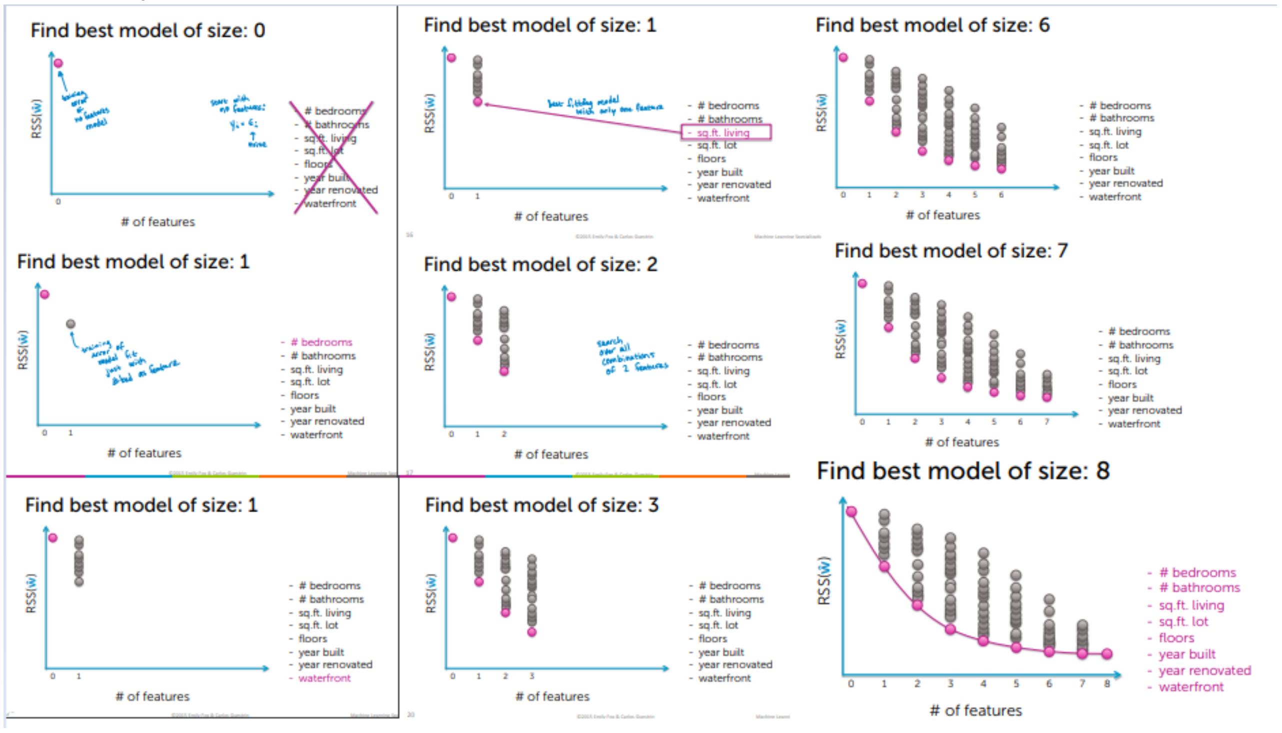
Lasso Regression : Regularization for feature selection

Feature selection task

- Why should you perform feature selection?
- If size(w) - large - 100B - each prediction is expensive (computationally);
- If w-hat - estimated parameters are **sparse** - with many 0's, computation only depends on # of non-zeros.
- Interpretability - Which features are relevant for predictions.

Option 1 : All subsets

- Search over every possible combination of features that we might want to include in the model and look at the performance of each of the model.

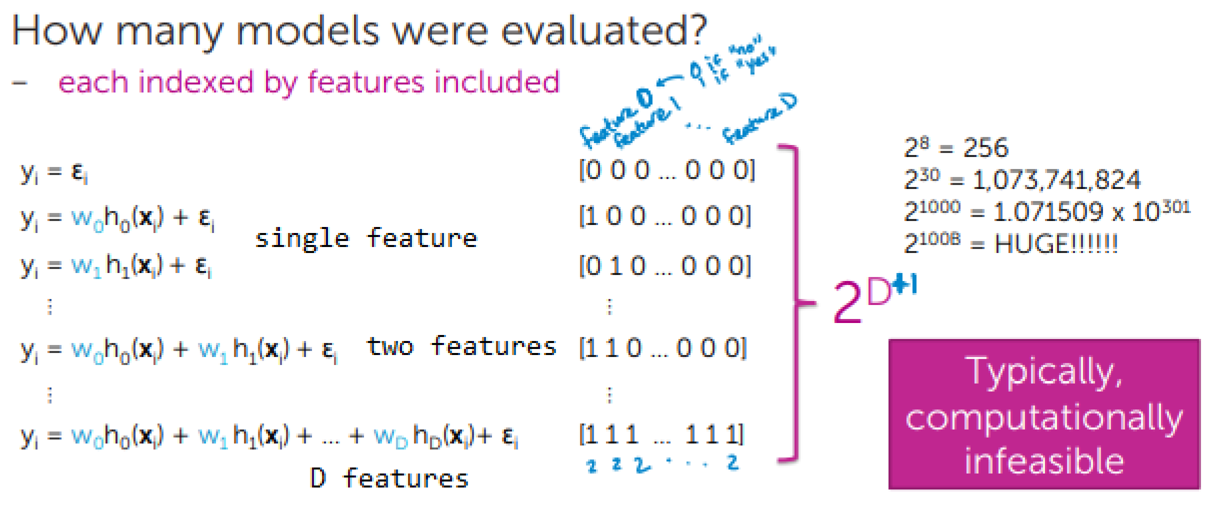


Choosing model complexity

- The training error reduces with an increase in the moddel complexity.
- Choice 1 : Assess on validation set. (enough data)
- Choice 2 : Cross validation. (insufficient data)
- Choice 3 : Other metrics for penalizing model complexity like BIC, ...

Complexity of "all subsets"

- Evaluating all the subset models will be computationlly expensive.
- Each indexed by the features included.



How many models were evaluated?

– each indexed by features included

Feature 0
feature 1
...
feature D
0 if "no"
1 if "yes"

$y_i = \varepsilon_i$ [0 0 0 … 0 0 0]

$y_i = w_0 h_0(\mathbf{x}_i) + \varepsilon_i$ [1 0 0 … 0 0 0]

single feature

$y_i = w_1 h_1(\mathbf{x}_i) + \varepsilon_i$ [0 1 0 … 0 0 0]

⋮

$y_i = w_0 h_0(\mathbf{x}_i) + w_1 h_1(\mathbf{x}_i) + \varepsilon_i$ two features [1 1 0 … 0 0 0]

⋮

$y_i = w_0 h_0(\mathbf{x}_i) + w_1 h_1(\mathbf{x}_i) + \ldots + w_D h_D(\mathbf{x}_i) + \varepsilon_i$ [1 1 1 … 1 1 1]

2 2 2 · · · 2

D features

2^{D+1}

$2^8 = 256$

$2^{30} = 1{,}073{,}741{,}824$

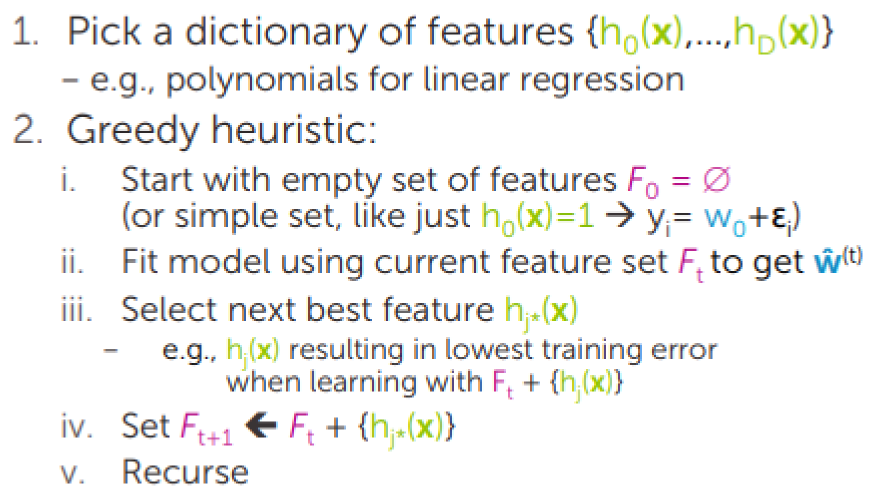
$2^{1000} = 1.071509 \times 10^{301}$

2^{100B} = HUGE!!!!!!

Typically, computationally infeasible

Option 2 : Greedy Algorithms

Forward step-wise algorithm:

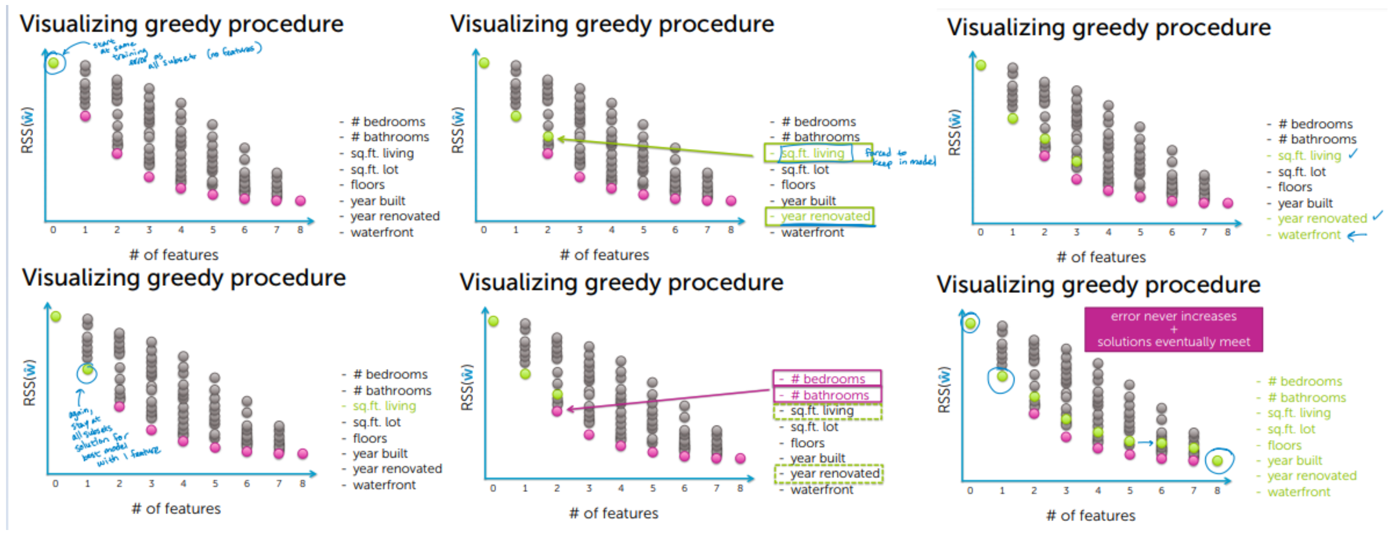


1. Pick a dictionary of features $\{h_0(\mathbf{x}),\ldots,h_D(\mathbf{x})\}$
   – e.g., polynomials for linear regression
2. Greedy heuristic:
   i. Start with empty set of features $F_0 = \varnothing$ (or simple set, like just $h_0(\mathbf{x})=1 \rightarrow y_i = w_0 + \varepsilon_i$)
   ii. Fit model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

- Here unlike the **all subset** model, the next feature selected will be dependent on the selected features.
- For example:

```
- Model with 0 features -> the training error is plotted.
- Model with 1 feature -> the feature with least training error is selecte
d. (sqft_living)
- Model with 2 features -> the feature associated with previous selected fe
atures - 'sqft_living that' is 'year renovated' is choosen. But the two fea
tures with least training error are - '# bedrooms' & '# bathrooms'.
```

- Eventually the greedy algorithm selectes all the features - reaches the endpoint as in all subsets.



When to stop the feature selection process:

- When training error is low enough ? NO (Training error decreases with increase in model complexity);
- When test error is low enough? NO (Test set must never be touched);
- Use **validation set** or **cross validation**.

Complexity of forward stepwise

- How many models were evaluated?
- 1st step, D models
- 2nd step, D-1 models (add 1 feature out of D-1 possible)
- 3rd step, D-2 models (add 2 features out of D-2 possible)
- ...
- How many steps?
- Depends
- At most D steps (to full model)
- **Complexity** of **forward stepwise algorithm** O(D^2);
- **Complexity** of **all subset** O(2^D);
- For large D - **O(D^2) << O(2^D)**.

Other greedy algorithms

1. **Backward stepwise** : Start with a full model and iteratively remove features least useful to the fit.
2. **Combining forward and backward steps** : In forward algorithm, insert steps to remove features no longer important.
3. many more...

Option 3 : Regularize

Ridge regression : L2 regularized regression

- Total cost = measure of fit (RSS(w)) + λ * Measure of the magnitude of the coefficients
- Total cost = RSS(w) + ||w||(2)^2
- ||w||(2)^2 - L2 norm - sum of thw feature weights squared

```
= w(0)^2 + ... + w(D)^2.
```

- It encourages **small wieghts** but not exactly 0. The magnitude of the coefficients decreases or reaches 0 as λ reaches infinity.

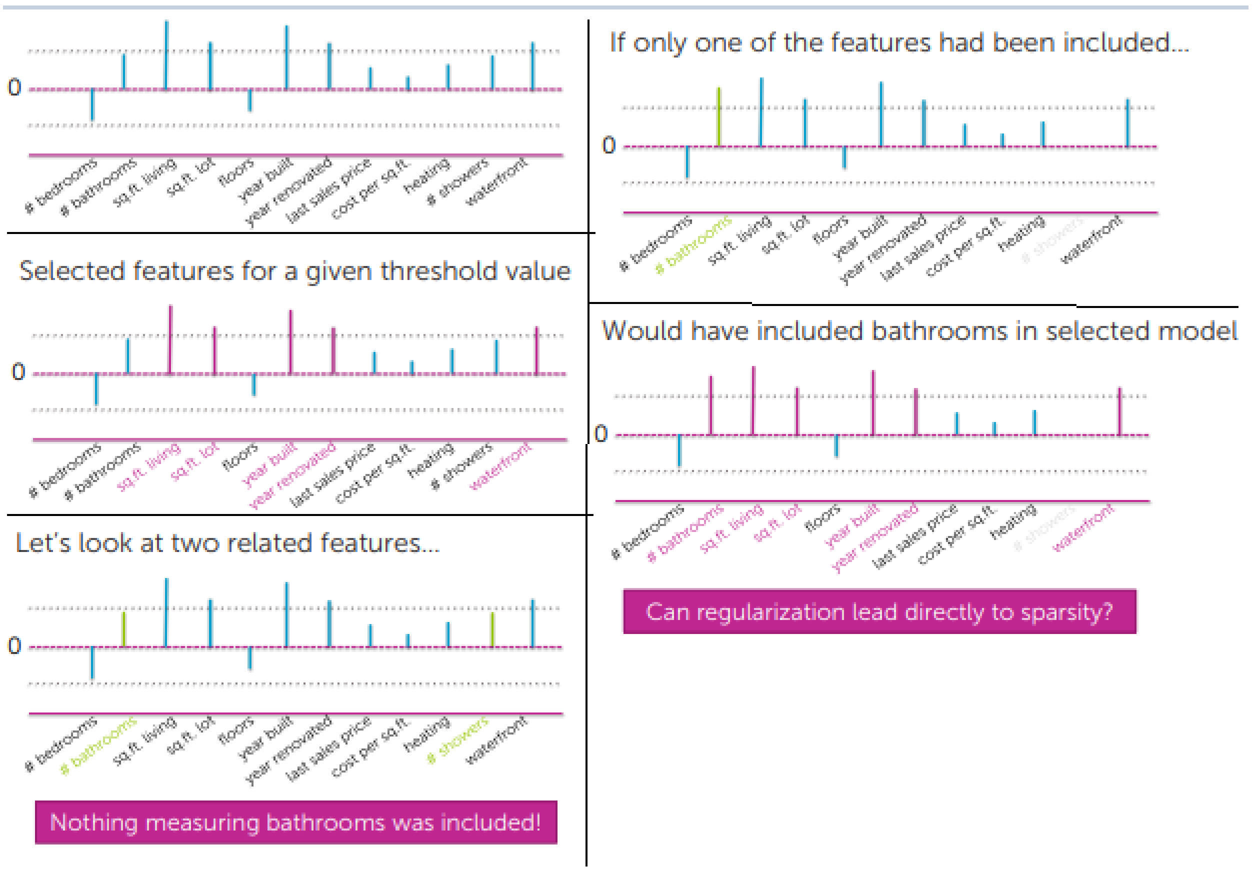
- If the coefficient weights become 0, then they are not needed in the computation and also not needed for the model prediction.

Using regularization for feature selection

- Instead of searching over a discrete set of solutions, it can be regularized.
- Start with full model (all possible features).
- "Shrink" some coefficients exactly to 0. (i.e., knock out certain values).
- Non-zero coefficients indicate "selected" features.

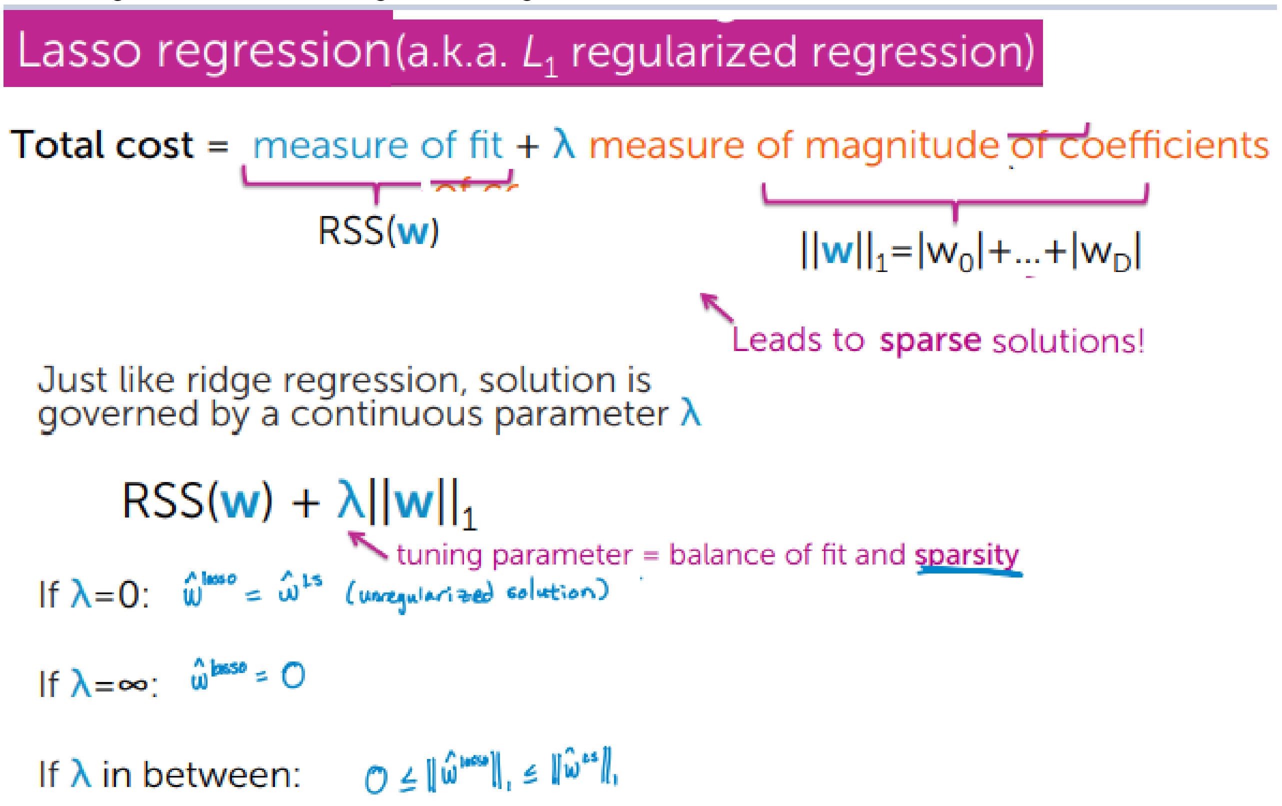
Thresholding ridge coefficients

- If the magnitude of the coefficients is close to 0 then they are discarded, only features with significant magnitude are taken into consideration.
- In case of closely related features - their magnitudes might be close to each other. Their weights are distributed and might be close to 0 and excluded from the model.
- Example : # bathrooms & # showers -> closely related features - their magnitude is similar and close to 0 hence excluded from model. But here if the # shower is removed then the weight of # shower is added to # bathrooms and hence the magnitude of this feature **# showers** increases This happens because it is a linear model -> Σ(w(bath) *#bath + w(shower)* #shower) -> transformed -> Σ((w(bath) + w(shower)) * #bath;
- Issue : In case of strongly related features / correlated features, then Ridge regression - prefers to place smaller weights on all features rather than one large weight on one of the features. Since the cost of the feature in Ridge regression - **Size of the feature squared** -> L2 penalty increases.
- Due to which certain significant features might gets lost in Ridge regularization due to correlated features.
- **Thresholding Ridge regression** is **not** a solution to **feature selection**.



The Lasso objective and its coefficient path

- In Ridge regression-> weight of the coefficients is distributed amongst the correlated features.
- Total Cost (Ridge regression) = RSS(w) + λ ||w||(2)^2;
- Ridge regression - a.k.a L2 regularized regression.
- In Lasso regression -> weight of the coefficient is associated with the feature.
- Total Cost (Lasso regression) = RSS(w) + λ ||w||(1)
- ||w||(1) = |w0|+|w1|+...|wD-1|+|wD|;(Leads to sparse solutions!)
- Lasso regression - a.k.a L1 regularized regression.



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

$$\text{Total cost} = \underbrace{\text{measure of fit}}_{RSS(\mathbf{w})} + \lambda \underbrace{\text{measure of magnitude of coefficients}}_{||\mathbf{w}||_1=|w_0|+\ldots+|w_D|}$$

Leads to sparse solutions!

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

$$RSS(\mathbf{w}) + \lambda||\mathbf{w}||_1$$

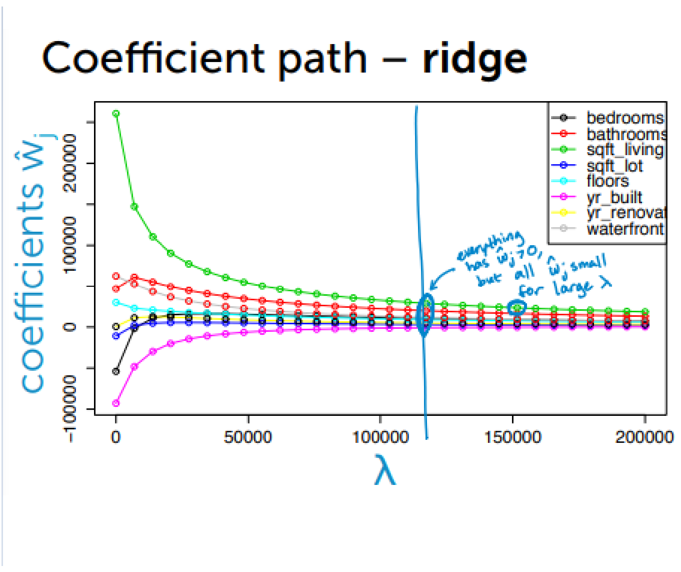
tuning parameter = balance of fit and sparsity

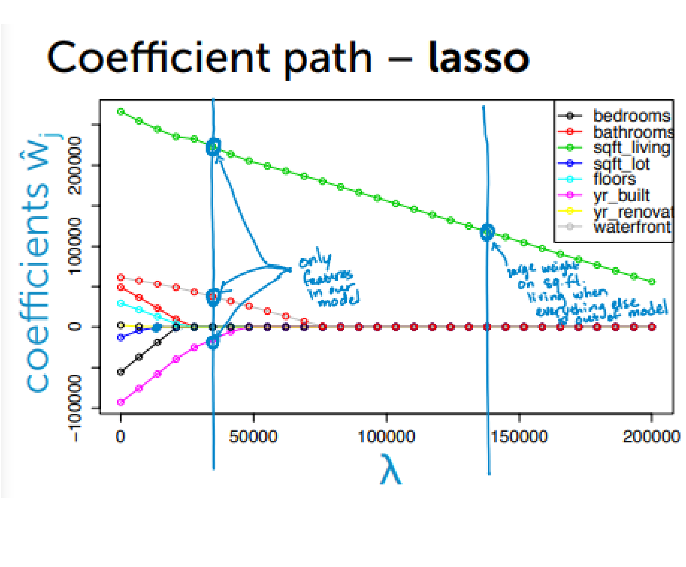
If $\lambda=0$: $\hat{w}^{lasso} = \hat{w}^{LS}$ (unregularized solution)

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

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

Coefficient-path ridge v/s lasso





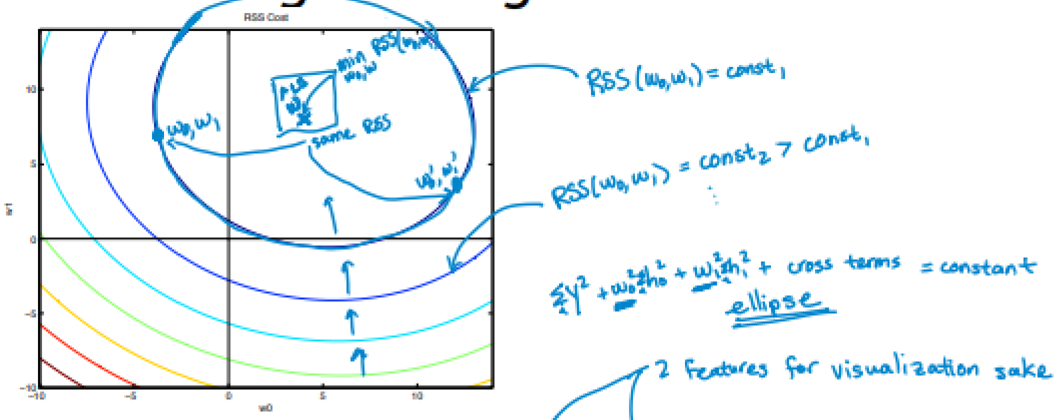
Geometric intuition for sparsity of lasso solutions

Visualizing the ridge cost in 2D

1. For the RSS term

- Consider 2 features h0(xi) and h1(xi);
- Contour plot for the RSS (Residual Sum of Squares) value.
- The equation of the RSS term -> forms an **ellipse**.
- The contour plot consists of ellipses form each (w0,w1) pair.
- Along the contour plot, all the RSS values are the same.
- The inner most ellipse, center point has the least RSS. So the RSS convergence from outside-in.

Visualizing the ridge cost in 2D

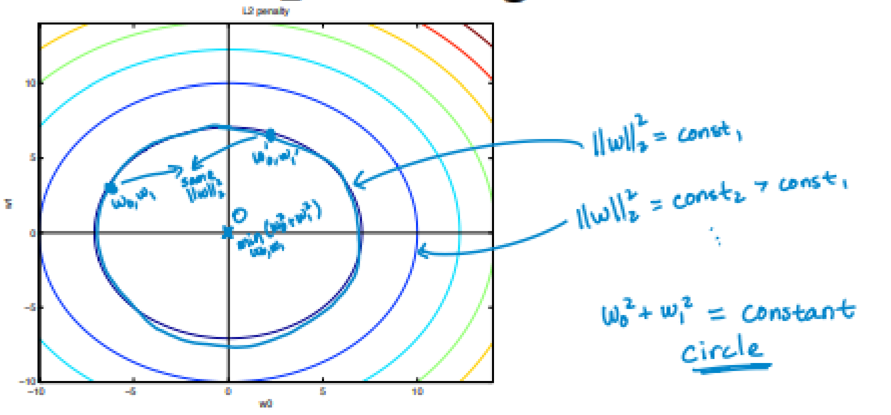


$$RSS(\mathbf{w}) + \lambda||\mathbf{w}||_2^2 = \sum_{i=1}^{N}(y_i - w_0 h_0(\mathbf{x}_i) - w_1 h_1(\mathbf{x}_i))^2 + \lambda\,(w_0^2 + w_1^2)$$

__2. For the L2 penalty__

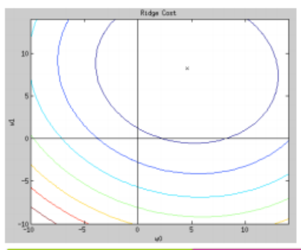
- **circle**.Each circle in the contour plot is a constant ||w||2^2.
- Around the circle, all the points have the same ||w||2^2.
- Minimum -> outside - in, the central point, where solution=0, and λ = ∞(infinity);

Visualizing the ridge cost in 2D



$$RSS(\mathbf{w}) + \lambda||\mathbf{w}||_2^2 = \sum_{i=1}^{N}(y_i - w_0 h_0(\mathbf{x}_i) - w_1 h_1(\mathbf{x}_i))^2 + \lambda\,(w_0^2 + w_1^2)$$

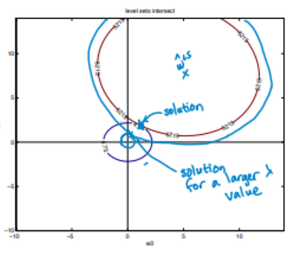
- The ridge cost = RSS(w) + λ||w||2^2 -> ellipse + circle;



Add contour plots together
RSS(w) + λ ||w||$_2^2$
ellipses ↑ circles
weighting

Movie: function of increasing λ

X mark optimal solution for a specific λ

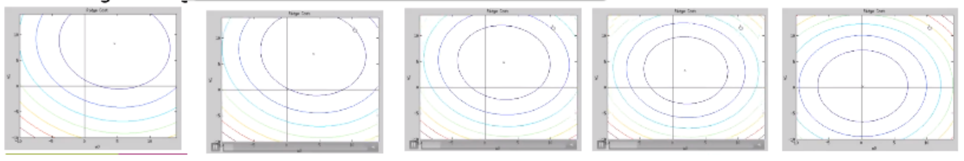


For a specific λ value, some balance between RSS and ||w||$_2^2$

```
Solution for λ -> has RSS-> 5215,
  ||w||2^2 - 12 penalty - 4.75

Solution for larger λ -> RSS -> 6000,
  ||w||2^2 - 12 penalty - 3.00
```

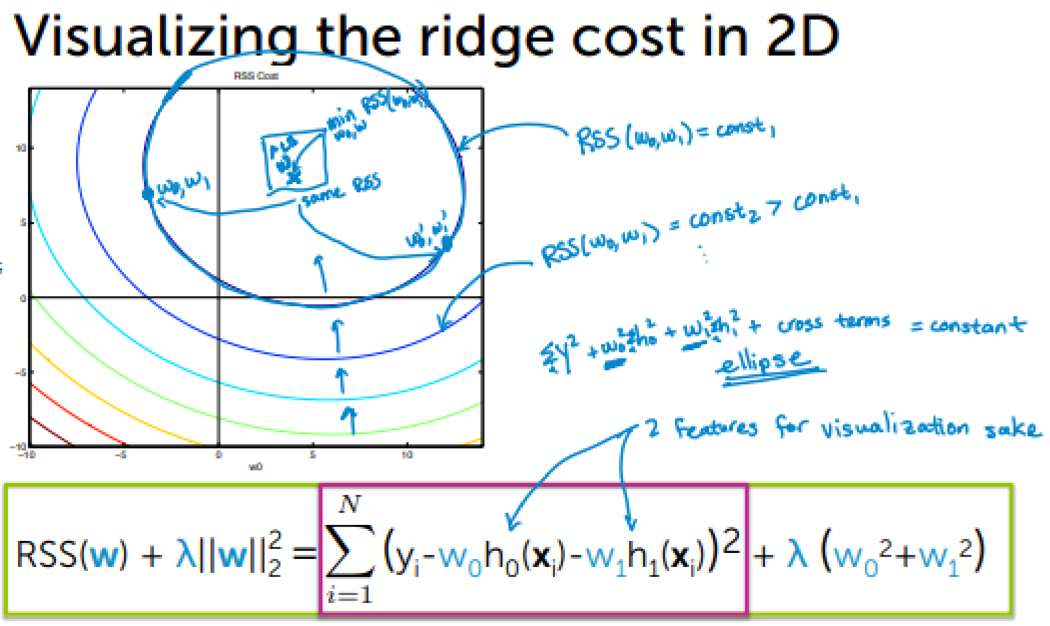
$$\mathrm{RSS}(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2 = \sum_{i=1}^{N}(y_i - w_0h_0(\mathbf{x}_i) - w_1h_1(\mathbf{x}_i))^2 + \lambda\,(w_0^2 + w_1^2)$$



```
With increasing λ, the magnitude of coefficients in decreasing as shown in the contour plot.
```

Visualizing the lasso cost in 2D

Term 1 : RSS = contour plot - ellipses same as those for Ridge.



Visualizing the ridge cost in 2D

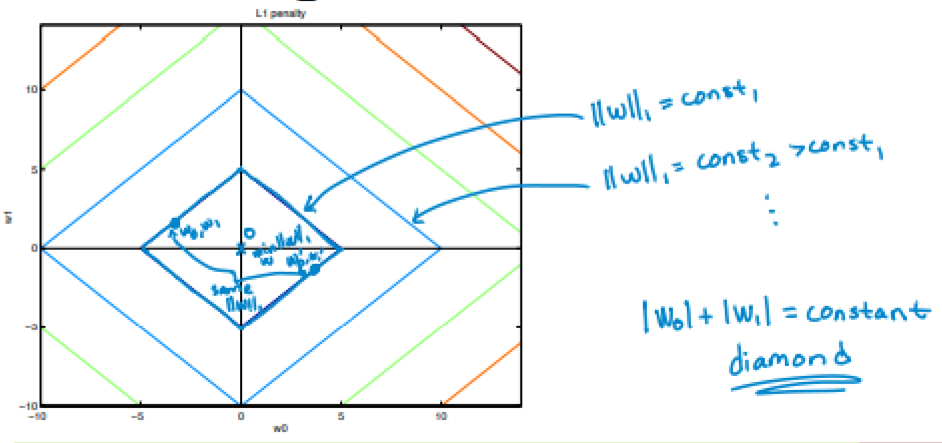
RSS Cost
min RSS(w0,w1)
$\hat{w}_0, \hat{w}_1$
w_0, w_1
same RSS
w_0', w_1'
RSS (w0,w1) = const1
RSS(w0,w1) = const2 > const1
$\sum y_i^2 + w_0^2 h_0^2 + w_1^2 h_1^2$ + cross terms = constant
ellipse
2 features for visualization sake
w0
w1

$$\mathrm{RSS}(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2 = \sum_{i=1}^{N}(y_i - w_0h_0(\mathbf{x}_i) - w_1h_1(\mathbf{x}_i))^2 + \lambda\,(w_0^2 + w_1^2)$$

Term 2 : L1 penalty = ||w|| -> absolute value

- **Diamond**.
- Every point (w0,w1)-pair along the Diamond -> has the same L1 norm value.
- To reduce the cost, make the magnitude as small as possible. central point.

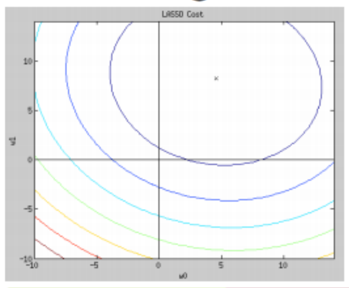
Visualizing the lasso cost in 2D

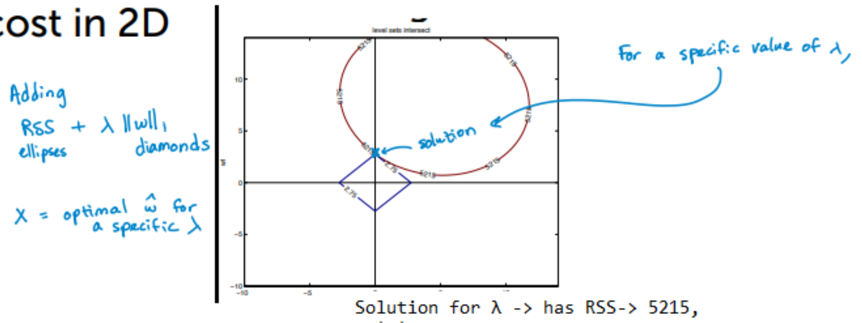


$$RSS(\mathbf{w}) + \lambda\|\mathbf{w}\|_1 = \sum_{i=1}^{N}(y_i - w_0h_0(\mathbf{x}_i) - w_1h_1(\mathbf{x}_i))^2 + \lambda\,(|w_0|+|w_1|)$$

- The lasso cost = RSS(w) + λ|w|

Visualizing the lasso cost in 2D

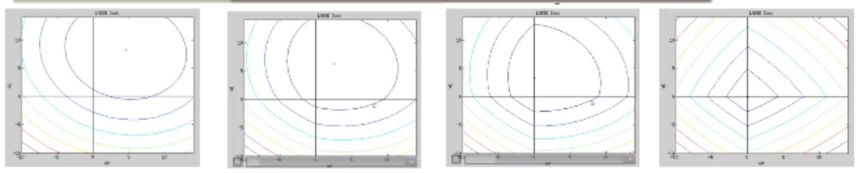




Solution for λ -> has RSS-> 5215,
 |w| - 11 penalty - 2.75

Solution for larger λ -> RSS -> 6000,
 |w| - 11 penalty - 1.25

$$RSS(\mathbf{w}) + \lambda\|\mathbf{w}\|_1 = \sum_{i=1}^{N}(y_i - w_0h_0(\mathbf{x}_i) - w_1h_1(\mathbf{x}_i))^2 + \lambda\,(|w_0|+|w_1|)$$



With increasing λ, the magnitude of coefficients in decreasing as shown in the contour plot.The w0 term decreases, coefficients decreases hits the y-axis as x-axis is 0. Thereafter the w1 term decreases and reaches the minimum 0.

- The solution point - we get sparse results.

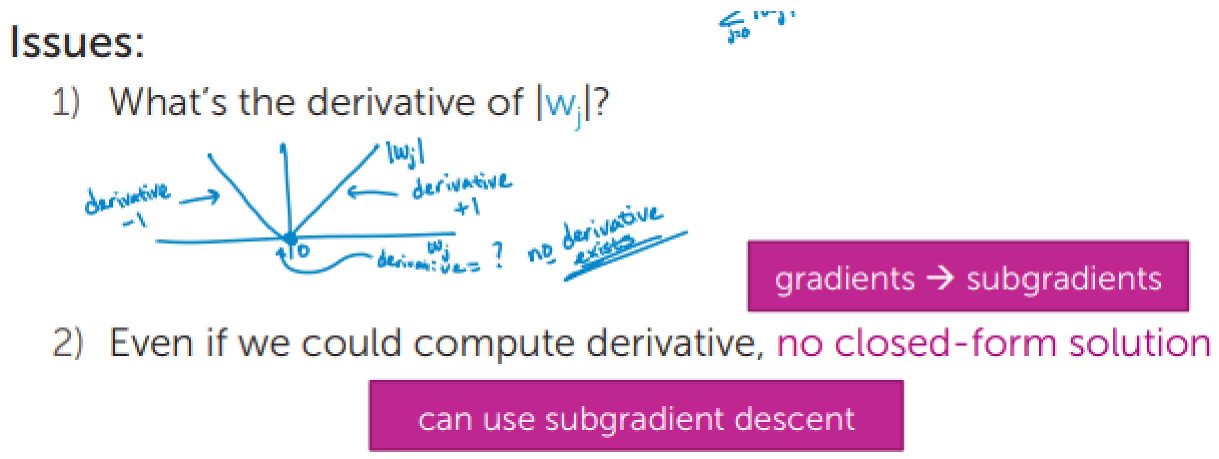
Fitting the lasso regression model (for a given λ value) -> ML Algorithm

Previous Optimization techniques

- To solve for w-hat(estimated parameters), previously took gradient of the total cost objective and either: 1) Derived closed-form solution. 2) Used in gradient descent algorithm.

Optimization technique for lasso objective

- Lasso total cost : RSS(w) + λ ||w||1;
- The derivative exists for the 1st term -> RSS(w);
- The derivative of the 2nd term -||w||1 - absolute value
  - No derivative exists at 0.
  - No closed-form solution.
  - Gradient Descent solution exists but in terms of sub-gradients.



Issues:

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

$\sum_{j=0} |w_j|$
derivative -1
$|w_j|$
derivative +1
w_j
+1/0
derivative = ?
no derivative exists

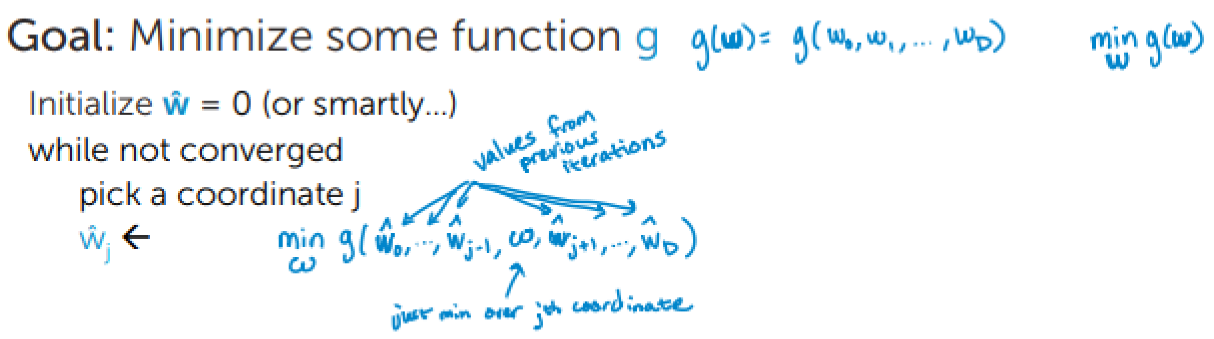
gradients → subgradients

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

can use subgradient descent

Coordinate Descent

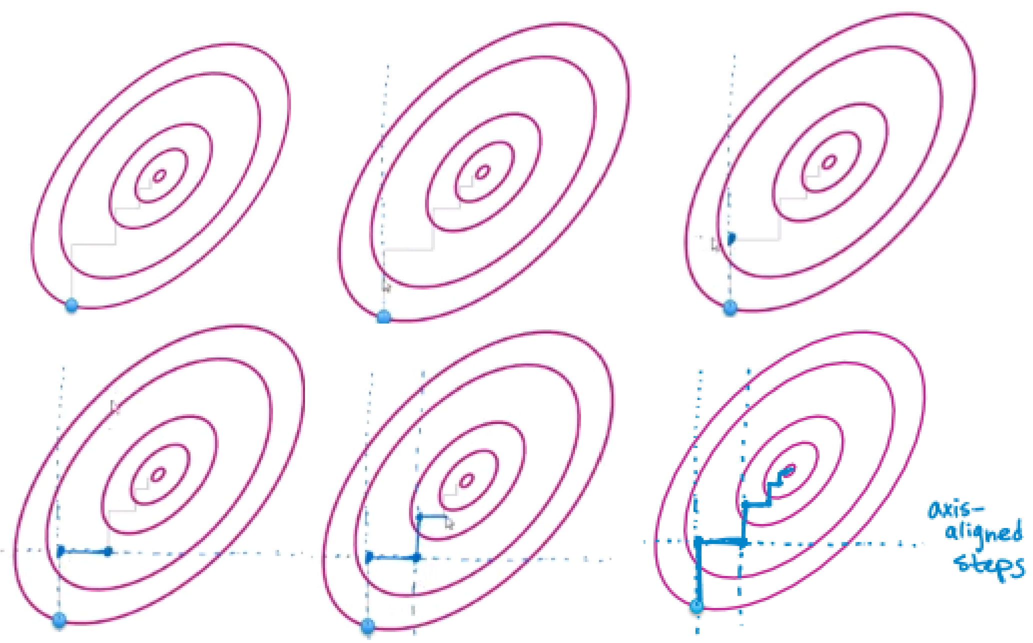
- Alternative optimization technique. (Instead of gradient-descent / sub-gradient descent);
- Goal is to minimize some function g.
- Often it is hard to minimize for all coordinates/dimensions, but easy for each coordinate **(when keeping other coordinates fixed)**.
- Step 1 : Initialize w-hat(estimated parameters) = 0(or smartly..)
- Step 2 : While not converged, pick a co-ordinate j.
  - w-hat(j) = min g(w0-hat,..., w(j-1)-hat, wj,w(j+1)-hat,..., w(D)-hat)
  - except wj, the values are taken from the previous iteration, inorder to find the min over the jth coordinate - wj;



Goal: Minimize some function g $g(w) = g(w_0, w_1, \dots, w_D)$ $\min_{w} g(w)$

```
Initialize $\hat{w}$ = 0 (or smartly...)
while not converged
    pick a coordinate j
    $\hat{w}_j \leftarrow \min_{\omega} g(\hat{w}_0, \dots, \hat{w}_{j-1}, \omega, \hat{w}_{j+1}, \dots, \hat{w}_D)$
```

values from previous iterations
just min over jth coordinate

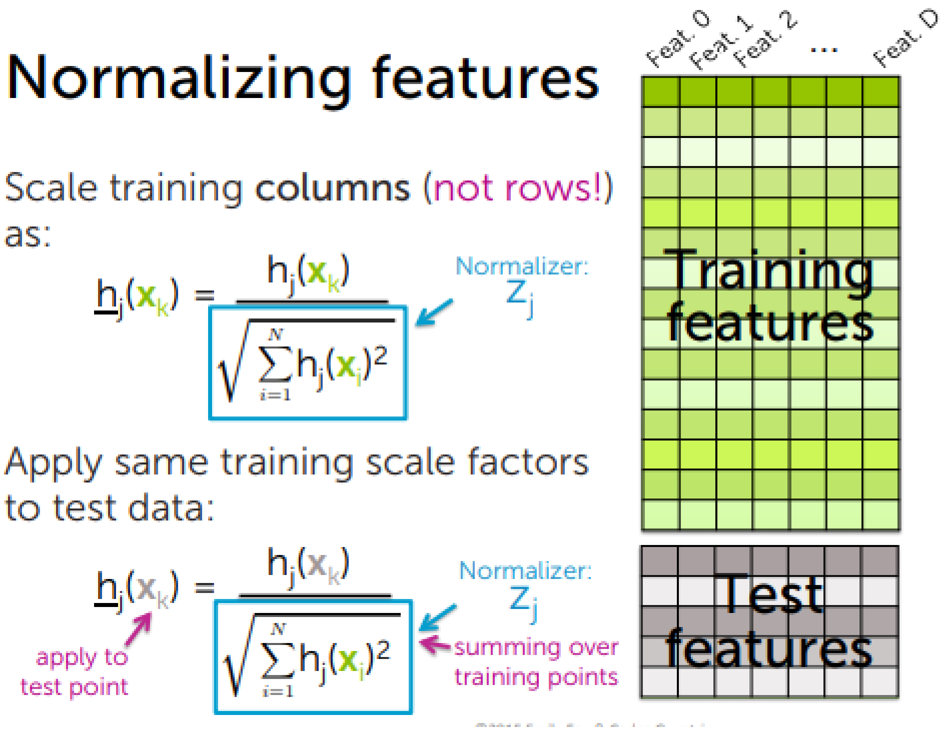


How do we pick next coordinate to optimize?

- At random("random" or "stochastic" coordinate descent), round robin(clycle through the features 0,1,2,..,D,0,1,2,3,...,D,0,1..), etc.
- **No stepsize to choose!**
- Super useful approach for many problems.
- Converges to optimum in some cases - e.g strongly convex.
- Converges for lasso objective.

Normalizing features

- This operation is performed on each column - feature -> sqft_living, #bathroom, #bedroom, #year_built, etc. Not rows -> across each observation.
- This operation puts each feature in the same numeric range.
- **Important Transformation**.
- It is vital to perform this operation on the whole dataset -> training and test set.
- for each column, take the column / (sqrt of all the features in the set);



Normalizing features

Scale training columns (not rows!) as:

$$\underline{h}_j(\mathbf{x}_k) = \frac{h_j(\mathbf{x}_k)}{\boxed{\sqrt{\sum_{i=1}^{N} h_j(\mathbf{x}_i)^2}}}$$

Normalizer: z_j

Apply same training scale factors to test data:

$$\underline{h}_j(\mathbf{x}_k) = \frac{h_j(\mathbf{x}_k)}{\boxed{\sqrt{\sum_{i=1}^{N} h_j(\mathbf{x}_i)^2}}}$$

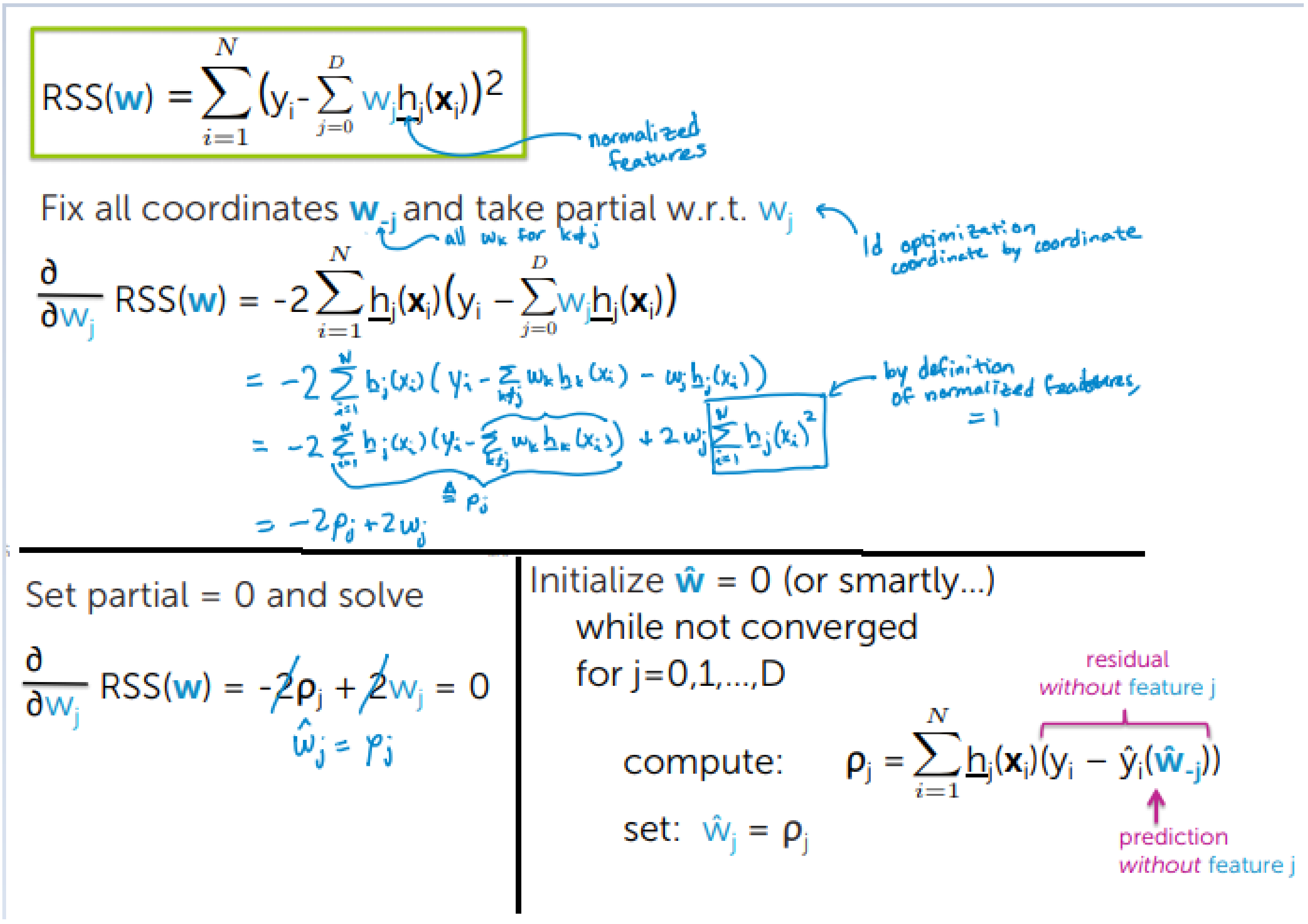
apply to test point

Normalizer: z_j

summing over training points

Coordinate descent for unregularized regression (for normalized features)

- The features hj(xi) -> normalized features.
- Here al the coordinates w(-j)/ w(k-j) except (j) are fixed.
- Take the partial w.r.t w(j) term.



$$\boxed{\mathrm{RSS}(\mathbf{w}) = \sum_{i=1}^{N}\Big(y_i - \sum_{j=0}^{D} w_j \underline{h}_j(\mathbf{x}_i)\Big)^2}$$

normalized features

Fix all coordinates $\mathbf{w}_{-j}$ and take partial w.r.t. w_j

all w_k for $k \neq j$

1d optimization coordinate by coordinate

$$\frac{\partial}{\partial w_j}\mathrm{RSS}(\mathbf{w}) = -2\sum_{i=1}^{N}\underline{h}_j(\mathbf{x}_i)\Big(y_i - \sum_{j=0}^{D} w_j \underline{h}_j(\mathbf{x}_i)\Big)$$

$$= -2\sum_{i=1}^{N} \underline{h}_j(x_i)\Big(y_i - \sum_{k\neq j} w_k \underline{h}_k(x_i) - w_j \underline{h}_j(x_i)\Big)$$

$$= \underbrace{-2\sum_{i=1}^{N} \underline{h}_j(x_i)\Big(y_i - \sum_{k\neq j} w_k \underline{h}_k(x_i)\Big)}_{\triangleq\ \rho_j} + 2w_j \boxed{\sum_{i=1}^{N} \underline{h}_j(x_i)^2}$$

by definition of normalized features, $= 1$

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

Set partial = 0 and solve

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

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

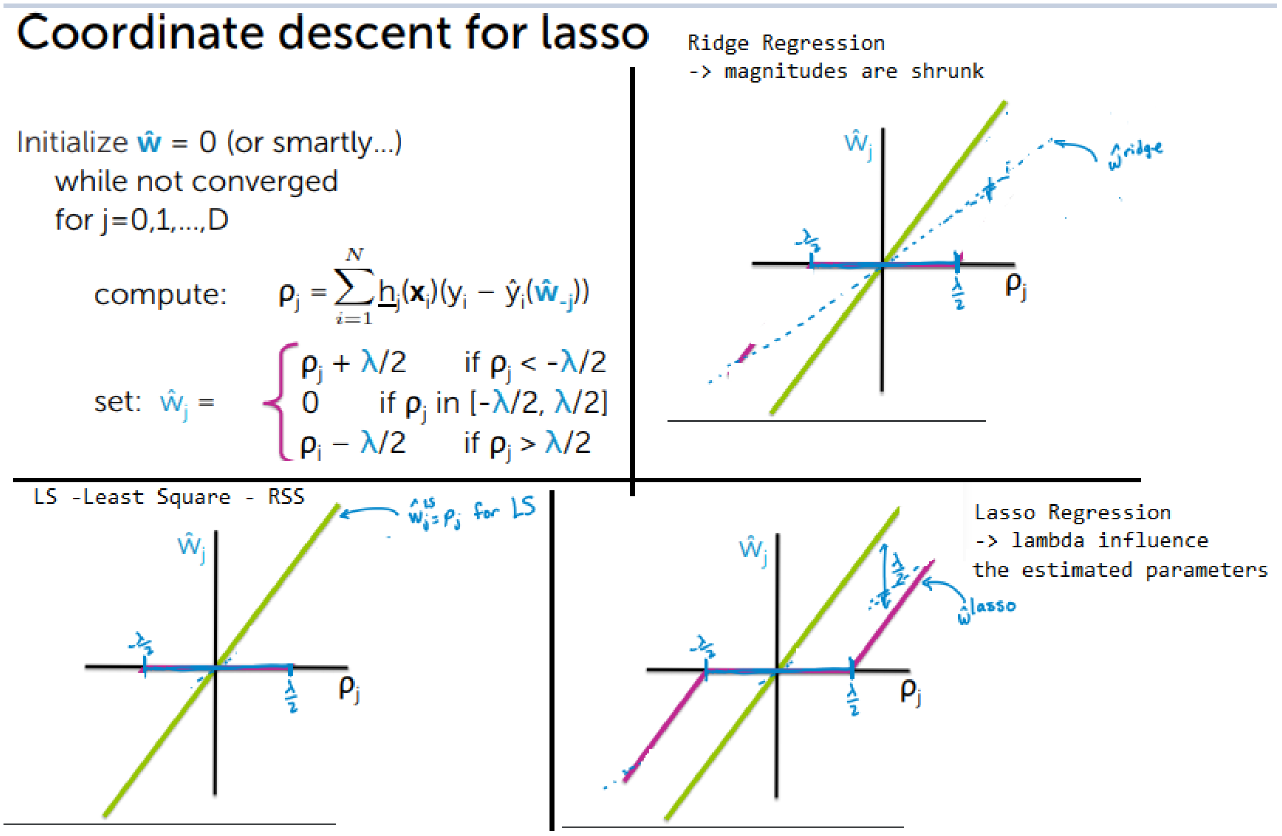
```
Initialize ŵ = 0 (or smartly...)
    while not converged
    for j=0,1,...,D
        compute: $\rho_j = \sum_{i=1}^{N} \underline{h}_j(\mathbf{x}_i)(y_i - \hat{y}_i(\hat{\mathbf{w}}_{-j}))$
        set: $\hat{w}_j = \rho_j$
```

residual *without* feature j

prediction *without* feature j

Coordinate descent for lasso (for normalized features)

- The estimated coefficients w-hat is derived from the predicted value **ρj** as shown above.
- But in lasso - the tuning parameter **λ** influence the **ρj** term.
- If **ρj** is **small** then **ρj** + **λ/2**.
- If **ρj** is **-λ/2 | λ/2** the **0**.
- If **ρj** is **large** then **ρj** - **λ/2**.



Coordinate descent for lasso

```
Initialize $\hat{\mathbf{w}} = 0$ (or smartly...)
    while not converged
        for j=0,1,...,D
            compute: $\rho_j = \sum_{i=1}^{N} h_j(\mathbf{x}_i)(y_i - \hat{y}_i(\hat{\mathbf{w}}_{-j}))$
            set: $\hat{w}_j = \begin{cases} \rho_j + \lambda/2 & \text{if } \rho_j < -\lambda/2 \\ 0 & \text{if } \rho_j \text{ in } [-\lambda/2, \lambda/2] \\ \rho_j - \lambda/2 & \text{if } \rho_j > \lambda/2 \end{cases}$
```

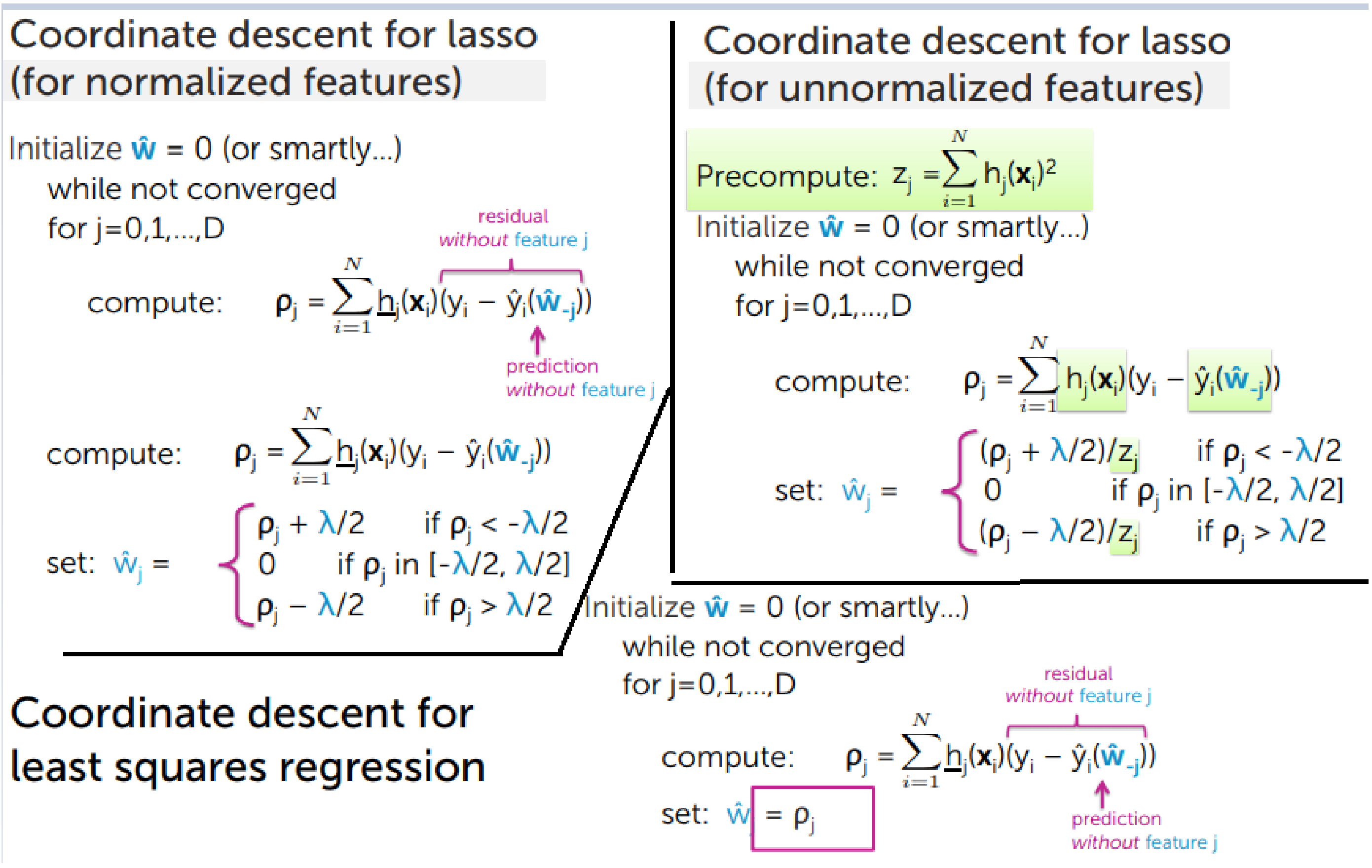
Convergence criteria:

- In gradient descent - we were looking at the magnitude of the gradient vector and stopped when the magnitude of the gradient vector was below some tolerance value - epsilon - **ε**.
- In coordinate descent - when to stop?
- For convex problems, the coordinate descent algorithm takes smaller and smaller steps at it approaches te optimum, or the minimum.
- Note : Measure the size of the steps taken in a **full loop over all features**. Stop when max step < **ε**.

Other lasso solvers

1. Classically - Least angle regression (LARS) - 2004
2. Coordinate Descent Algorithm - 2008
3. Now:
    - Parallel CD (2011)
    - Other parallel learning approaches for linear models
        - Parallel stochastic gradient descent (SGD) (2011).
        - Parallel independent solutions then averaging (2012).
    - Alternating directions methods of multipliers (ADMM) (2011).

Coordinate descent for normalized v/s unnormalized features



Coordinate descent for lasso (for normalized features)

```
Initialize $\hat{\mathbf{w}} = 0$ (or smartly...)
    while not converged
    for j=0,1,...,D
        compute: $\rho_j = \sum_{i=1}^{N} h_j(\mathbf{x}_i)(y_i - \hat{y}_i(\hat{\mathbf{w}}_{-j}))$
        set: $\hat{w}_j = \begin{cases} \rho_j + \lambda/2 & \text{if } \rho_j < -\lambda/2 \\ 0 & \text{if } \rho_j \text{ in } [-\lambda/2, \lambda/2] \\ \rho_j - \lambda/2 & \text{if } \rho_j > \lambda/2 \end{cases}$
```

$$\rho_j = \sum_{i=1}^{N} h_j(\mathbf{x}_i)\underbrace{(y_i - \underset{\substack{\uparrow\\ \text{prediction}\\ \textit{without}\text{ feature j}}}{\hat{y}_i(\hat{\mathbf{w}}_{-j})})}_{\text{residual } \textit{without} \text{ feature j}}$$

Coordinate descent for lasso (for unnormalized features)

```
Precompute: $z_j = \sum_{i=1}^{N} h_j(\mathbf{x}_i)^2$
Initialize $\hat{\mathbf{w}} = 0$ (or smartly...)
    while not converged
    for j=0,1,...,D
        compute: $\rho_j = \sum_{i=1}^{N} h_j(\mathbf{x}_i)(y_i - \hat{y}_i(\hat{\mathbf{w}}_{-j}))$
        set: $\hat{w}_j = \begin{cases} (\rho_j + \lambda/2)/z_j & \text{if } \rho_j < -\lambda/2 \\ 0 & \text{if } \rho_j \text{ in } [-\lambda/2, \lambda/2] \\ (\rho_j - \lambda/2)/z_j & \text{if } \rho_j > \lambda/2 \end{cases}$
```

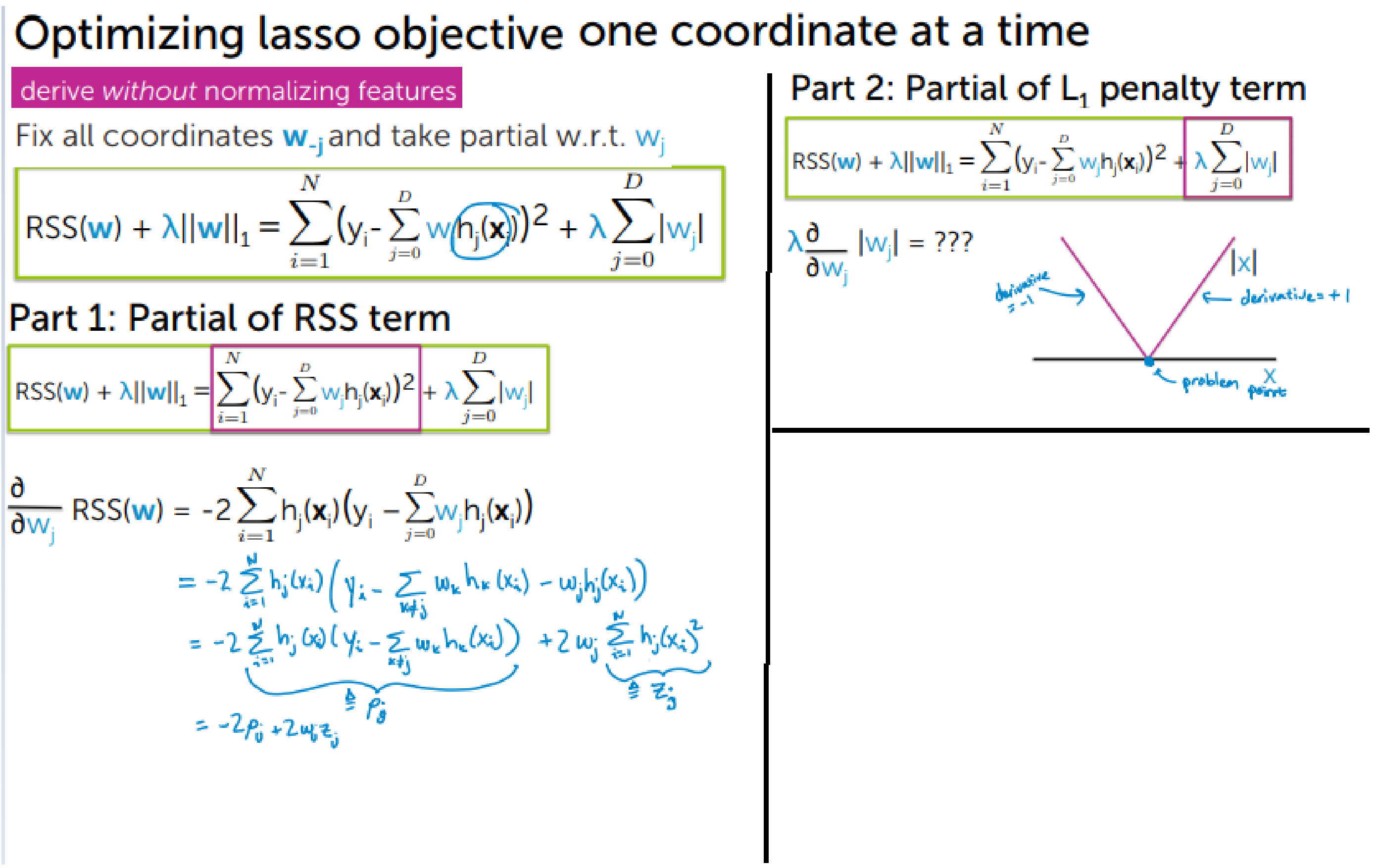
Coordinate descent for least squares regression

```
Initialize $\hat{\mathbf{w}} = 0$ (or smartly...)
    while not converged
    for j=0,1,...,D
        compute: $\rho_j = \sum_{i=1}^{N} h_j(\mathbf{x}_i)(y_i - \hat{y}_i(\hat{\mathbf{w}}_{-j}))$
        set: $\hat{w}_j = \boxed{\rho_j}$
```

$$\rho_j = \sum_{i=1}^{N} h_j(\mathbf{x}_i)\underbrace{(y_i - \underset{\substack{\uparrow\\ \text{prediction}\\ \textit{without}\text{ feature j}}}{\hat{y}_i(\hat{\mathbf{w}}_{-j})})}_{\text{residual } \textit{without} \text{ feature j}}$$

Deriving the lasso coordinate descent update

- Here unlike before considering unnormalized features, and therefore the sum of the square of the features will **not** be 0.



Optimizing lasso objective one coordinate at a time

derive *without* normalizing features

Fix all coordinates $\mathbf{w}_{-j}$ and take partial w.r.t. w_j

$$\boxed{RSS(\mathbf{w}) + \lambda\|\mathbf{w}\|_1 = \sum_{i=1}^{N}\Big(y_i - \sum_{j=0}^{D} w_j h_j(\mathbf{x}_i)\Big)^2 + \lambda\sum_{j=0}^{D}|w_j|}$$

Part 1: Partial of RSS term

$$RSS(\mathbf{w}) + \lambda\|\mathbf{w}\|_1 = \boxed{\sum_{i=1}^{N}\Big(y_i - \sum_{j=0}^{D} w_j h_j(\mathbf{x}_i)\Big)^2} + \lambda\sum_{j=0}^{D}|w_j|$$

$$\frac{\partial}{\partial w_j} RSS(\mathbf{w}) = -2\sum_{i=1}^{N} h_j(\mathbf{x}_i)\Big(y_i - \sum_{j=0}^{D} w_j h_j(\mathbf{x}_i)\Big)$$

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

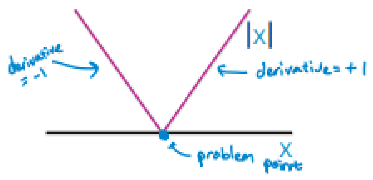
$$= \underbrace{-2\sum_{i=1}^{N} h_j(x_i)\Big(y_i - \sum_{k\neq j} w_k h_k(x_i)\Big)}_{\triangleq\ \rho_j} + 2w_j \underbrace{\sum_{i=1}^{N} h_j(x_i)^2}_{\triangleq\ z_j}$$

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

Part 2: Partial of L₁ penalty term

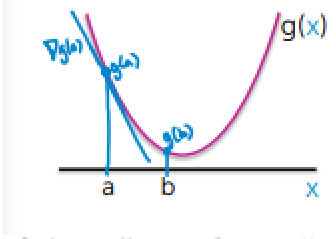
$$RSS(\mathbf{w}) + \lambda\|\mathbf{w}\|_1 = \sum_{i=1}^{N}\Big(y_i - \sum_{j=0}^{D} w_j h_j(\mathbf{x}_i)\Big)^2 + \boxed{\lambda\sum_{j=0}^{D}|w_j|}$$

$$\lambda\frac{\partial}{\partial w_j}|w_j| = ???$$



Subgradients of convex functions

Gradients lower bound convex functions:

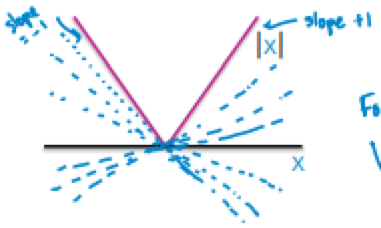


$g(b) \geq g(a) + \underline{\nabla g(a)}(b-a)$

unique at x if function differentiable at x

Subgradients: Generalize gradients to non-differentiable points:

- Any plane that lower bounds function

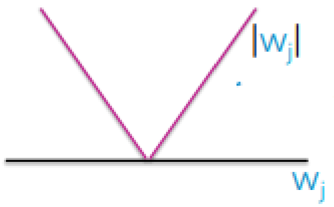


For $|x|$
$V \in [-1,1]$

$$\boxed{\begin{array}{l} V \in \partial g(x) \quad \text{subgradient of } g \text{ at } x \\ \text{if} \\ g(b) \geq g(a) + V(b-a) \end{array}}$$

Part 2: *Subgradient* of L_1 term

$$\boxed{RSS(\mathbf{w}) + \lambda\|\mathbf{w}\|_1 = \sum_{i=1}^{N}(y_i - \sum_{j=0}^{D} w_j h_j(\mathbf{x}_i))^2} + \boxed{\lambda \sum_{j=0}^{D} |w_j|}$$

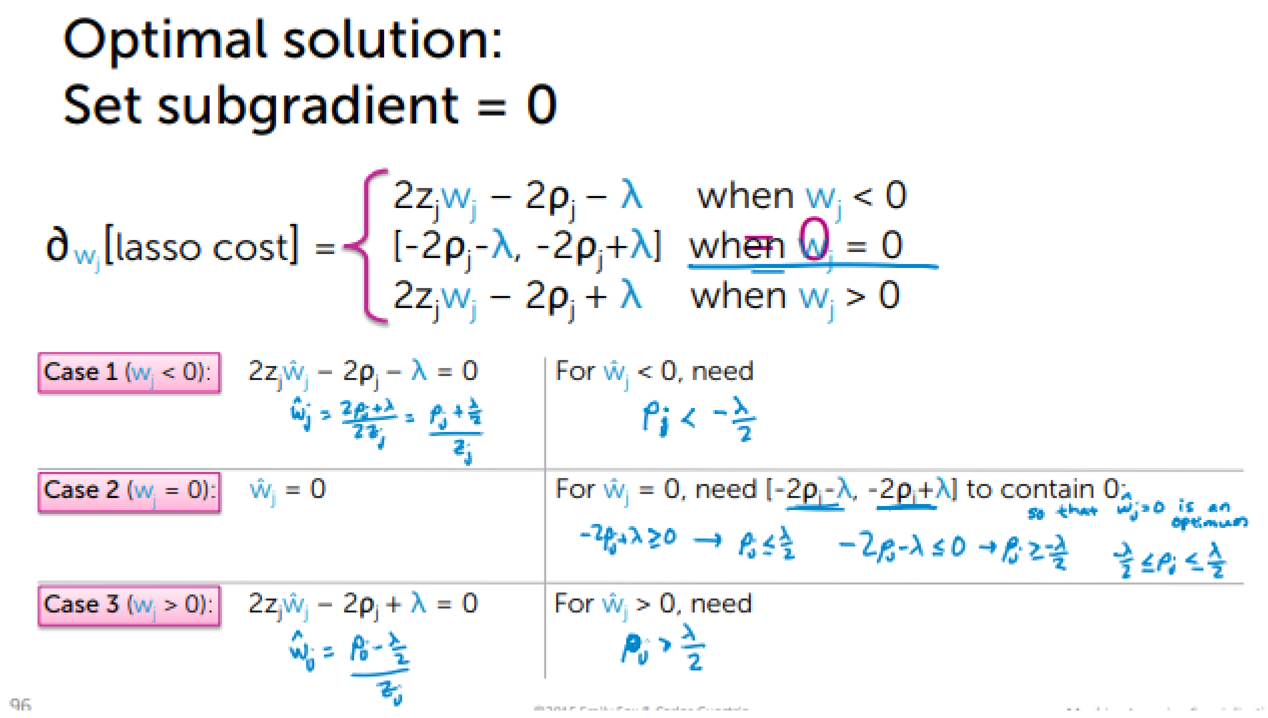


$$\lambda \partial_{w_j}|w_j| = \begin{cases} -\lambda \xleftarrow{\lambda \cdot -1} & \text{when } w_j < 0 \\ [-\lambda,\lambda] \xleftarrow{\lambda \cdot [-1,1]} & \text{when } w_j = 0 \\ \lambda \xleftarrow{\lambda \cdot 1} & \text{when } w_j > 0 \end{cases}$$

Putting it all together...

$$\partial_{w_j}[\text{lasso cost}] = \overbrace{2z_j w_j - 2\rho_j}^{\text{from RSS}} + \overbrace{\begin{cases} -\lambda & \text{when } w_j < 0 \\ [-\lambda, \lambda] & \text{when } w_j = 0 \\ \lambda & \text{when } w_j > 0 \end{cases}}^{\text{from } \lambda\|\cdot\|_1 \text{ penalty}}$$

$$= \begin{cases} 2z_j w_j - 2\rho_j - \lambda & \text{when } w_j < 0 \\ [-2\rho_j - \lambda, -2\rho_j + \lambda] & \text{when } w_j = 0 \\ 2z_j w_j - 2\rho_j + \lambda & \text{when } w_j > 0 \end{cases}$$

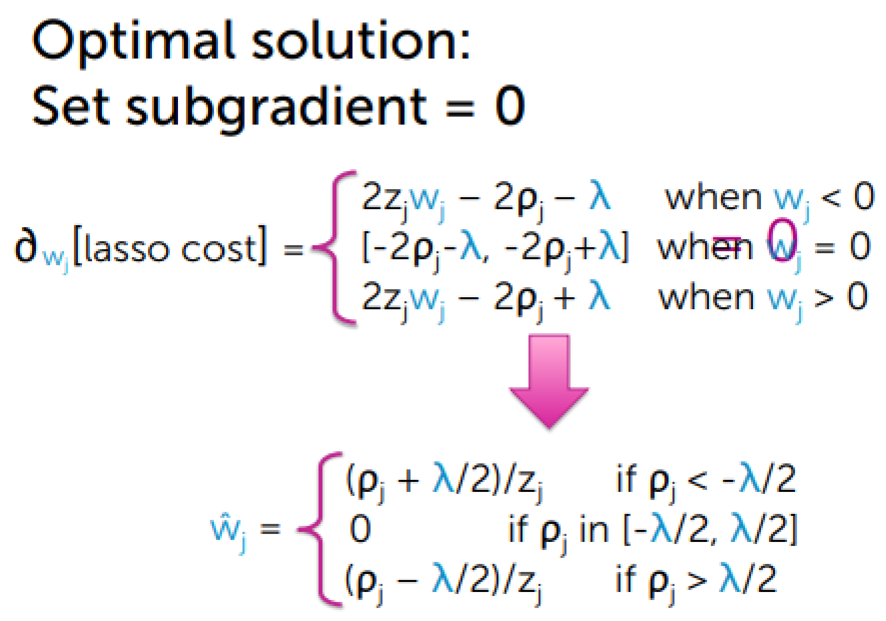


Optimal solution: Set subgradient = 0

$$\partial_{w_j}[\text{lasso cost}] = \begin{cases} 2z_j w_j - 2\rho_j - \lambda & \text{when } w_j < 0 \\ [-2\rho_j-\lambda,\ -2\rho_j+\lambda] & \text{when } w_j = 0 \\ 2z_j w_j - 2\rho_j + \lambda & \text{when } w_j > 0 \end{cases}$$

Case 1 ($w_j < 0$):	$2z_j\hat{w}_j - 2\rho_j - \lambda = 0$ $\hat{w}_j = \frac{2\rho_j+\lambda}{2z_j} = \frac{\rho_j+\frac{\lambda}{2}}{z_j}$	For $\hat{w}_j < 0$, need $\rho_j < -\frac{\lambda}{2}$
Case 2 ($w_j = 0$):	$\hat{w}_j = 0$	For $\hat{w}_j = 0$, need $[\underline{-2\rho_j-\lambda},\ \underline{-2\rho_j+\lambda}]$ to contain 0, so that $\hat{w}_j = 0$ is an optimum $-2\rho_j+\lambda \geq 0 \rightarrow \rho_j \leq \frac{\lambda}{2}$ $\quad -2\rho_j-\lambda \leq 0 \rightarrow \rho_j \geq -\frac{\lambda}{2}$ $\quad -\frac{\lambda}{2} \leq \rho_j \leq \frac{\lambda}{2}$
Case 3 ($w_j > 0$):	$2z_j\hat{w}_j - 2\rho_j + \lambda = 0$ $\hat{w}_j = \frac{\rho_j-\frac{\lambda}{2}}{z_j}$	For $\hat{w}_j > 0$, need $\rho_j > \frac{\lambda}{2}$

96

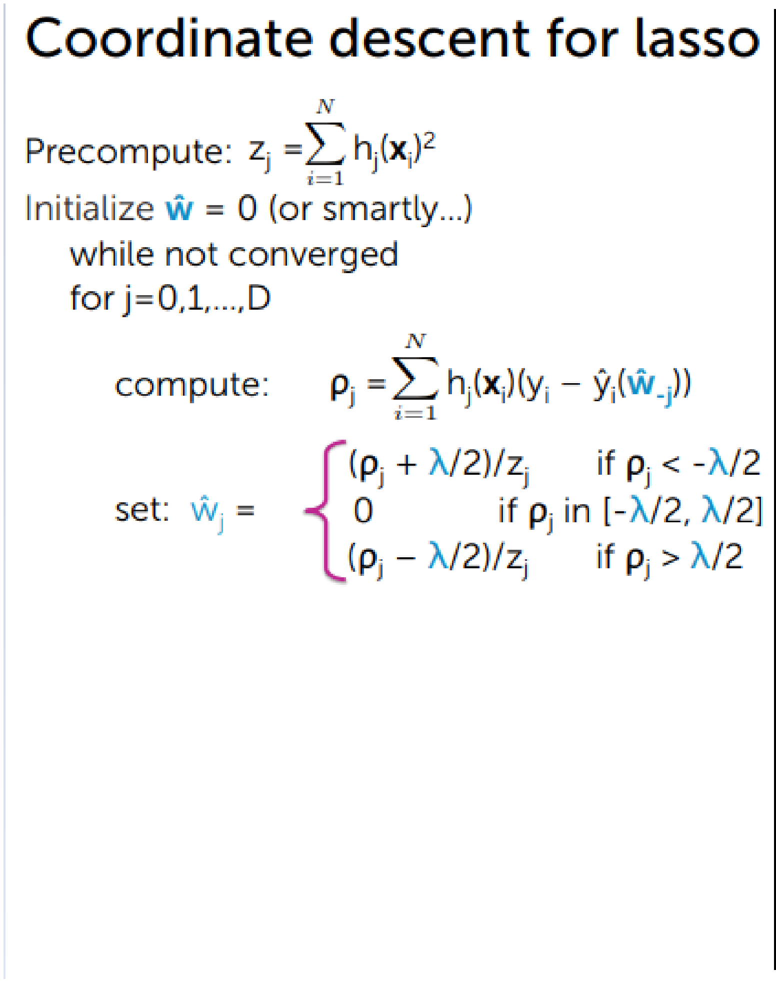


Optimal solution: Set subgradient = 0

$$\partial_{w_j}[\text{lasso cost}] = \begin{cases} 2z_j w_j - 2\rho_j - \lambda & \text{when } w_j < 0 \\ [-2\rho_j-\lambda,\ -2\rho_j+\lambda] & \text{when } w_j = 0 \\ 2z_j w_j - 2\rho_j + \lambda & \text{when } w_j > 0 \end{cases}$$

$$\Big\Downarrow$$

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

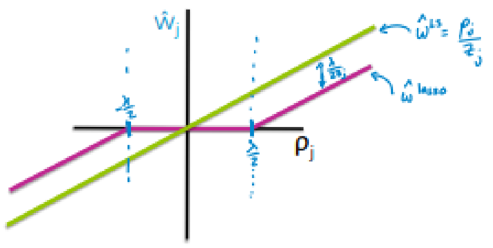


Coordinate descent for lasso

```
Precompute: $z_j = \sum_{i=1}^{N} h_j(\mathbf{x}_i)^2$
Initialize $\hat{\mathbf{w}} = 0$ (or smartly...)
    while not converged
    for j=0,1,...,D
        compute: $\rho_j = \sum_{i=1}^{N} h_j(\mathbf{x}_i)(y_i - \hat{y}_i(\hat{\mathbf{w}}_{-j}))$
        set: $\hat{w}_j = \begin{cases} (\rho_j + \lambda/2)/z_j & \text{if } \rho_j < -\lambda/2 \\ 0 & \text{if } \rho_j \text{ in } [-\lambda/2, \lambda/2] \\ (\rho_j - \lambda/2)/z_j & \text{if } \rho_j > \lambda/2 \end{cases}$
```

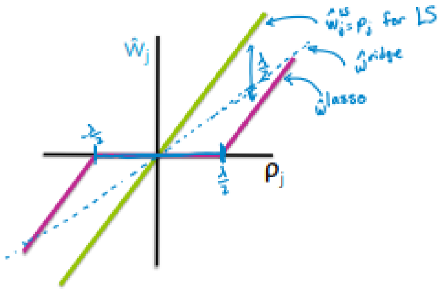
Soft thresholding UNORMALIZED

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



Soft thresholding NORMALIZED

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



How to choose λ

- Incase of sufficient amount of data:
  - Use the training set to fit the estimated parameters w-hat for λ.
  - Use the validation set to test the performance of w-hat λ to select λ* that minimizes the error.
  - Use the test set to assess generalization error of w-hat λ*.
- Incase of insufficient data - perfrom K-fold cross validation
  - For k = 1,...,k
    1. Estimate w-hat λ(k) on the training blocks.
    2. Compute the error on validation block: error-k(λ)
  - Compute average error : CV(λ) = 1/K Σ(k = 1...K) error-k(λ).

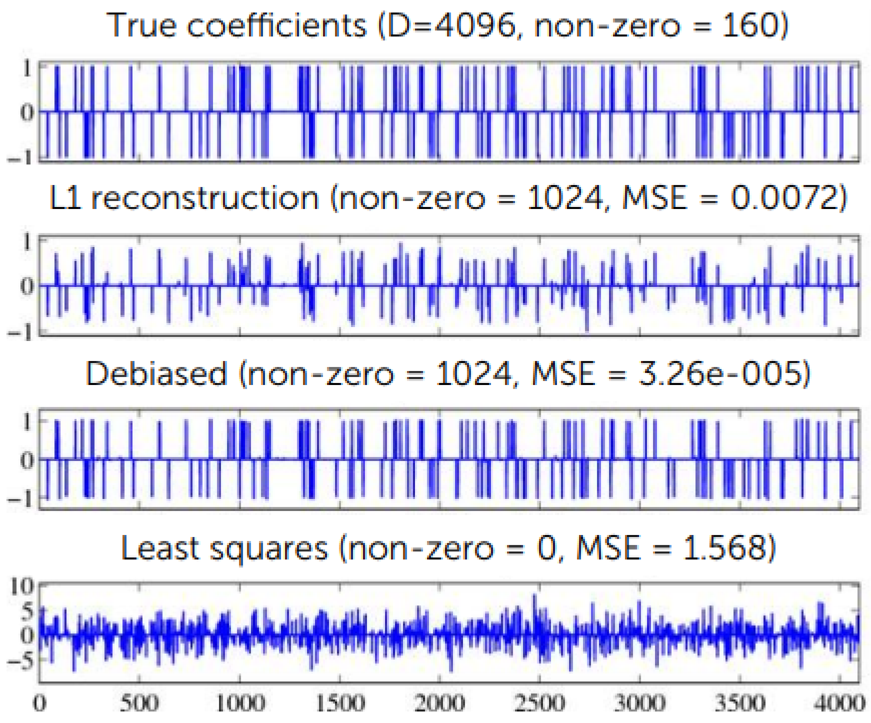
Choosing λ via cross validation

- Cross validation is choosing the λ that provides the best predictive accuracy.
- Tends to favour less sparse solutions, and thus smaller λ, than optimal choice for feature selection.

Practical concerns with lasso

1. Debiasing lasso

- Lasso shrinks the coefficients relative to the LS solution. Leads to more bias, less variance. -> fig2
- Can reduce bias as follows:
  - Run lasso to select feature. fig 3
  - Run the least square regression with only selected features. fig 4
- **Relevant features no longer shrunk relative to LS fit of the same reduced model.**



2. With group of highly correlated features, lasso tends to select amongst them arbitraily.

- Often prefers to select all together.

3. Often, empirically ridge has better predictive performance than lasso, but lasso leads to sparser solutions.

Alternative, Elastic net aims to address these issues.

- Hybrid between lasso and ridge regression.
- Uses L1 and L2 penalities

impact of feature selection and lasso:

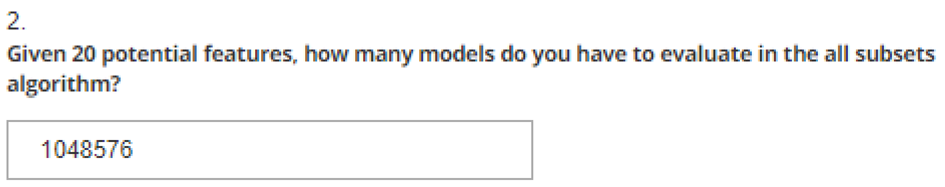
- Selection only considers features included.
- Sensitive to correlations between features.
- Results depends on algorithm used.
- There are theoretical

Quiz

1
point

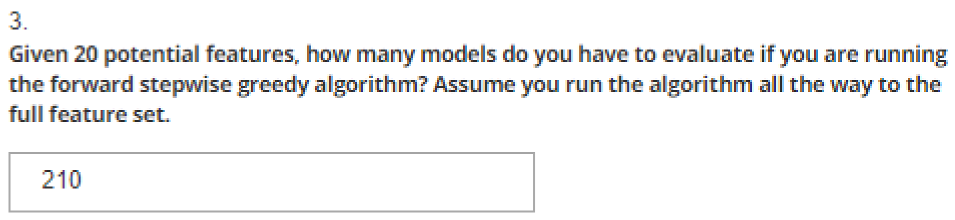
1.
The best fit model of size 5 (i.e., with 5 features) always contains the set of features from best fit model of size 4.

True [✓ False]


2.
Given 20 potential features, how many models do you have to evaluate in the all subsets algorithm?

1048576

- The all-subset - # models evaluated = 2^D = s^ 20 = 1048576;


3.
Given 20 potential features, how many models do you have to evaluate if you are running the forward stepwise greedy algorithm? Assume you run the algorithm all the way to the full feature set.

210

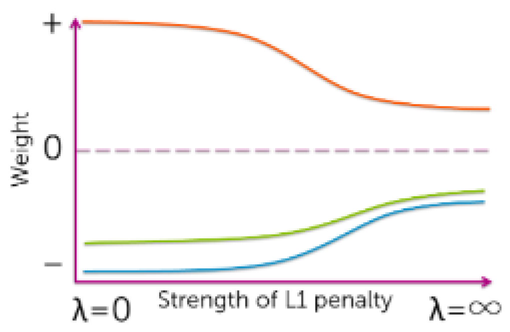
- Accord to Greedy algorithm , for 20 features:
- Step 1 : Evaluate 20 models and pick one feature out of 20.
- step 2 : Evaluate 19 possible models and pick one feature out of 19. ...
- Step 20 : Build the model.

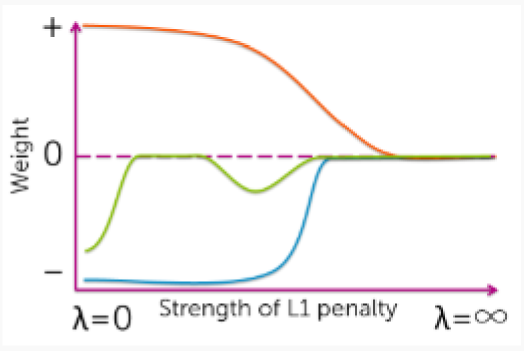
For each of the above steps 1 to n, the no of possible models evaluated deceases by 1.

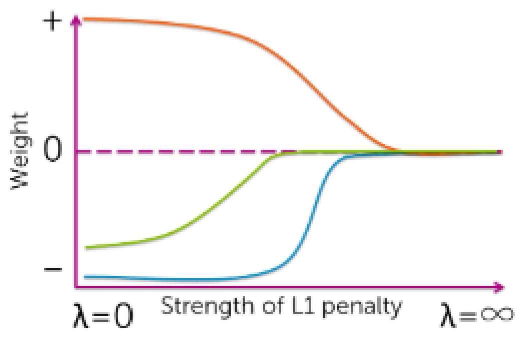
- So For 20 potential features -> # models evaluated = 20 + 19 + 18 + ... + 1 = 210.

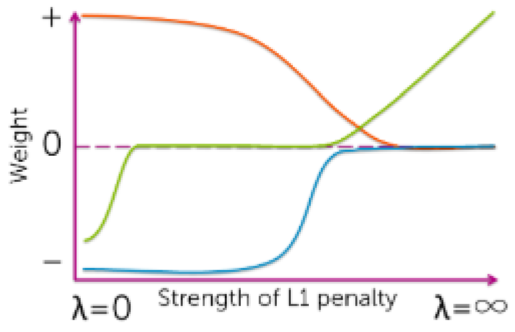
4.
Which of the plots could correspond to a lasso coefficient path? Select ALL that apply.

Hint: notice $\lambda = \infty$ in the bottom right of the plots. How should coefficients behave eventually as λ goes to infinity?

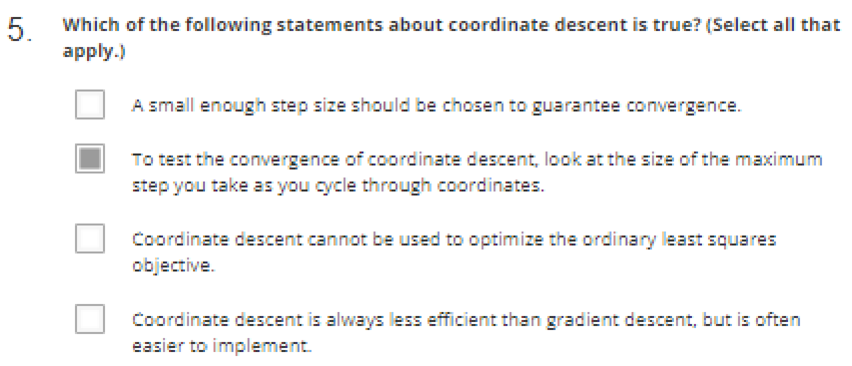
[ ]


[✓]


[✓]


[ ]


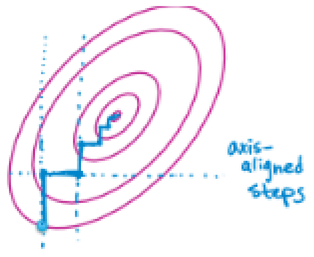
- As the λ goes to infinity, the magnitude of the coefficients becomes 0.

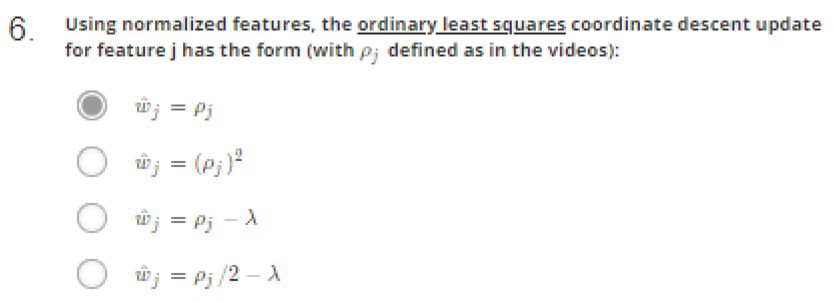


5. **Which of the following statements about coordinate descent is true? (Select all that apply.)**

- ☐ A small enough step size should be chosen to guarantee convergence.
- [✓ To test the convergence of coordinate descent, look at the size of the maximum step you take as you cycle through coordinates.]
- ☐ Coordinate descent cannot be used to optimize the ordinary least squares objective.
- ☐ Coordinate descent is always less efficient than gradient descent, but is often easier to implement.

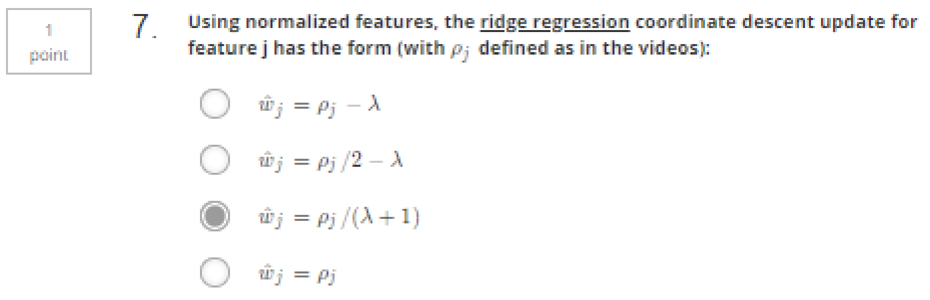
- At you approach the optimum or the minimum in the coordinate descent case, the size of the step pver all the coordinates tends to decrease





6. **Using normalized features, the <u>ordinary least squares</u> coordinate descent update for feature j has the form (with ρ_j defined as in the videos):**

- [✓ $\hat{w}_j = \rho_j$]
- ○ $\hat{w}_j = (\rho_j)^2$
- ○ $\hat{w}_j = \rho_j - \lambda$
- ○ $\hat{w}_j = \rho_j / 2 - \lambda$



1 point

7. **Using normalized features, the <u>ridge regression</u> coordinate descent update for feature j has the form (with ρ_j defined as in the videos):**

- ○ $\hat{w}_j = \rho_j - \lambda$
- ○ $\hat{w}_j = \rho_j / 2 - \lambda$
- [✓ $\hat{w}_j = \rho_j / (\lambda + 1)$]
- ○ $\hat{w}_j = \rho_j$

$$\boxed{RSS(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2 = \sum_{i=1}^{N}(y_i - w_0h_0(\mathbf{x}_i) - w_1h_1(\mathbf{x}_i))^2 + \underline{\lambda}} \boxed{(w_0^2 + w_1^2)}$$

$$\boxed{RSS(\mathbf{w}) = \sum_{i=1}^{N}(y_i - \sum_{j=0}^{D} w_j\underline{h_j}(\mathbf{x}_i))^2}$$

normalized features

Fix all coordinates $\mathbf{w}_{-j}$ and take partial w.r.t. w_j

all w_k for $k \neq j$

1d optimization coordinate by coordinate

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

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

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

by definition of normalized features, $=1$

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

```
 Second term
-------------
 λ*w ^2
derivative of the above term
 = 2* λw

Together -> ridge regression coordinate descent

 = -2ρj + 2 wj +2λ wj
 = 2(-ρj + wj(1+λ))
 wj = ρj / (1+λ)
```

In [ ]: