

Lecture 5

LASSO Regularization

EE-UY 4563/EL-GY 9123: INTRODUCTION TO MACHINE LEARNING

PROF. SUNDEEP RANGAN

Learning Objectives

- ❑ 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
- ❑ Set regularizer based on a probabilistic prior

Outline



Motivating Example: Predicting prostate cancer from a PSA test

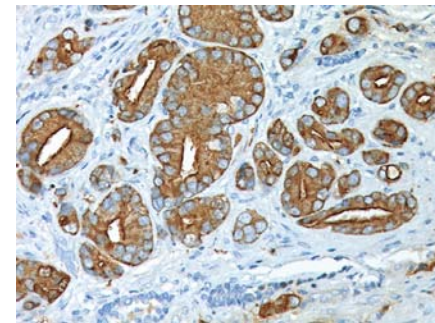
- Model selection from LASSO regularization

- Probabilistic interpretation

Prostate Specific Antigen Testing

- ❑ PSA levels easily tested
- ❑ High PSA believed to be associated with prostate cancer
 - Potential tool for screening
- ❑ Classic 1989 study by Thomas et al:
 - Measured PSA level of 102 men prior to prostate removal
 - Measured characteristics of prostate from samples
 - Characteristics include cancer volume, weight, ...
- ❑ Data analysis:
 - What characteristics predict PSA?

Stamey, Thomas A., et al. "[Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. II. Radical prostatectomy treated patients.](#)" The Journal of urology 141.5 (1989): 1076-1083.



Data

- ❑ Prostate dataset widely-used in ML classes
- ❑ Can be downloaded from many sites
- ❑ Samples = 97 patients
- ❑ 8 features of the prostate
- ❑ Target variable = lpsa (log PSA)

```
# Get data
url = 'https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data'
df = pd.read_csv(url, sep='\t', header=0)
df = df.drop('Unnamed: 0', axis=1) # skip the column of indices
```

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)

First Try: Linear Model

□ Simple idea: Use linear regression

$$y \approx \hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

- $y = \text{lpsa}$ (target PSA level)
- $x_1, \dots, x_d = \text{prostate features}$ ($d = 8$)

□ Why linear regression?

- Easy to compute / interpret
- Coefficients are easy to interpret
- Larger coefficients \Rightarrow larger influence of feature on PSA

```
: ns_train = nsamp // 2
  ns_test = nsamp - ns_train
  X_tr = X[:ns_train,:]      # Gets the first ns_train rows of X
  y_tr = y[:ns_train]        # Gets the corresponding rows of y

  print("num samples train = %d, test = %d" % (ns_train, ns_test))

num samples train = 48, test = 49
```

```
regr = linear_model.LinearRegression()
regr.fit(X_tr, y_tr)
```

Model Does Not Generalize

❑ Evaluate model with cross validation

- Train on 48 samples
- Measure RSS on 49 samples

❑ Test RSS is very high

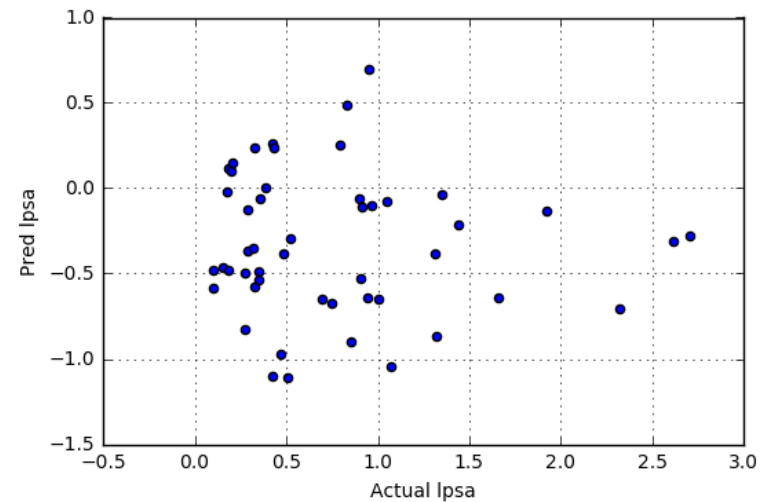
❑ Scatter plot shows no predictive ability

❑ What happened?


❑ Can we do a better model?

```
X_ts = X[ns_train,:]  
y_ts = y[ns_train:]  
y_ts_pred = regr.predict(X_ts)  
RSS_rel_ts = np.mean((y_ts_pred-y_ts)**2)/(np.std(y_ts)**2)  
print("Normalized test RSS = {0:f}".format(RSS_rel_ts))
```

Normalized test RSS = 4.539225



Outline

- Motivating Example: Predicting prostate cancer from a PSA test
-  □ Model selection from LASSO regularization
- Probabilistic interpretation

Intuition

- ❑ We know from last lecture:
 - Too many parameters \Rightarrow Large generalization error
- ❑ In this data set, only a few factors are likely significant
- ❑ But, we don't know which one
- ❑ Can we automatically identify them?

- ❑ **Idea:** Fit model under constraint:
 - Force only a few parameters to be non-zero
- ❑ General idea of **regularization**:
 - Constrain the parameters with prior knowledge

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)
```

Regularized LS Estimation

□ Standard least squares estimation (from Lecture 3):

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

□ Regularized estimator:

$$\hat{\beta} = \arg \min_{\beta} J(\beta), \quad J(\beta) = RSS(\beta) + \phi(\beta)$$

- $RSS(\beta)$ = 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)

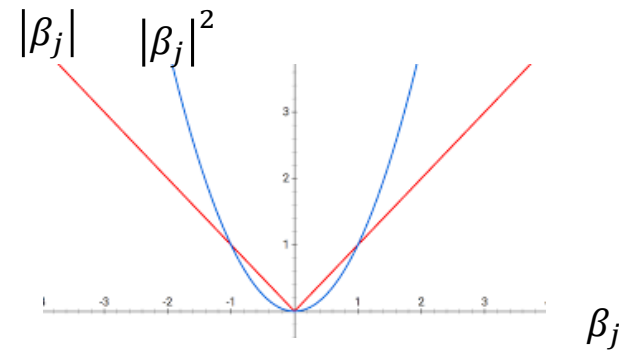
$$\phi(\beta) = \alpha \sum_{j=1}^d |\beta_j|^2$$

□ LASSO regression (called L1)

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

□ Both penalize large β_j

□ Level of regularization controlled by α



L1 and L2 Norm

□ Ridge and LASSO regularized estimated are often written with vector norms

□ Ridge cost function:

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

□ LASSO cost function:

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

◦ $\|\beta\|_1$ = L1 norm (pronounced ell-1)

Ridge vs LASSO

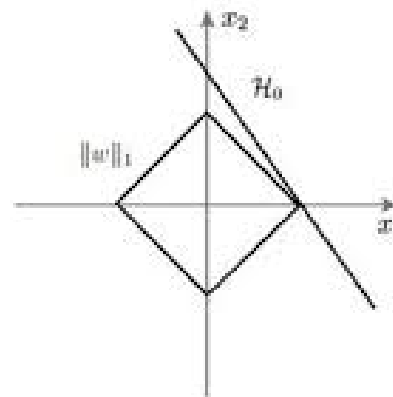
□ Optimization can be easily performed for L1 and L2 regularizers

- Regularizer is convex
- More on this later

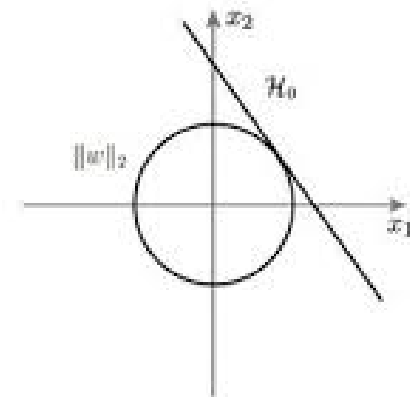
□ L1 tends to lead to more sparse solutions

- More coefficients are zero
- Will focus this lecture on L1

A L1 regularization



B L2 regularization

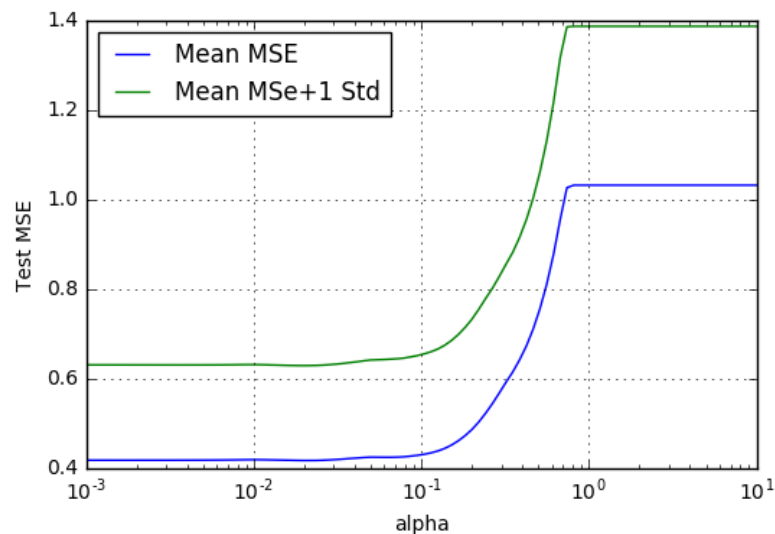


Selecting Regularization Level

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

Computing LASSO in python

- ❑ Use sklearn Lasso method
- ❑ Cross validation loop
 - Outer loop: Loop over folds
 - Inner loop: Loop over α
 - Measure mean and std deviation of MSE



```
: # Create a k-fold cross validation object
nfold = 10
kf = sklearn.model_selection.KFold(n_splits=nfold, shuffle=True)

# Create the LASSO model. We use the `warm start` parameter so
# This speeds up the fitting.
model = linear_model.Lasso(warm_start=True)

# Regularization values to test
nalpha = 100
alphas = np.logspace(-3, 1, nalpha)

# MSE for each alpha and fold value
mse = np.zeros((nalpha, nfold))
for ifold, ind in enumerate(kf.split(X)):

    # Get the training data in the split
    Itr, Its = ind
    X_tr = X[Itr, :]
    y_tr = y[Itr]
    X_ts = X[Its, :]
    y_ts = y[Its]

    # Compute the Lasso path for the split
    for ia, a in enumerate(alphas):

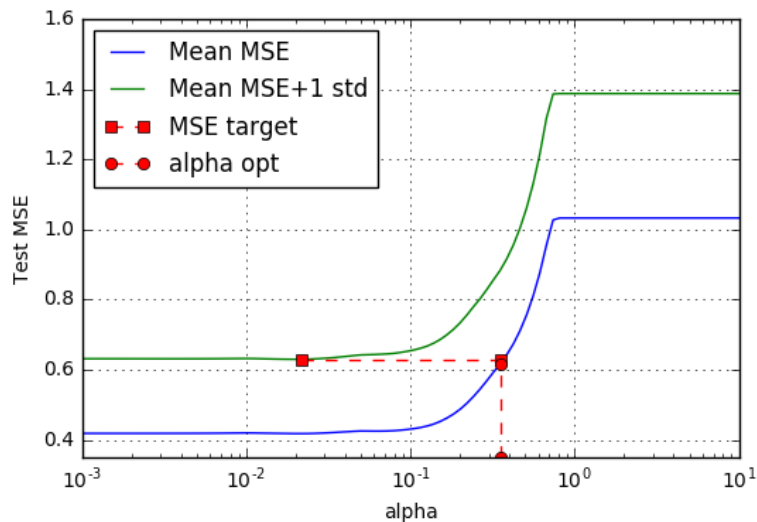
        # Fit the model on the training data
        model.alpha = a
        model.fit(X_tr, y_tr)

        # Compute the prediction error on the test data
        y_ts_pred = model.predict(X_ts)
        mse[ia, ifold] = np.mean((y_ts_pred - y_ts)**2)
```



Using One Standard Deviation Rule

- Use one standard deviation rule from before
 - Find α_0 with minimum mean MSE, mean_mean
 - Set $\text{mse_tgt} = \text{mse_mean}[\alpha_0] + \text{mse_std}[\alpha_0]$
 - Find largest α where $\text{mse_mean}[\alpha] < \text{mse_tgt}$



```
# Find the minimum MSE and MSE target
imin = np.argmin(mse_mean)
mse_tgt = mse_mean[imin] + mse_std[imin]
alpha_min = alphas[imin]

# Find the least complex model with mse_mean < mse_tgt
I = np.where(mse_mean < mse_tgt)[0]
iopt = I[-1]
alpha_opt = alphas[iopt]
print("Optimal alpha = %f" % alpha_opt)
```


Coefficients

- ❑ Select α via cross-validation
- ❑ Then, find coefficients using all training data.
- ❑ Final coefficients are sparse:
 - Only two factors are non-zeros
 - Lcavol: log cancer volume

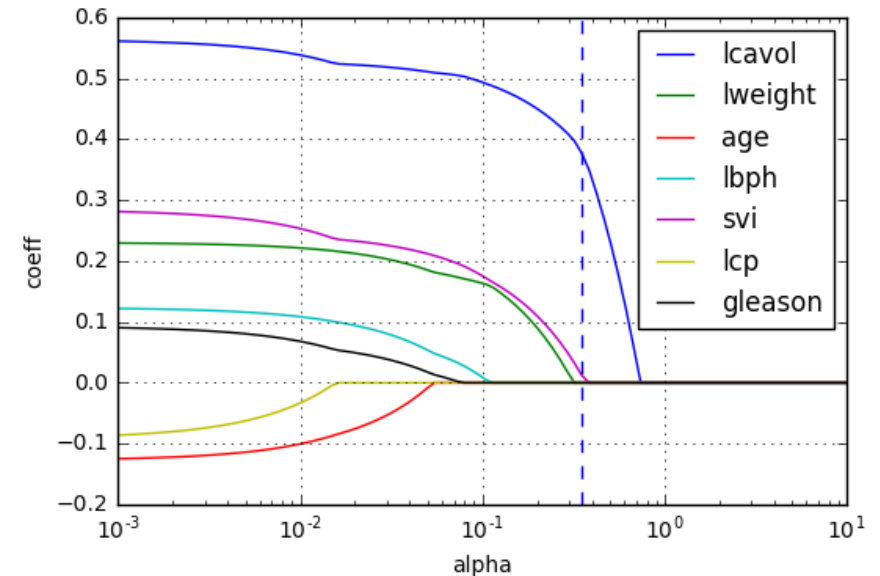
```
model.alpha = alpha_opt
model.fit(X,y)

# Print the coefficients
for i, c in enumerate(model.coef_):
    print("%8s %f" % (names_x[i], c))


lcavol 0.376872
lweight 0.000000
age 0.000000
lbph 0.000000
svi 0.012024
lcp 0.000000
gleason 0.000000
```

LASSO path

- Useful to plot coefficients as a function of α .
- Called the LASSO path
- Indicates relative importance of different factors
- For this data set:
 - lcavol most important
- Don't draw medical conclusions
 - Need more detailed significance testing
 - Complex subject for another class...



Outline

- Motivating Example: Predicting prostate cancer from a PSA test
- Model selection from LASSO regularization
-  Probabilistic interpretation

Bayesian Estimation

- Suppose that true data generated from probabilistic model:

$$y = A\beta + w, \quad w \sim N(0, \sigma^2 I)$$

- Consider **maximum a posterior (MAP)** estimator of β :

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

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

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

- Hence: $\hat{\beta} = \arg \max_{\beta} p(y|A, \beta)p(\beta)$

- **Likelihood:** $p(y|A, \beta)$ How well β matches data
- **Prior:** $p(\beta)$: How well β agrees with prior knowledge / constraints

- More in probability class...

Bayes Estimation with Logarithms

□ Often easier to use logarithms:

$$\begin{aligned}\hat{\beta} &= \arg \max_{\beta} p(y|A, \beta)p(\beta) = \arg \min_{\beta} [-\ln p(y|A, \beta)p(\beta)] \\ &= \arg \min_{\beta} [-\ln p(y|A, \beta) - \ln p(\beta)]\end{aligned}$$

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

□ Hence

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

□ **Conclusion:** MAP estimate = regularized LS with $\phi(\beta) = -2\sigma^2 \ln p(\beta)$

- Penalize β proportional to $-\ln p(\beta)$
- Less likely β penalized more