \ldots, p$ should be close to zero, but not necessarily exactly zero.

(b) Most of the coefficients $\beta_j, j = 1, \ldots, p$ should be exactly zero

(c) Among the first fifty coefficients, $\beta_j, j = 1, \ldots, 50$, most coefficients should be zero. But, the other coefficients should be uncontrained.

Outline

- Motivating Example: Feature selection in predicting housing prices
- Model selection and regularization
- Housing prices prediction with LASSO
- Probabilistic interpretation
- Other model selection examples
- Other model selection methods
- In-Class Exercise: Audio Pitch Detection

LASSO Regression in Python

- Sklearn built in Lasso class
- Easy to use
  - Set alpha
  - Fit on training data
  - Predict and score on test

```
Test R^2= 0.899122
```

```python
from sklearn.linear_model import Lasso
from warnings import simplefilter
from sklearn.exceptions import ConvergenceWarning
simplefilter("ignore", category=ConvergenceWarning)

# Select alpha
alpha = 3e-3

# Create Lasso object and fit on training data
reg = Lasso(alpha=alpha)
reg.fit(Xtr1, ytr1)

# Predict and score on test
yhat1 = reg.predict(Xts1)
rsq = r2_score(yts1, yhat1)

print('Test R^2= %f' % rsq)
```

Optimizing Alpha via Cross Validation

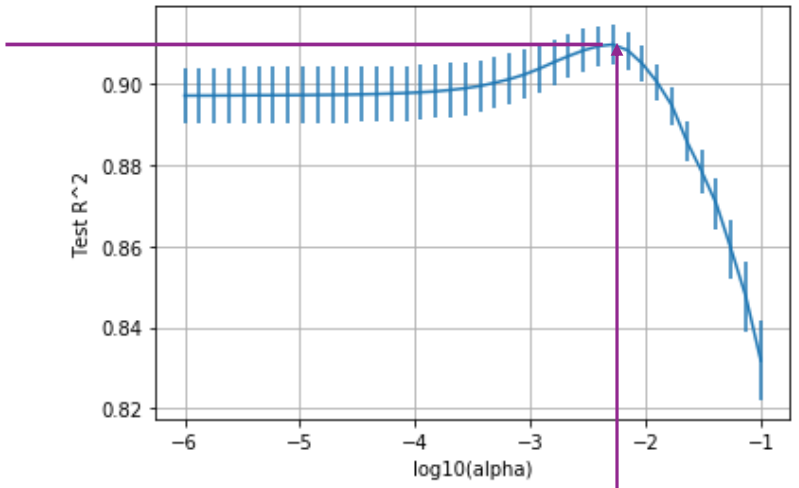
- In each fold we:
  - Split data into training and test
  - Fit the scale on the training
  - Transform training and test
  - For each alpha:
  - Fit training and score on test
- Note: Scaling is redone on each fold
  - Ensures scaling is part of the training

```
# Run the cross-validation
rsq = np.zeros((nalpha, nfold))
for ifold, ind in enumerate(kf.split(X)):

    # Get the training data in the split
    Itr,Its = ind
    Xtr = X[Itr,:]
    ytr = y[Itr]
    Xts = X[Its,:]
    yts = y[Its]

    # Fit and transform the data
    Xtr1 = xscal.fit_transform(Xtr)
    Xts1 = xscal.transform(Xts)
    ytr1 = yscal.fit_transform(ytr[:,None])
    yts1 = yscal.transform(yts[:,None])

    for i, alpha in enumerate(alphas):

        # Fit on the training data
        reg = Lasso(alpha=alpha)
        reg.fit(Xtr1, ytr1)

        # Score on the test data
        yhat1 = reg.predict(Xts1)
        rsq[i, ifold] = r2_score(yts1, yhat1)

    print('Fold = %d' % ifold)

# Compute mean and SE
rsq_lasso_mean = np.mean(rsq, axis=1)
rsq_lasso_se  = np.std(rsq, axis=1) / np.sqrt(nfold-1)
```

Cross Validation: Normal Rule

- Select alpha to maximize mean test R^2
  - Normal rule
  - Lower values of $\alpha \Rightarrow$ overfit
  - Higher values of $\alpha \Rightarrow$ underfit

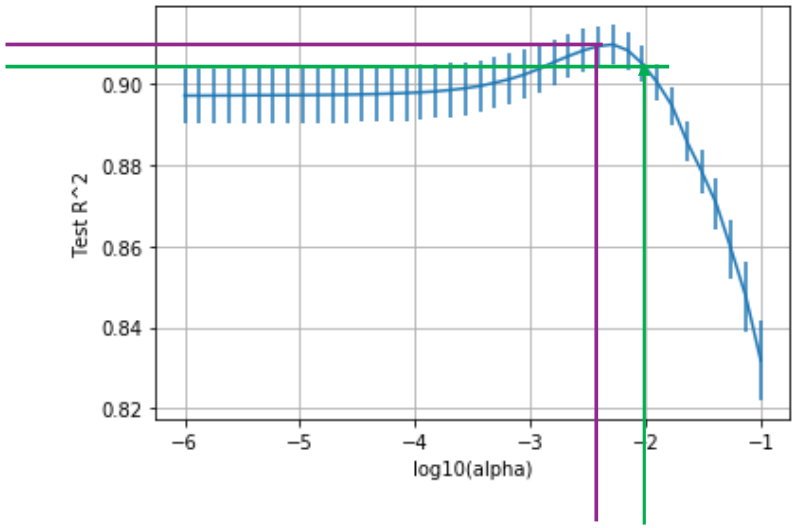
```
Alpha optimal (normal rule) =   5.2233e-03
Mean test R^2 (normal rule) =   0.910
```



Cross Validation: One SE Rule

Max mean test R^2

Max mean test R^2-one SE



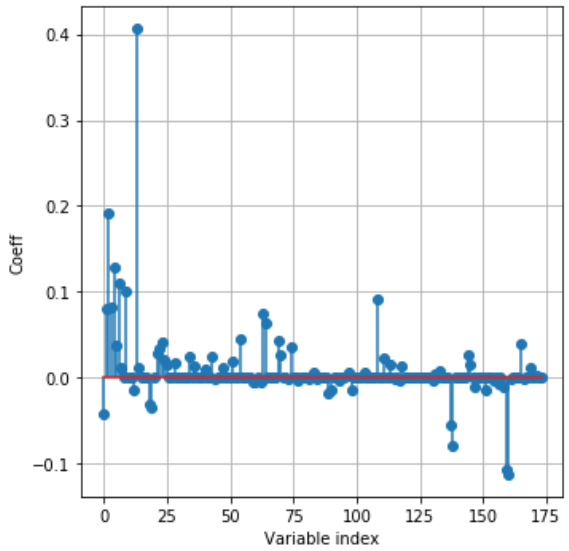
Alpha with normal rule

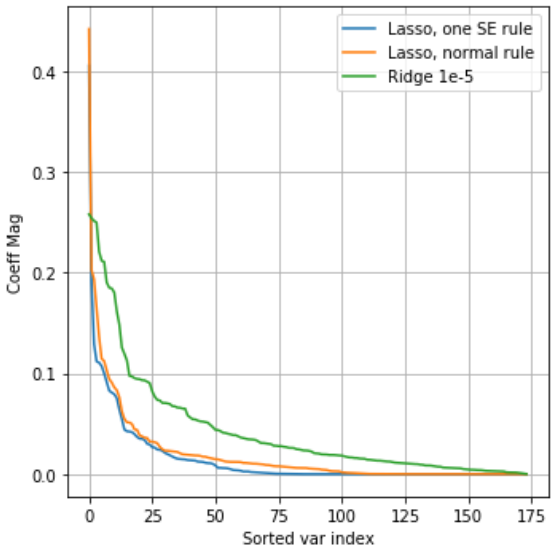
Alpha with one SE rule

- Can also use one SE rule:
  - Selects a higher regularized model
  - More sparse solution

Sparsity in the Coefficients

- Adding L1 regularization:
  - Makes coefficient smaller
  - Many coefficients approx. 0





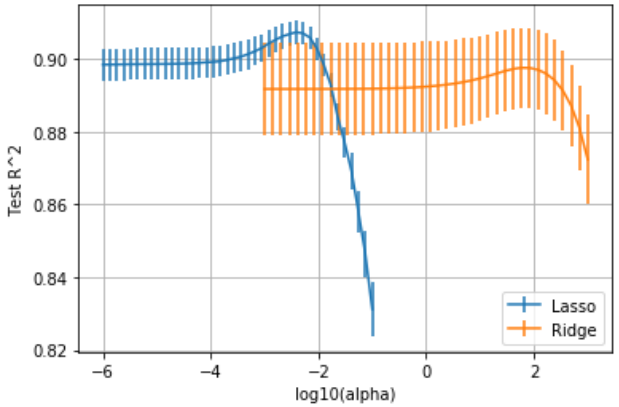
Most Important Variables

- Right table: Variables with 10 large coefficient magnitudes
- Minimally regularized (Ridge) has:
  - Variables that are highly correlated
  - Ex: GrLivArea and 2ndFlrSF
  - Several large variables
- Lasso:
  - Reduces correlated variables
  - Selects GrLivArea alone
  - Gives the variables more importance

Ridge		Lasso	
GrLivArea	0.29	GrLivArea	0.42
2ndFlrSF	0.26	OverallQual	0.18
KitchenQual_Gd	-0.21	KitchenQual_TA	-0.17
KitchenQual_TA	-0.20	KitchenQual_Gd	-0.16
LotArea	0.18	YearBuilt	0.13
YearBuilt	0.16	BsmtFinSF1	0.12
OverallQual	0.16	Neighborhood_NoRidge	0.09
ExterQual_Gd	-0.15	OverallCond	0.09
Exterior2nd_VinylSd	0.15	TotalBsmtSF	0.09
ExterQual_TA	-0.15	LotArea	0.08

Ridge Vs. Lasso

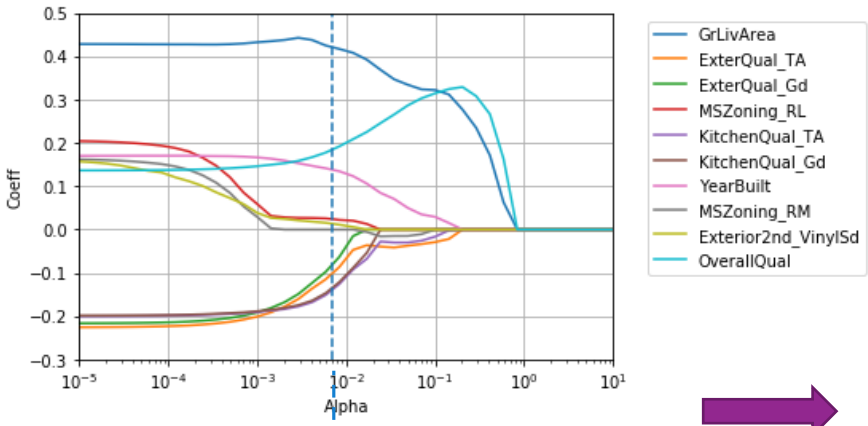
- Can optimize alpha for both regularizer
- Optimal mean test R^2 is better for LASSO
- Offers better feature selection




```
Optimal R^2 Lasso:  0.907283
Optimal R^2 Ridge:  0.897498
```

Lasso Path

- Plot of coefficients vs. alpha
- For large alpha:
  - All coefficients are zero
- As alpha is decreased:
  - One coefficient is activated at a time
  - Indicates an ordering of importance



Increasing Regularization

Optimal Alpha
(one SE rule)

Finding the Final Regressor

- Select features from cross-validation
- Re-run ordinary (un-regularized) regression on reduced features
- Use K −fold validation
- K-folds yield K weights and biases
- Take mean of the weights and biases for the final parameter estimate
- Take mean of the test MSE for the estimate of the test MSE

Outline

- ❑Motivating Example: Feature selection in predicting housing prices
- ❑Model selection and regularization
- ❑Housing prices prediction with LASSO
- ❑Probabilistic interpretation
- ❑Other model selection examples
- ❑Other model selection methods
- ❑In-Class Exercise: Audio Pitch Detection

Maximum Likelihood Estimate

- Suppose that true data generated from probabilistic model with Gaussian noise:

$$\boldsymbol{y} = A\boldsymbol{\beta} + \boldsymbol{w}, \qquad w_i \sim N(0, \sigma^2)$$

- Maximum likelihood estimator:

$$\widehat{\boldsymbol{\beta}} = \arg\max_{\beta} p(\boldsymbol{y}|A, \boldsymbol{\beta}) = \arg\min_{\beta}[-\ln p(y|A, \boldsymbol{\beta})]$$

- Gaussian density for noise in y: $\ln p(\boldsymbol{y}|A, \boldsymbol{\beta}) = -\frac{1}{2\sigma^2} \|\boldsymbol{y} - A\boldsymbol{\beta}\|^2$

- Hence

$$\widehat{\boldsymbol{\beta}} = \arg\max_{\beta} p(\boldsymbol{y}|A, \boldsymbol{\beta}) = \arg\min_{\beta}[\|\boldsymbol{y} - A\boldsymbol{\beta}\|^2] = \text{Least Squares Solution}$$

Bayes Estimation (MAP Estimate)

- Maximum a posterior (MAP) estimator of $\boldsymbol{\beta}$:

$$\widehat{\boldsymbol{\beta}} = \arg max_{\beta}\, p(\boldsymbol{\beta}|\boldsymbol{y}, A)$$

  - $\hat{\beta}$ = Most likely parameter value given evidence y, A

- Bayes Rule: $p(\boldsymbol{\beta}\,|\boldsymbol{y}, A) = p(\boldsymbol{y}|A, \boldsymbol{\beta}\,)p(\boldsymbol{\beta})/p(\boldsymbol{y}|A)$

- Hence: $\widehat{\boldsymbol{\beta}} = \arg\max_{\beta} p(y|A, \boldsymbol{\beta}\,)p(\boldsymbol{\beta})$ (because $\boldsymbol{y}$ and A are fixed)

  - Likelihood: $p(y|A, \boldsymbol{\beta}\,)$ How well $\boldsymbol{\beta}$ matches data
  - Prior: $p(\boldsymbol{\beta}\,)$: How well $\boldsymbol{\beta}$ agrees with prior knowledge about its distribution (constraints)

- More in probability class…

Bayes Estimation with Logarithms

- Often easier to use logarithms:

$$\widehat{\boldsymbol{\beta}} = \arg\max_{\beta} p(\boldsymbol{y}|A,\boldsymbol{\beta})p(\boldsymbol{\beta}) = \arg\min_{\beta}[-\ln p(\boldsymbol{y}|A,\boldsymbol{\beta})p(\boldsymbol{\beta})]$$
$$= \arg\min_{\beta}[-\ln p(\boldsymbol{y}|A,\boldsymbol{\beta}) - \ln p(\boldsymbol{\beta})]$$

- Gaussian density for noise in y: $\ln p(\boldsymbol{y}|A,\boldsymbol{\beta}) = -\frac{1}{2\sigma^2}\|\boldsymbol{y} - A\boldsymbol{\beta}\|^2$

- Hence

$$\hat{\beta} = \arg\min_{\beta}\left[\frac{1}{2\sigma^2}\|\boldsymbol{y} - A\boldsymbol{\beta}\|^2 - \ln p(\boldsymbol{\beta})\right] = \arg\min_{\beta}[\|\boldsymbol{y} - A\boldsymbol{\beta}\|^2 + \phi(\boldsymbol{\beta})]$$

- Conclusion: MAP estimate = regularized LS with $\phi(\boldsymbol{\beta}) = -2\sigma^2 \ln p(\boldsymbol{\beta})$
  - Penalize $\boldsymbol{\beta}$ proportional to $-\ln p(\boldsymbol{\beta})$: Less likely $\boldsymbol{\beta}$ penalized more

Ridge and Lasso as Bayesian Estimators

- Bayesian Estimator:

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\beta}\left[\frac{1}{2\sigma^2}\|\boldsymbol{y} - A\,\boldsymbol{\beta}\|^2 - \ln p(\boldsymbol{\beta})\right]$$

- Assuming β_j are i.i.d. Gaussian with zero mean:

$$p(\beta_j) = \tfrac{1}{2\pi\sigma} exp\left(-{\beta_j}^2/2\gamma^2\right),\ -\log p(\beta_j) = {\beta_j}^2/2\gamma^2 + constants$$

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\beta}\left[\|\boldsymbol{y} - A\boldsymbol{\beta}\|^2 + \tfrac{\sigma^2}{\gamma^2}\|\,\boldsymbol{\beta}\|^2\right] \quad = \text{Ridge Regression!}$$

- Assuming β_j are i.i.d. Laplacian with zero mean:

$$p(\beta_j) = \tfrac{1}{2\sigma} exp\left(-\left|\beta_j\right|/\gamma\ \right),\ -\log p(\beta_j) = \left|\beta_j\right|/\gamma\ + constant$$

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\beta}\left[\|\boldsymbol{y} - A\boldsymbol{\beta}\ \|^2 + \tfrac{2\sigma^2}{\gamma}\|\boldsymbol{\beta}\|_1\right] \quad = \text{Lasso Regression!}$$

Outline

- ❑Motivating Example: Feature selection in predicting housing prices
- ❑Model selection and regularization
- ❑Housing prices prediction with LASSO
- ❑Probabilistic interpretation
- ❑Other model selection examples
- ❑Other model selection methods
- ❑In-Class Exercise: Audio Pitch Detection

Example 1: Medical Modeling

- Ex: Prostate specific antigen (PSA) test
  - Many years ago, PSA level was being consider for cancer screening
  - Question: Is a PSA test good for cancer?
  - Obtain features of prostate and correlate with PSA level
  - Determine if cancer volume is a relevant feature
  - See demo 1 on github site
  - Also in text

The data frame has the following components:

`lcavol`
: log(cancer volume)

`lweight`
: log(prostate weight)

`age`
: age

`lbph`
: log(benign prostatic hyperplasia amount)

`svi`
: seminal vesicle invasion

`lcp`
: log(capsular penetration)

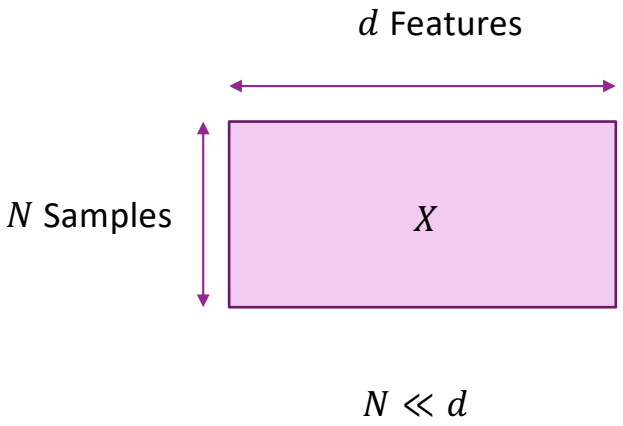
`gleason`
: Gleason score

`pgg45`
: percentage Gleason scores 4 or 5

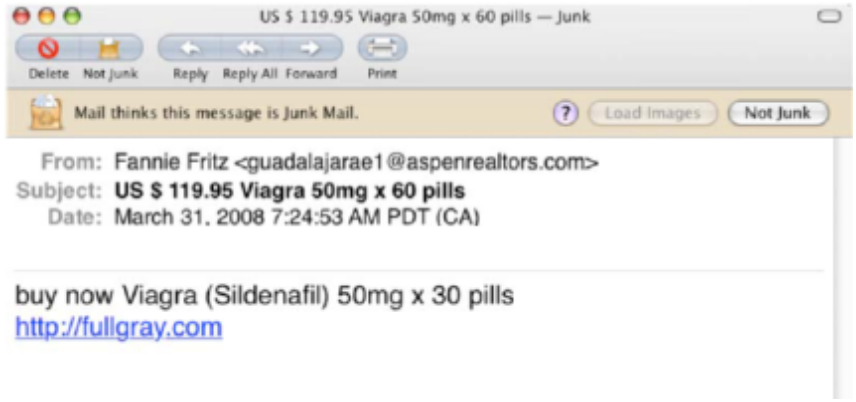
`lpsa`
: log(prostate specific antigen)

Ex 2: Model Selection with Limited Data

- Model selection is particularly valuable when data is limited
- Ex: Consider linear model: $\hat{y} = b + w_1 x_1 + \cdots + w_d x_d$
  - Model has $d + 1$ parameters
- From previous lecture, we need $N > d + 1$ data points $(\boldsymbol{x}_i, y_i)$
- In many cases we have $N \ll d$
  - Examples below
  - Many few data points than features
  - Classic linear fit will not work
- But, suppose we can restrict to $K \ll N$ non-zero parameters
  - Then, we can find a good fit on those parameters
- Challenge: How do we find a small number K of relevant features



Example 3: Spam Detection



- Classification problem:
  - Is email junk or not junk?
- Typical bag-of-word model:
  - Enumerate all words, $i = 1, \ldots, d$
  - Represent email via word count x_i = num instances of word i
- Model selection:
  - d =vocabulary size is typically very large
  - But, only a few words are likely relevant
  - Want to find $K \ll d$ relevant words

Example 4: EEG

- EEG: Electroencephalography
- Measure brain activity from electrodes on scalp
- Source localization problem:
  - Find brain region responsible for evoked response
- Problem:
  - Many possible brain regions
    Typically use d >10,000 voxels
  - But, limited number of measurements:
    100s of electrodes
  - Cannot fit a model from all brain regions
- Model selection:
  - We know that responses are likely from a small brain region
  - Find a small number of voxels that explain response
  - See lab!

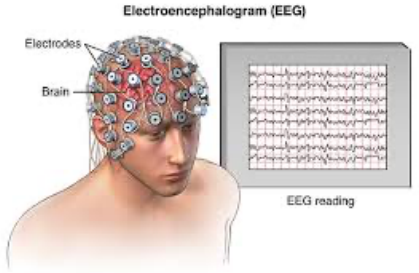
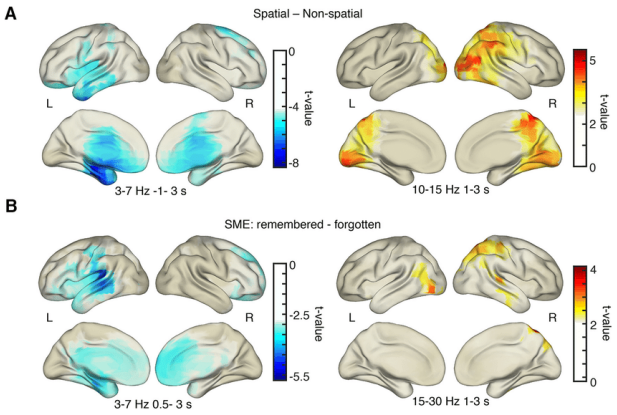
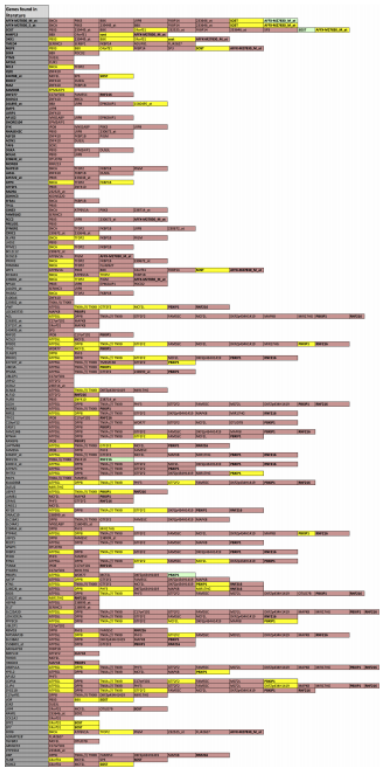


Image: mayoclinic.org



Example 5: DNA MicroArray Data

- Basic genetic problem
  - Which genes determine some characteristic (i.e. phenotype)?
- DNA microarrays:
  - Measure "expression" levels of large numbers of genes
  - Expression levels = amount of protein produced by gene
- Data modeling:
  - Fit phenotype to expression levels
  - Usually have large numbers of genes $(d \sim 1000)$
  - But, small number of data points $(n \sim 100)$
  - We know only a small number of genes are responsible
  - So, we can use model selection



Outline

- Motivating Example: Feature selection in predicting housing prices
- Model selection and regularization
- Housing prices prediction with LASSO
- Probabilistic interpretation
- Other model selection examples
- Other model selection methods
- In-Class Exercise: Audio Pitch Detection

Other feature selection methods

- Filtering method:
  - Rank the features based on their correlation or mutual information with the target and possibly the redundancy among the features
  - Simple but not very good
- Wrapper method:
  - For each candidate feature subset, apply a chosen classifier/regressor, evaluate the cross validation accuracy. Go through all possible feature subsets, or test the subsets in some greedy way
  - Computationally expensive
- Embedded method:
  - Some regression/classification method naturally lead to feature ranking and selection
- What is available in Python:
  - http://scikit-learn.org/stable/modules/feature_selection.html

Filtering method

- Rank the features based on their correlation with the target
  - Can use other metrics: Correlation, F-test, mutual information, …
- Also should consider the redundancy (correlation) among chosen features
  - Minimal Redundancy Maximum Relevance (mRMR)
  - Peng, H.C., Long, F., and Ding, C., "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, No. 8, pp. 1226–1238, 2005.
  - http://home.penglab.com/proj/mRMR/
  - https://www.mathworks.com/matlabcentral/fileexchange/14916-minimum-redundancy-maximum-relevance-feature-selection

Ranking metrics

- Correlation coefficient between a feature and the target

- F-test: test the significance of using one feature vs. not using any (use the mean of y only. Essentially measure the difference in the MSE when using only the mean value of y vs. using a single feature.

$$ftest = \frac{r^2}{1-r^2}(\text{nsample-2})$$

- Mutual information between a feature and the target

$$I(X,Y) = \iint p(x,y) log \frac{p(x,y)}{p(x)p(y)} dxdy$$

Embedded Method

- Results from some regression/classification methods allow feature selection
  - Linear regression: based on coefficient magnitude
  - Neural net: based on weight magnitude
  - Decision tree: based on tree level
  - Can add regularization terms on the coefficients/weights to encourage sparsity
    - LASSO regression
- Recursive feature elimination
  - Starting with all features, remove one feature that has the lowest importance (e.g. smallest coefficient magnitude)
  - Recursive feature elimination in sklearn
    - http://scikit-learn.org/stable/auto_examples/feature_selection/plot_rfe_digits.html#sphx-glr-auto-examples-feature-selection-plot-rfe-digits-py
    - http://scikit-learn.org/stable/auto_examples/feature_selection/plot_rfe_with_cross_validation.html#sphx-glr-auto-examples-feature-selection-plot-rfe-with-cross-validation-py

Wrapper method

- For each candidate feature subset, apply a chosen classifier/regressor, evaluate the cross validation accuracy. Go through all possible feature subsets, or test the subsets in some greedy way
  - Exhaustive search
  - Genetic algorithm
  - Forward stepwise
  - Backward stepwise

Exhaustive search for feature selection

- Suppose you want to consider feature subset of size up to p
- For all possible feature subsets of size 1 to p:
  - use cross validation to find mean RSS mean and standard deviation
- Choose the subset with the minimal RSS mean,
  - Or use the one standard error rule.
- When the number of features is large, may not be computationally feasible
- Fast search algorithms:
  - Genetic algorithm

Greedy feature selection

- Forward-Stepwise Selection
  - Select one feature from all features that provides the lowest RSS with cross validation
  - Select one new feature from all remaining features, so that previously chosen features plus the new feature provides the lowest RSS
  - Repeat until the maximum feature number is reached, or when the RSS starts to increase
- Backward-Stepwise
  - First use all features and find the RSS (using cross validation)
  - Remove one feature and find the new RSS. Go through all possible features to remove.
  - Find the one that leads to the least RSS increase. Remove this feature.
  - Repeat the above, remove one from the remaining features, to find the next most important feature.
- Except exhaustive search, can all lead to suboptimal solution

Comparison of feature selection methods

Figure from [Hastie2008]: Hastie, Tibshirani, Friedman, The elements of statistical learning.

For more on this subject, see Sec. 3.3

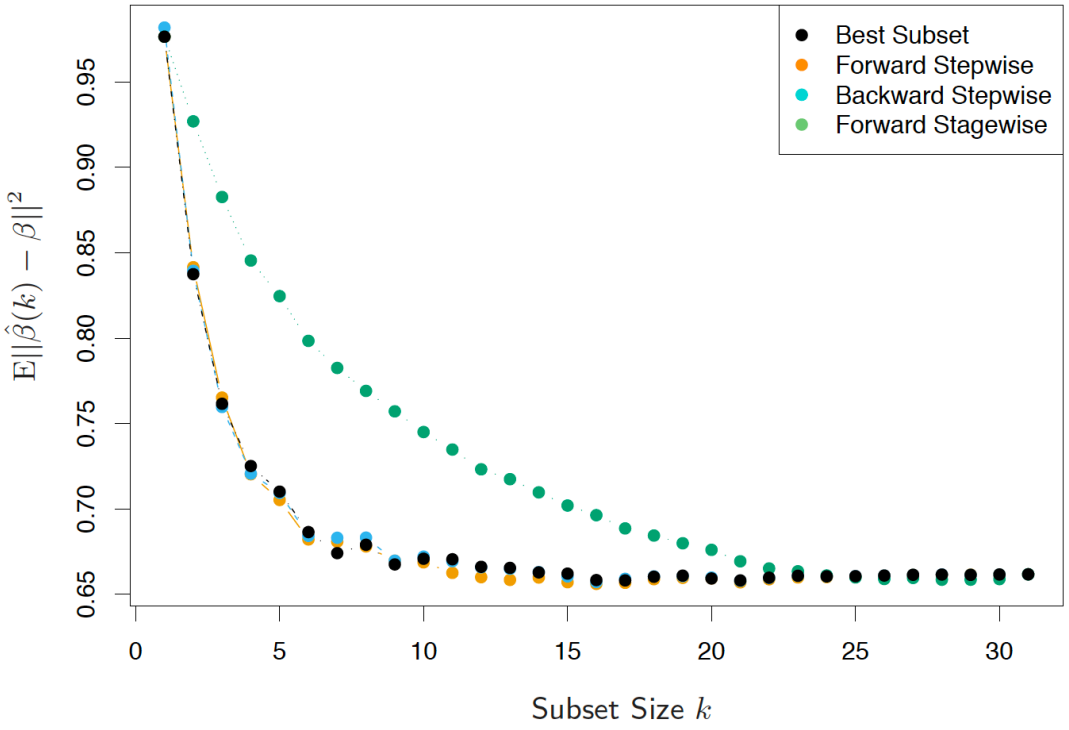


FIGURE 3.6. *Comparison of four subset-selection techniques on a simulated linear regression problem* $Y = X^T\beta + \varepsilon$. *There are* $N = 300$ *observations on* $p = 31$ *standard Gaussian variables, with pairwise correlations all equal to* 0.85. *For* 10 *of the variables, the coefficients are drawn at random from a* $N(0, 0.4)$ *distribution; the rest are zero. The noise* $\varepsilon \sim N(0, 6.25)$, *resulting in a signal-to-noise ratio of* 0.64. *Results are averaged over* 50 *simulations. Shown is the mean-squared error of the estimated coefficient* $\hat{\beta}(k)$ *at each step from the true* β.

More about cross validation

- Why do we use cross validation?
  - To estimate the test error when there are insufficient training data so that we can partition the total data to a **large** training set and a **large** test set.
  - Whether a dataset is large depends on the number of parameters of the model to be trained.
  - Ideally the number of samples should be >100x of the number of parameters, but at least 10x.
- When you have sufficient training data, you can just use a certain percentage (e.g. 50%) for training and remaining for testing. The error on the testing set would be a reliable estimate of the test error.
- Two ways of using cross validation
  - When the "best" model class, model order, and feature set are known:
    - Use CV to estimate the test error
  - Use CV to determine the appropriate model class, model order and feature subset
    - For each candidate model class, model order, and feature subset, evaluate CV error
    - Determine which candidate yields the least CV error.

More about cross validation

- How to use the multiple estimated models from multiple trials?
  - Apply each on a test sample and take the average (for regression) or majority (for classification) of results
  - For linear regression, equivalent to average the model coefficients
- When your data is limited, you may want to go beyond K-folds
  - Ex: 5-fold means that you partition the data to 5 parts in some way, each part has 20% of data, and only do 5 fold training and testing
  - When your data is small, the average CV error is still very sensitive to how the data is partitioned to 5 parts. If you use random shuffling, you will get different result each time.
  - Instead, you could do L trials (L>>5) of random sampling, each time using 80% for training and 20% for testing
- How to handle limited data in machine learning is still a challenging topic!

Outline

- ❑Motivating Example: Predicting prostate cancer from a PSA test
- ❑Model Selection
- ❑Model Selection from LASSO regularization
- ❑Probabilistic interpretation
- ❑Other Model Selection Methods
- ❑In-Class Exercise: Audio Pitch Detection

In Class Exercise

- https://github.com/pliugithub/MachineLearning/tree/master/ unit05_lasso/lasso_in_class.ipynb

LASSO Regression In-Class Exercise

In this exercise, we will see how to use LASSO for pitch detection in audio.

We load the following packages.

```
import numpy as np
import matplotlib.pyplot as plt
import pickle
```

Load the data

The data is taken from a sample of about 20~ms of audio from a viola. I have already pre-processed the data. You can load it with following command. The value `t` is the time (in seconds) and `y` is the sample of audio (this is a mono recording).

```
fn_src = 'https://raw.githubusercontent.com/sdrangan/introml/master/unit05_lasso/viola_sample.p'
fn_dst = 'viola_sample.p'

import os
from six.moves import urllib

if os.path.isfile(fn_dst):
```

What You Should Know to Do

- ❑ Formulate a linear estimation problem with a regularization
- ❑ Compute an L1-regularized estimate (LASSO) using sklearn tools
- ❑ Compute the optimal regularization level using cross validation
- ❑ Interpret results from a LASSO path
- ❑ Determine final regression function from cross validation
- ❑ Set regularizer based on a probabilistic prior
- ❑ Perform other feature selection methods