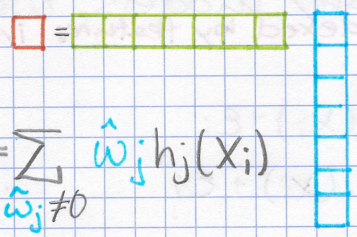
Lasso Regression
Regularization for feature selection

Why might you want to perform feature selection?

Efficiency:

- If size(w) = 100B, each prediction is expensive
- If $\hat{w}$ sparse, computation only depends on # of non-zeros

many zeros

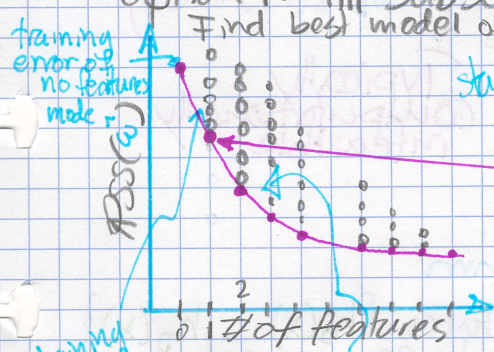


$$y_i = \sum_{\hat{w}_j \neq 0} \hat{w}_j h_j(x_i)$$

Interpretability:

- Which features are relevant for predictions?

Option 1: All subsets



Not necessarily nested

- # bedrooms
- # bathrooms
- sq. ft living
- sq. ft lot
- floors
- year built
- year renovated
- waterfront

search over all combinations of 2 features
#bed
#bath

Best model of size K need not contain features of best model of size K-1

Choosing model complexity?

Option1: Assess on validation set

Option2: Cross validation

Option3+: Other metrics for penalizing model complexity like BIC

Complexity of "all subsets"

How many models were evaluated?

- each indexed by features included

0 if "no" feature included
1 if "yes" feature included

feature 1 feature 2 ... feature D

$y_i = \varepsilon_i$ [000...000]

$y_i = w_0 h_0(x_i) + \varepsilon_i$ [100...000]

$y_i = w_1 h_1(x_i) + \varepsilon_i$ [010...000]

$\vdots$

$y_i = w_0 h_0(x_i) + w_1 h_1(x_i) + \varepsilon_i$

$\vdots$

$y_i = w_0 h_0(x_i) + w_1 h_1(x_i) + \cdots + w_D h_D(x_i) + \varepsilon_i$

2^{D+1}

Typically computationally infeasible

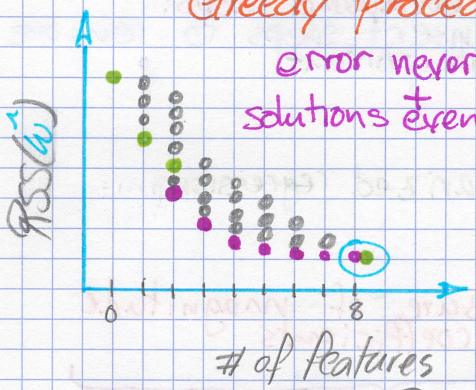
Option2:

Greedy algorithms

Forward stepwise algorithm

1. Pick a dictionary of features $\{h_0(x), \ldots, h_D(x)\}$
   - e.g. polynomials for linear regression
2. Greedy heuristic:
   i. Start with empty set of features $F_0 = \emptyset$ (or simple set, like just $h_0(x) = 1 \Rightarrow y_i = w_0 + \varepsilon_i$)
   ii. Fit model using current feature set F_t to get $\hat{w}^{(t)}$
   iii. Select next best feature $h_{j^*}(x)$
   - eg $h_j(x)$ resulting in lowest training error when learning with $F_t + \{h_j(x)\}$
   iv. Set $F_{t+1} \leftarrow F_t + \{h_{j^*}(x)\}$ // v. Recurse

Greedy procedure



error never increases
+
solutions eventually meet

- #bedrooms
- #bathrooms
- sq. ft living
- sq. ft lot
- floors
- year built
- year renovated
- waterfront

forced to keep in model

When do we stop?

When training error is low enough?
NO!

When test error is low enough?
NO!

Use validation set or cross validation!

Complexity of forward stepwise

How many models were evaluated
- 1st step D models
- 2nd step D-1 models
- 3rd step D-2 models
- ...

How many steps?
- Depends
- At most D steps (to full model)

$O(D^2) << 2^D$ for large D

Other greedy algorithms

Instead of starting from simple model and always growing.

Backward stepwise: Start with full model and iteratively remove features least useful to fit

Combining forward and backward steps:

In forward algorithm, insert steps to remove features no longer as important

Option 3: Regularize

Ridge regression: L_2 regularized regression

Total cost =

measure of fit + λ measure of magnitude of coefficients

$$\underbrace{RSS(w)}_{\text{measure of fit}} + \lambda \underbrace{\|w\|_2^2 = w_0^2 + \ldots + w_D^2}_{\text{measure of magnitude of coefficients}}$$

Encourages small weights but not exactly 0

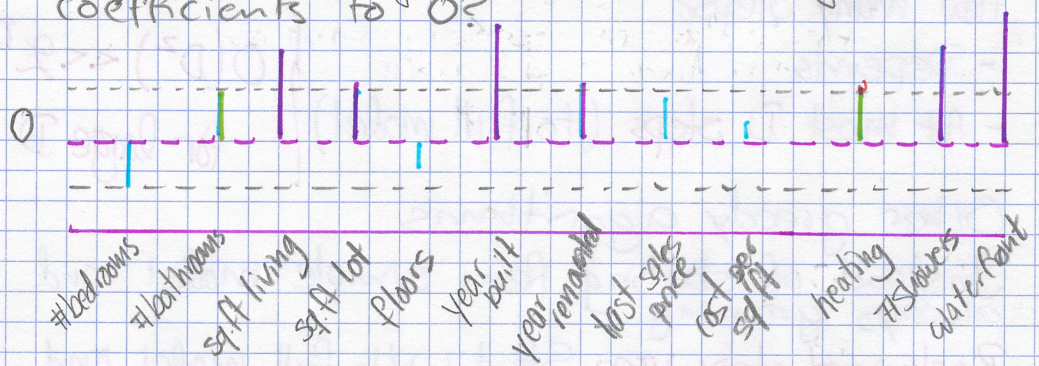
Using regularization for feature selection

Instead of searching over a discrete set of solutions, can we use regularization?

- Start with full model (all possible features)
- "Shrink" some coefficients exactly to 0
  - i.e. knock out certain features
- Non-zero coefficients indicate "selected" features

Thresholding ridge coefficients?

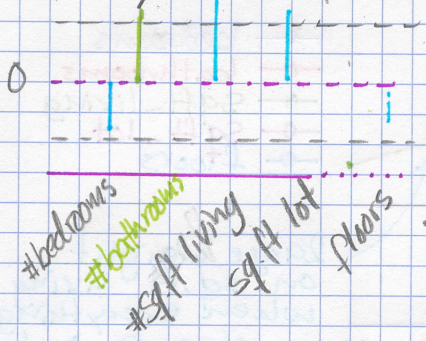
Why don't we just set small ridge coefficients to 0?



Let's look at two related features.. (in green)

Nothing measuring bathrooms was included!

If only one of the features had been included



{would have included bathrooms in selected model

~~#showers~~ waterfront

Can regularization lead directly to sparsity?

Try this cost instead of ridge...

Total cost =

measure of fit + λ measure of magnitude of coefficients

RSS(w)

$$\|w\|_1 = |w_0| + \ldots + |w_D|$$

↑ Leads to sparse solutions!

Lasso regression
(a.k.a. L_1 regularized regression)

Lasso regression:
L_1 regularized regression

Just like ridge regression, solution is governed by a continuous parameter λ

$$RSS(w) + \lambda \|w\|_1$$

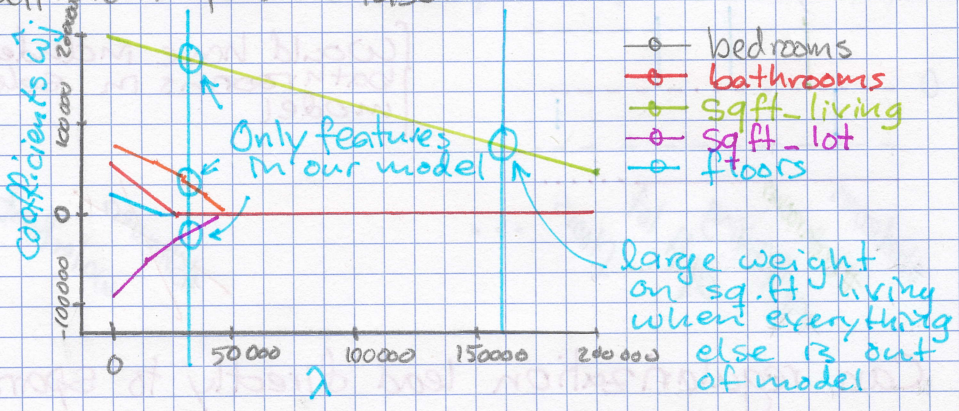
↑ tuning parameter = balance of fit and sparsity

If $\lambda = 0$: $\hat{w}^{lasso} = \hat{w}^{LS}$ (unregularized solution)

If $\lambda = \infty$: $\hat{w}^{lasso} = 0$

If λ in between $0 \le \|\hat{w}^{lasso}\|_1 \le \|\hat{w}^{LS}\|_1$

Coefficient path - lasso

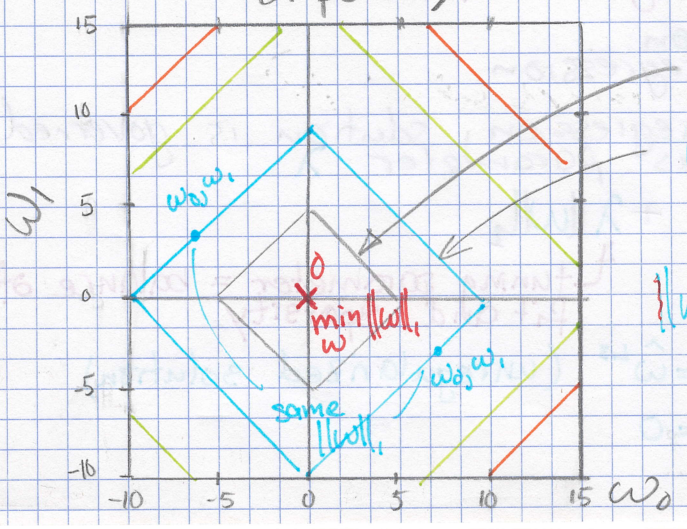


$$RSS(w) + \lambda\|w\|_2^2 = \sum_{i=1}^{N}\left(y_i - w_0 h_0(x_i) - w_1 h_1(x_i)\right)^2 + \lambda(w_0^2 + w_1^2)$$

$RSS(w)$ + $\lambda\|w\|_2^2$

ellipses — weighting — circles

L1 penalty



$\|w\|_1 = const_1$

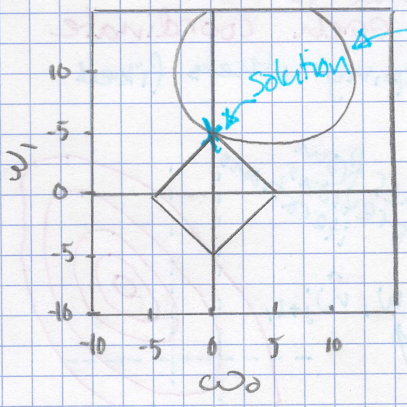
$\|w\|_1 = const_2 > const_1$

$|w_0| + |w_1| = constant$ diamond

$$RSS(w) + \lambda\|w\|_1 = \sum_{i=1}^{N}(y_i - w_0 h_0(x_i) - w_1 h_1(x_i))^2 + \lambda(|w_0| + |w_1|)$$

RSS + $\lambda\|w\|_1$

ellipses — diamonds



Fitting the lasso regression model (for given λ value)

How we optimized past objectives

To solve for $\hat{w}$, previously took gradient of total cost objective and either:

1) Derived closed-form solution
2) Used in gradient descent algorithm

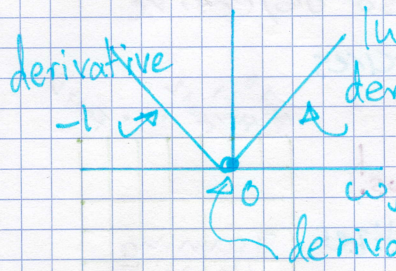
Optimizing the lasso objective

Lasso total cost $RSS(w) + \lambda\|w\|_1$

$\|w\|_1 = \sum_{j=0}^{D}|w_j|$

Issues:

1) What's the derivative of $|w_j|$?



derivative = ? no derivative exists

gradients → subgradients

2) Even if we could compute derivative, no closed-form solution

Aside 1: Coordinate descent

Coordinate descent

Goal: Minimize some function g $\min_w g(w)$

$g(w) = g(w_0, w_1, \ldots, w_D)$

Often, hard to find minimum for all coordinates, but easy for each coordinate when keeping others fixed

Coordinate descent

Initialize $\hat{w} = 0$

while not converged

pick a coordinate j

$\hat{w}_j \leftarrow \min_w g(\hat{w}_0, \ldots, \hat{w}_{j-1}, w, \hat{w}_{j+1}, \ldots, \hat{w}_D)$

values from previous iterations

just min over j^{th} coordinate

Comments on coordinate descent

How do we pick next coordinate?

- At random ("random" or "stochastic" coordinate descent), round robin, ...

No stepsize to choose!

Super useful approach for many problems

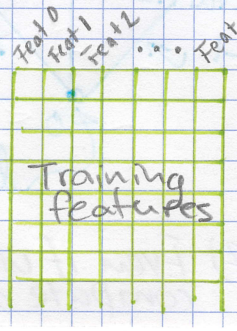
- Converges to optimum in some cases (e.g. strongly convex)
- Converges for lasso objective

Aside 2: Normalizing features

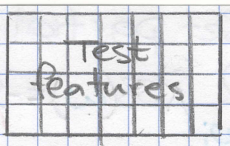
Normalizing features

Scale training columns (not rows!) as

$$\underline{h_j(x_k)} = \frac{h_j(x_k)}{\underbrace{\sqrt{\sum_{i=1}^{N} h_j(x_i)^2}}_{\text{Normalizer } z_j}}$$



Apply same training scale factors to test data



$$h_j(x_k) = \frac{h_j(x_k)}{\underbrace{\sqrt{\sum h_j(x_i)^2}}_{\text{Normalizer } Z_j}}$$

apply to test point

summing over training points

Aside 3: Coordinate descent for unregularized regression (for normalized features)

Optimizing least squares objective one coordinate at a time

$$RSS(w) = \sum_{i=1}^{N} \left(y_i - \sum_{j=0}^{D} w_j \underline{h_j}(x_i)\right)^2$$

unnormalized features.

Fix all coordinates w_j and take partial w.r.t w_j ← all w_k for $k \neq j$

$$\frac{\partial}{\partial w_j} RSS(w) = -2\sum_{i=1}^{N} \underline{h_j}(x_i)\left(y_i - \sum_{j=0}^{D} w_j \underline{h_j}(x_i)\right)$$

1d optimization coordinate by coordinate

$$= -2\sum_{i=1}^{N} \underline{h_j}(x_i)\left(y_i - \sum_{k\neq j} w_k \underline{h_k}(x_i) - w_j \underline{h_j}(x_i)\right)$$

$$= -2\sum_{i=1}^{N} \underline{h_j}(x_i)\underbrace{\left(y_i - \sum_{k\neq j} w_k \underline{h_k}(x_i)\right)}_{\triangleq \rho_j} + 2w_j \boxed{\sum_{i=1}^{N} \underline{h_j}(x_i)^2}$$

by definition of normalized features $= 1$

$$= -2\rho_j + 2w_j$$

Set partial = 0 and solve

$$\frac{\partial}{\partial w_j} RSS(w) = -2\rho_j + 2w_j = 0$$

$$w_j = \rho_j$$

Coordinate descent for least squares regression

Initialize $\hat{w} = 0$ (or smartly...)

while not converged:

for $j = 0, 1, \ldots, D$

compute $\rho_j = \sum_{i=1}^{N} h_j(x_i)(y_i - \hat{y}_i(w_{-j}))$

(residual without feature j; prediction without feature j)

set: $\hat{w}_j = \rho_j$

Coordinate descent for lasso (for normalized features)

Initialize $\hat{w} = 0$ (or smartly)

while not converged:

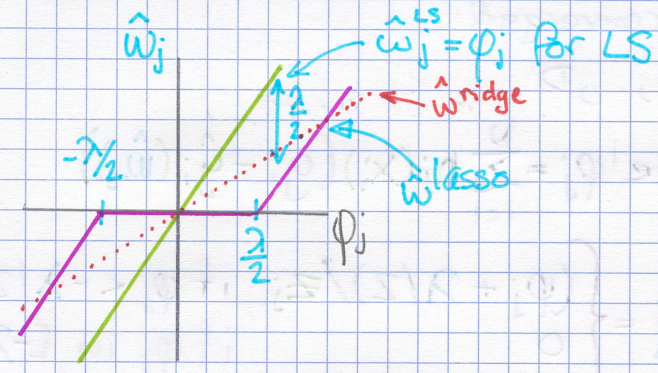
for $j = 0, 1, \ldots, D$

compute $\rho_j = \sum_{i=1}^{N} h_j(x_i)(y_i - \hat{y}_i(\hat{w}_{-j}))$

set $\hat{w}_j = \begin{cases} \rho_j + \lambda/2 & \text{if } \rho_j < -\lambda/2 \\ 0 & \text{if } \rho_j \text{ in } [-\lambda/2, \lambda/2] \\ \rho_j - \lambda/2 & \text{if } \rho_j > \lambda/2 \end{cases}$

Soft thresholding

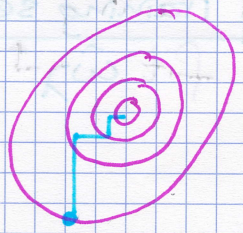
$$\hat{w}_j = \begin{cases} \rho_j + \lambda/2 & \text{if } \rho_j < -\lambda/2 \\ 0 & \text{if } \rho_j \text{ in } [-\lambda/2, \lambda/2] \\ \rho_j - \lambda/2 & \text{if } \rho_j > \lambda/2 \end{cases}$$



Convergence criteria When to stop?

For convex problems, will start to take smaller and smaller steps

Measure size of steps taken in a full loop over all features
- stop when max step $< \epsilon$



Other lasso solvers

Classically: Least angle regression (LARS) [Efron et al. '04]

Then: Coordinate descent algorithm [Fu '98, Friedman, Hastie, & Tibshirani '08]

Now:
- Parallel CD (e.g. Shotgun, [Bradley et al. '11])
- Other parallel learning approaches for linear models
  - Parallel stochastic gradient descent (SGD) (e.g. Hogwild! [Niu et al. '11])
  - Parallel independent solutions then averaging [Zhang et al. '12]
- Alternating directions method of multipliers (ADMM) [Boyd et al. '11]

Coordinate descent for lasso (for unnormalized features)

Precompute $z_j = \sum_{i=1}^{N} h_j(x_i)^2$

Initialize $\hat{w} = 0$

while not converged:

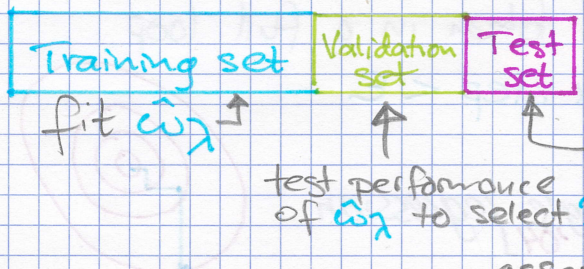
for $j = 0, 1, \ldots, D$

compute $\rho_j = \sum_{i=1}^{N} h_j(x_i)(y_i - \hat{y}_i(\hat{w}_{-j}))$

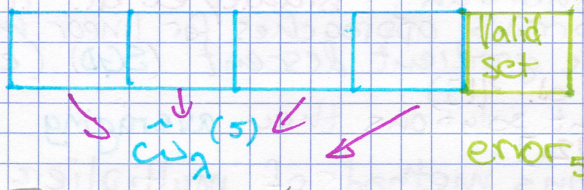
set $\hat{w}_j = \begin{cases} (\rho_j + \lambda/2)/z_j & \text{if } \rho_j < -\lambda/2 \\ 0 & \text{if } \rho_j \text{ in } [-\lambda/2, \lambda/2] \\ (\rho_j - \lambda/2)/z_j & \text{if } \rho_j > \lambda/2 \end{cases}$

How to choose λ

If sufficient amount of data



k-fold cross validation



For $k = 1, \ldots, k$

1. Estimate $\hat{w}_\lambda^{(k)}$ on the training blocks
2. Compute error on validation block $error_k(\lambda)$

Compute average error $CV(\lambda) = \frac{1}{K}\sum_{k=1}^{K} \text{error}_k(\lambda)$

Choosing λ via cross validation

Cross validation is choosing the λ that provides best predictive accuracy

Tends to favor less sparse solutions, and thus smaller λ than optimal choice for feature selection

c.f. "Machine Learning: A Probabilistic Perspective", Murphy, 2012 for further discussion

Practical concerns with lasso

Debiasing lasso

Lasso shrinks coefficients relative to LS solution
→ more bias, less variance

Can reduce bias as follows:
1. Run lasso to select features
2. Run least squares regression with only selected features

"Relevant" features no longer shrunk relative to LS fit of some reduced model

Issues with standard lasso objective

1. With group of high correlated features, lasso tends to select amongst them arbitrarily
   - often prefer to select all together
2. Often, empirically ridge has better predictive performance than lasso, but lasso leads to sparser solution

Elastic net aims to address these issues
- hybrid between lasso and ridge regression
- uses L_1 and L_2 penalties

See Zou & Hastie '05 for further discussion

Impact of feature selection and lasso

Lasso has changed machine learning, statistics & electrical engineering

But, for feature selection in general, be careful about interpreting selected features
- selection only considers features included
- sensitive to correlations between features
- result depends on algorithm used
- there are theoretical guarantees for lasso under certain conditions