Lasso Regression:

Regularization for feature selection

Emily Fox & Carlos Guestrin

Machine Learning Specialization

University of Washington

Feature selection task

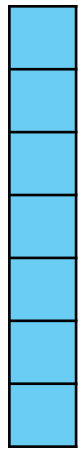
Why might you want to perform feature selection?

Efficiency:

- If size(**w**) = 100B, each prediction is expensive
- If **ŵ** sparse, computation only depends on # of non-zeros

many zeros

$$\hat{y}_i = \sum_{\hat{w}_j \neq 0} \hat{w}_j \, h_j(\mathbf{x}_i)$$



Interpretability:

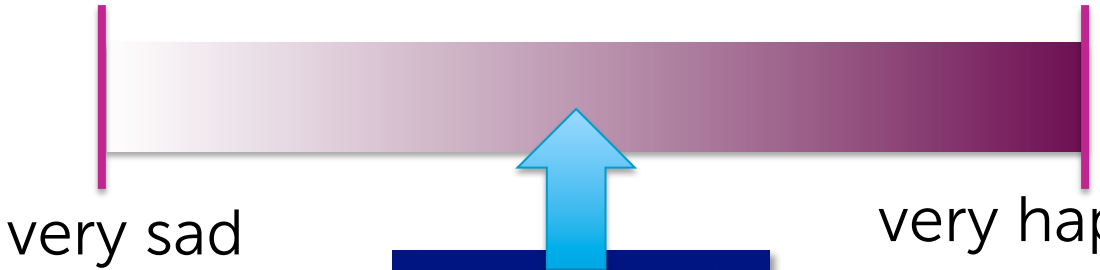
- Which features are relevant for prediction?

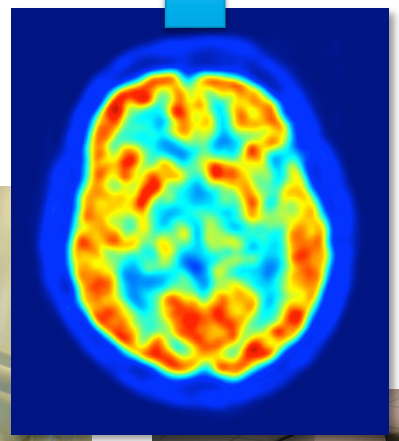
Sparsity: Housing application

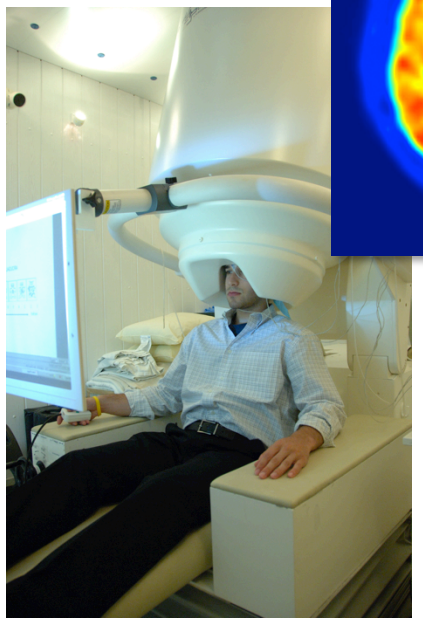
Lot size
Single Family
Year built
Last sold price
Last sale price/sqft
Finished sqft
Unfinished sqft
Finished basement sqft
floors
Flooring types
Parking type
Parking amount
Cooling
Heating
Exterior materials
Roof type
Structure style

Dishwasher
Garbage disposal
Microwave
Range / Oven
Refrigerator
Washer
Dryer
Laundry location
Heating type
Jetted Tub
Deck
Fenced Yard
Lawn
Garden
Sprinkler System
⋮

Sparsity: Reading your mind



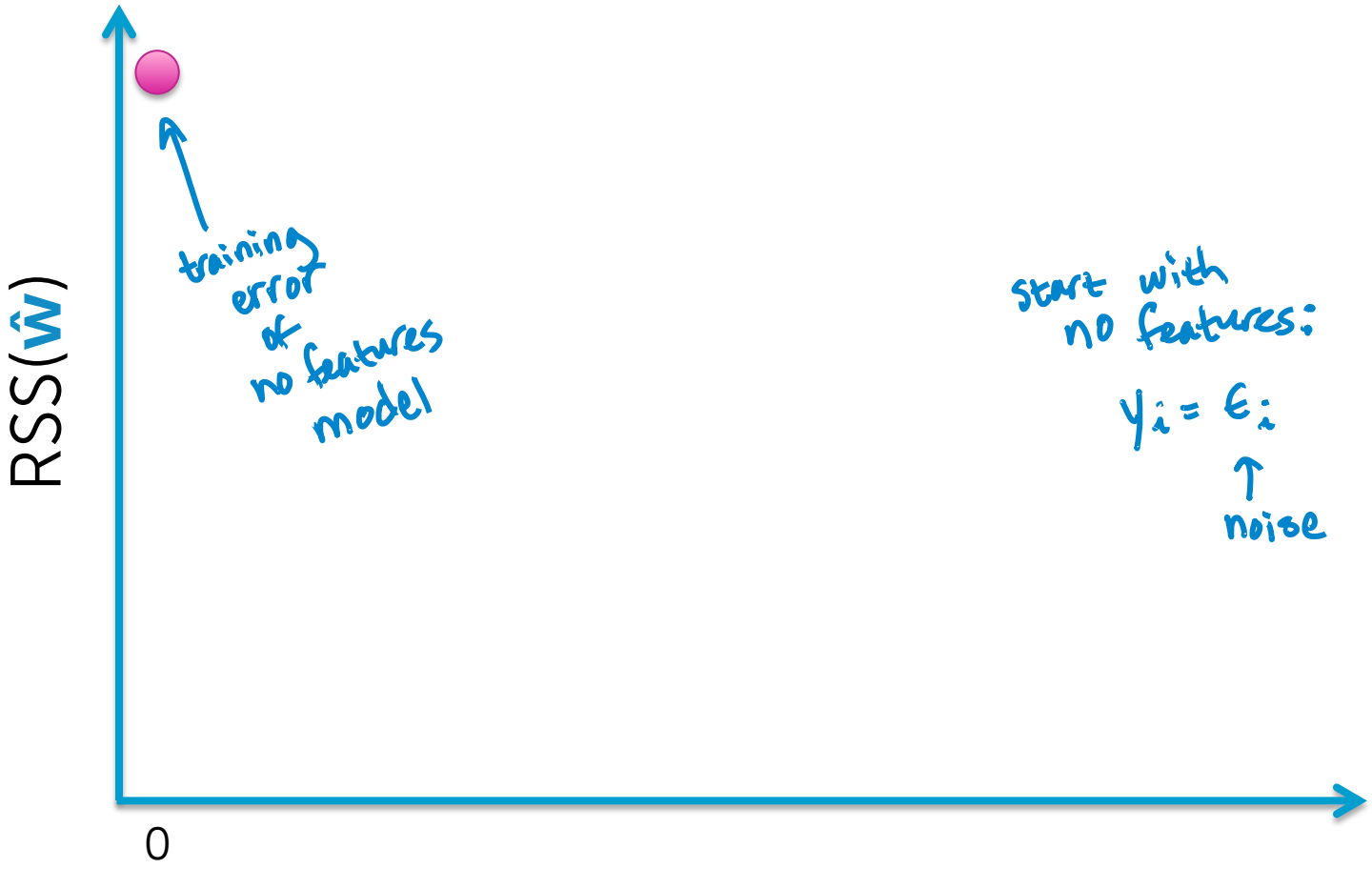




Activity in which brain regions can predict happiness?

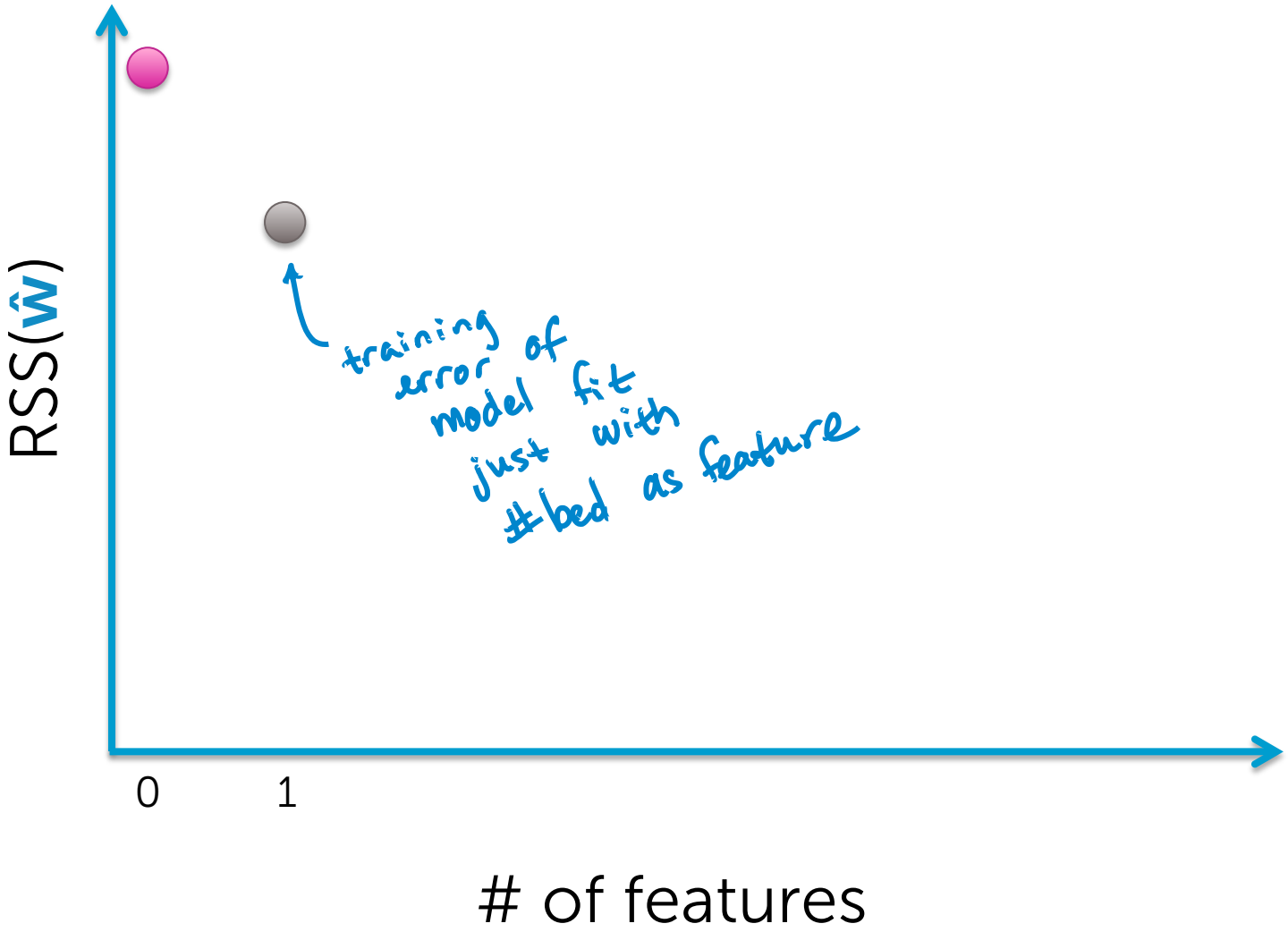
Option 1: All subsets

Find best model of size: 0



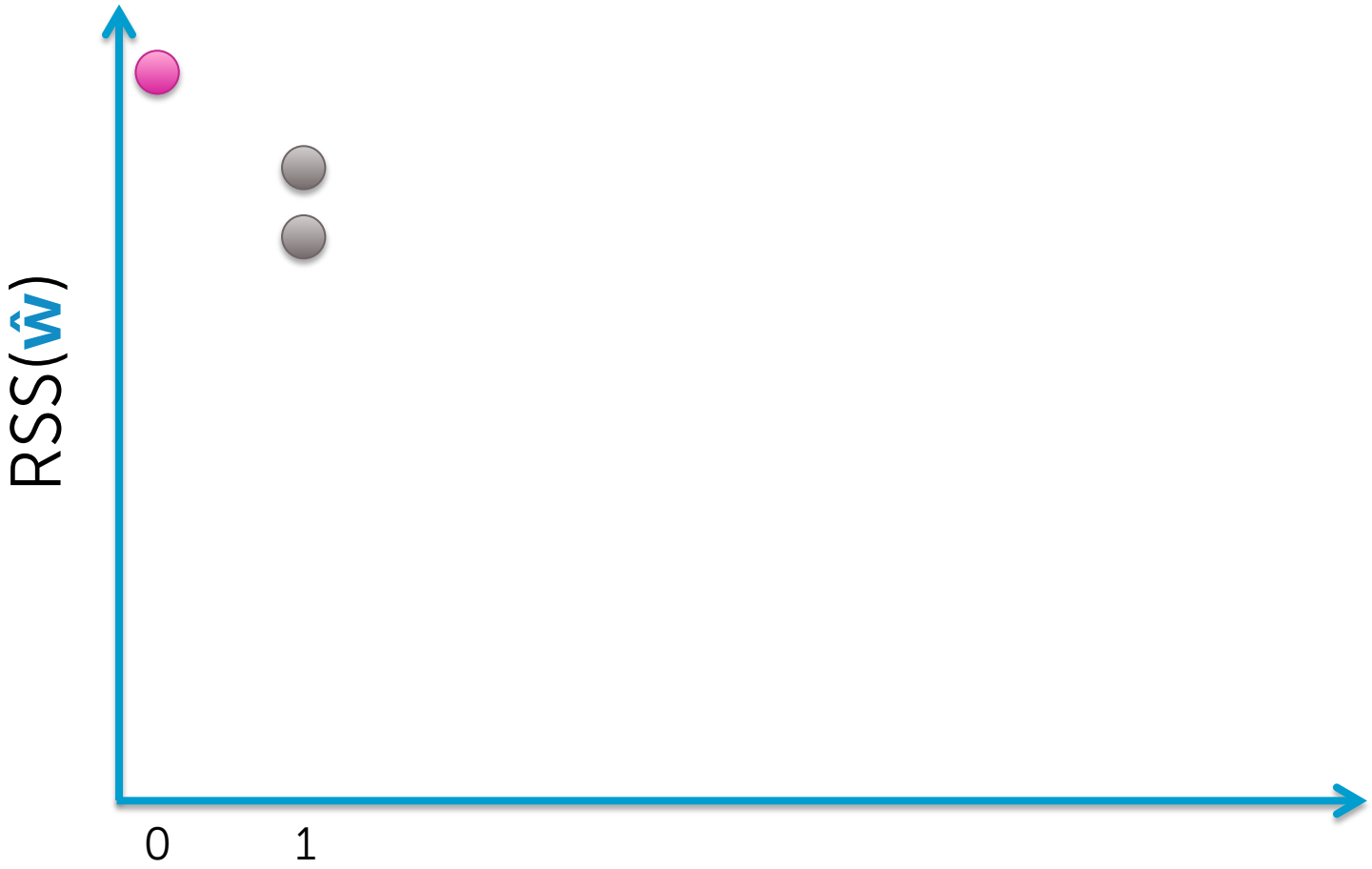
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



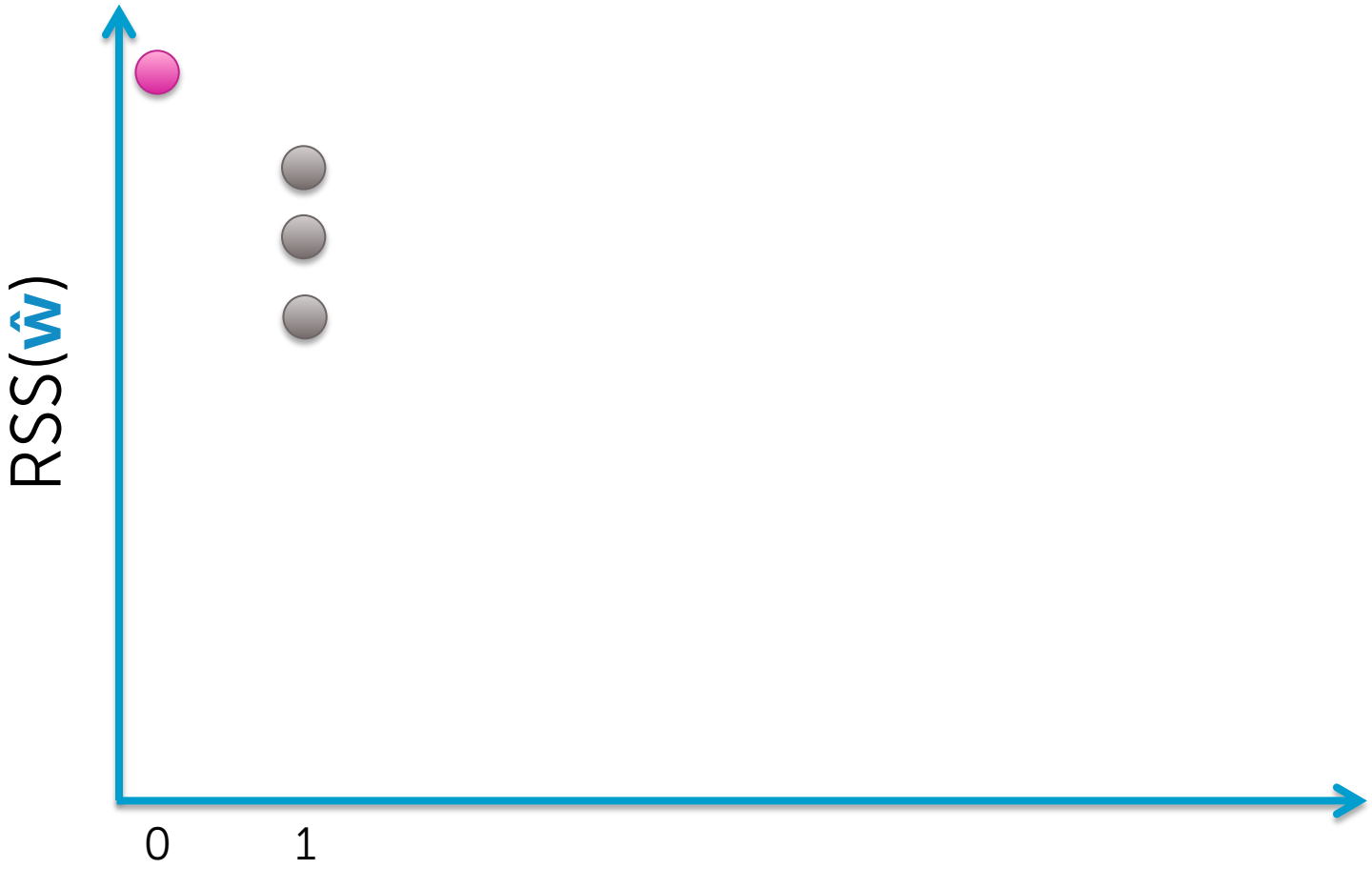
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



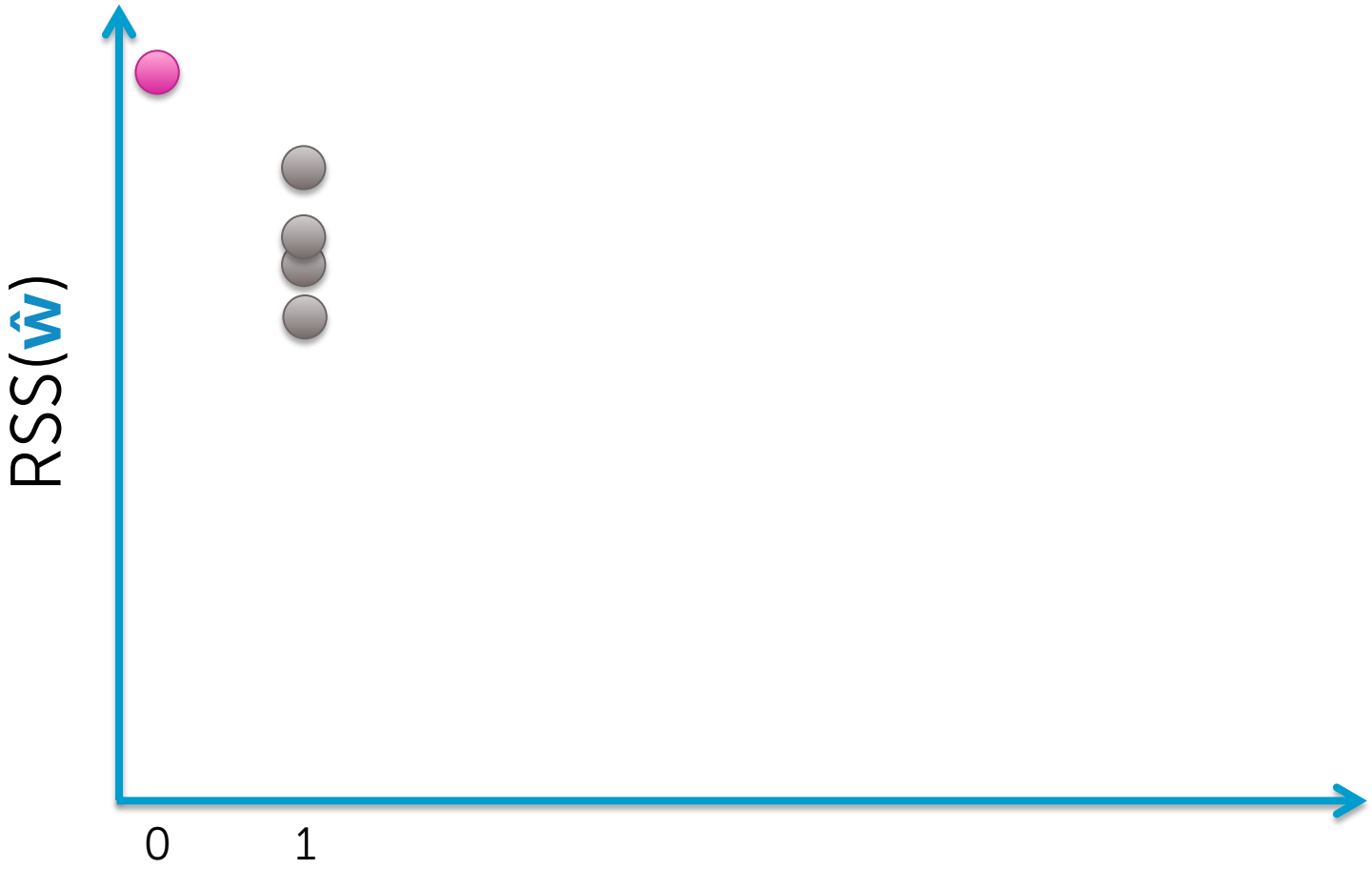
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



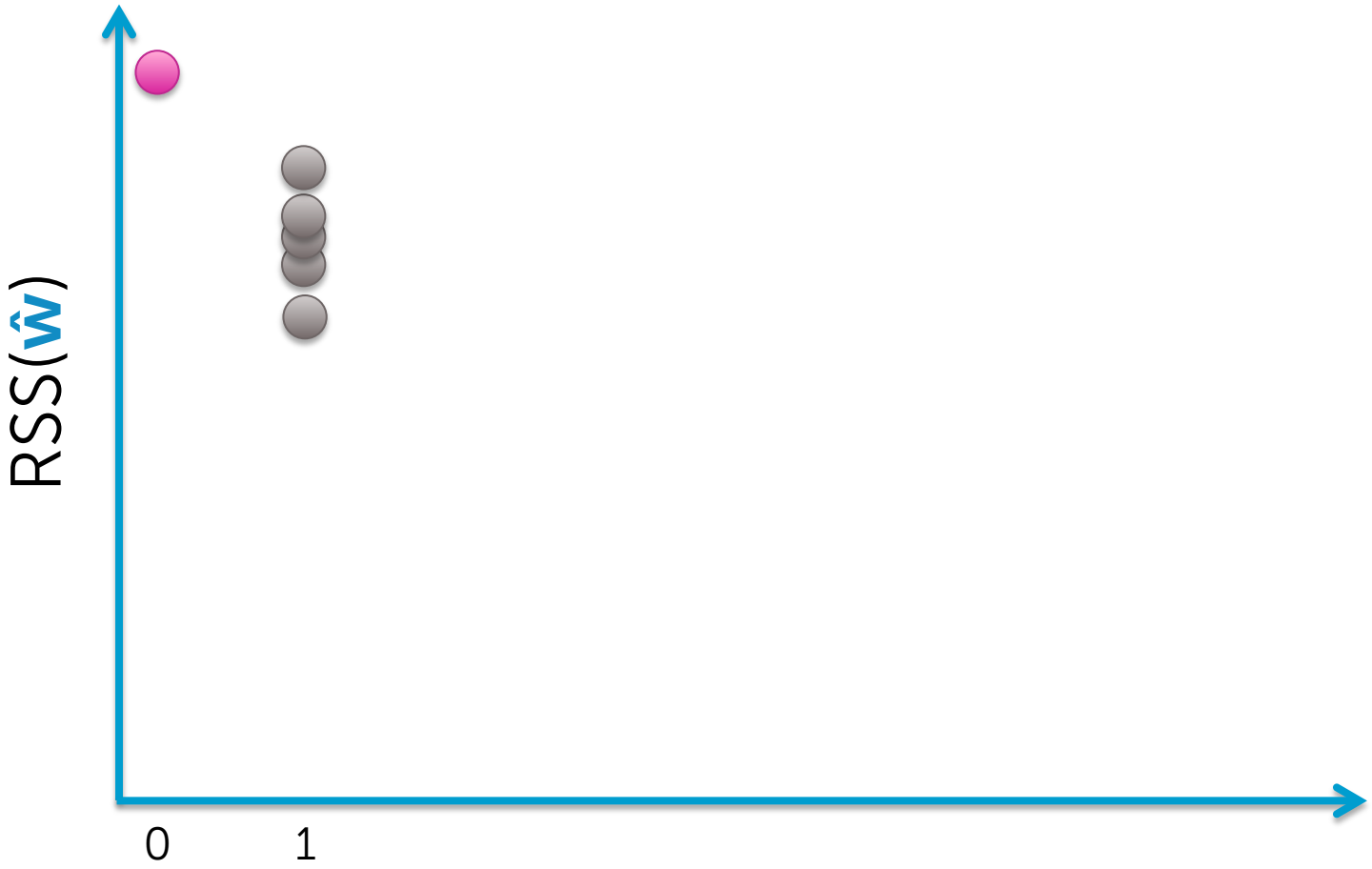
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



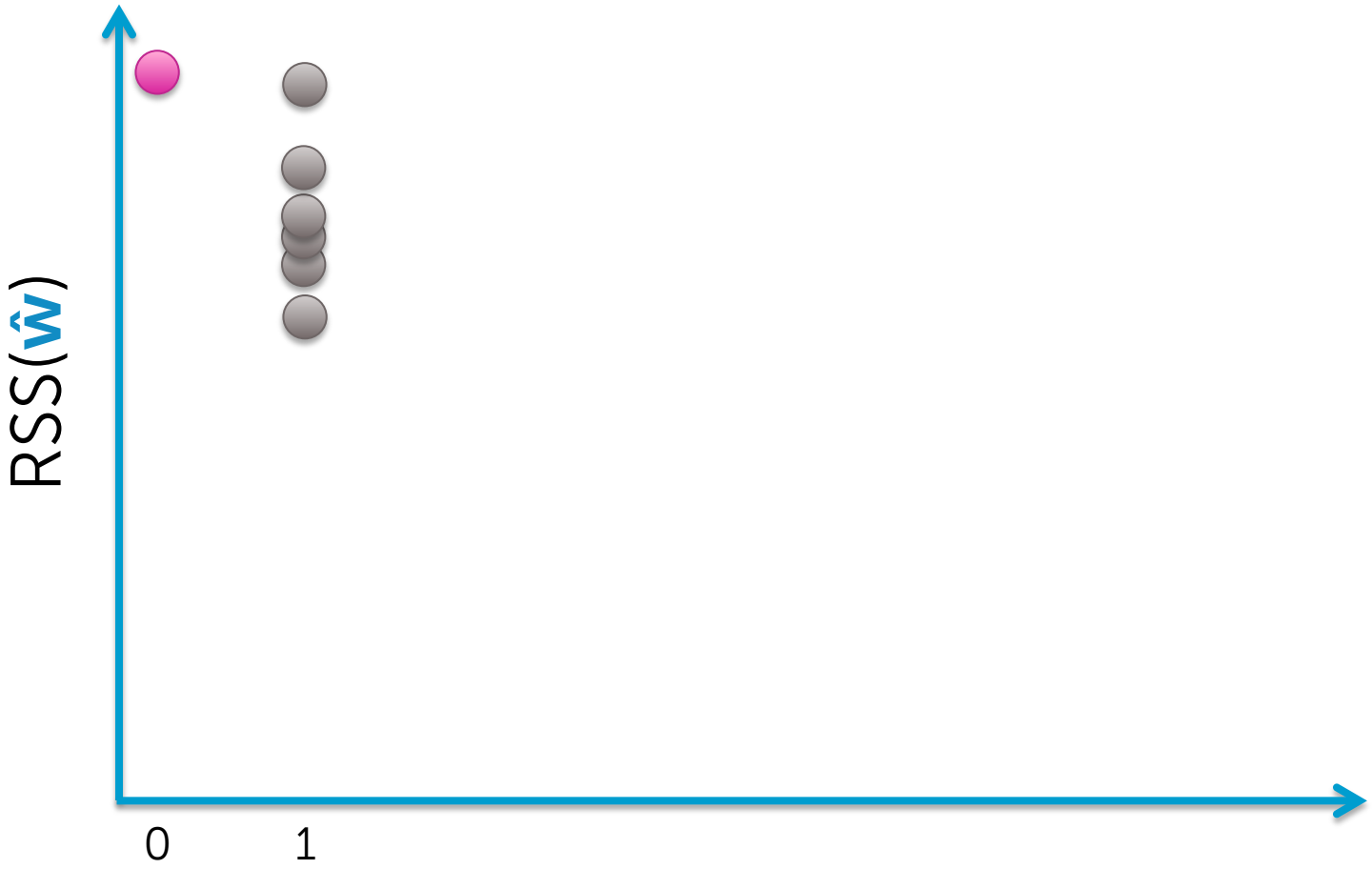
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



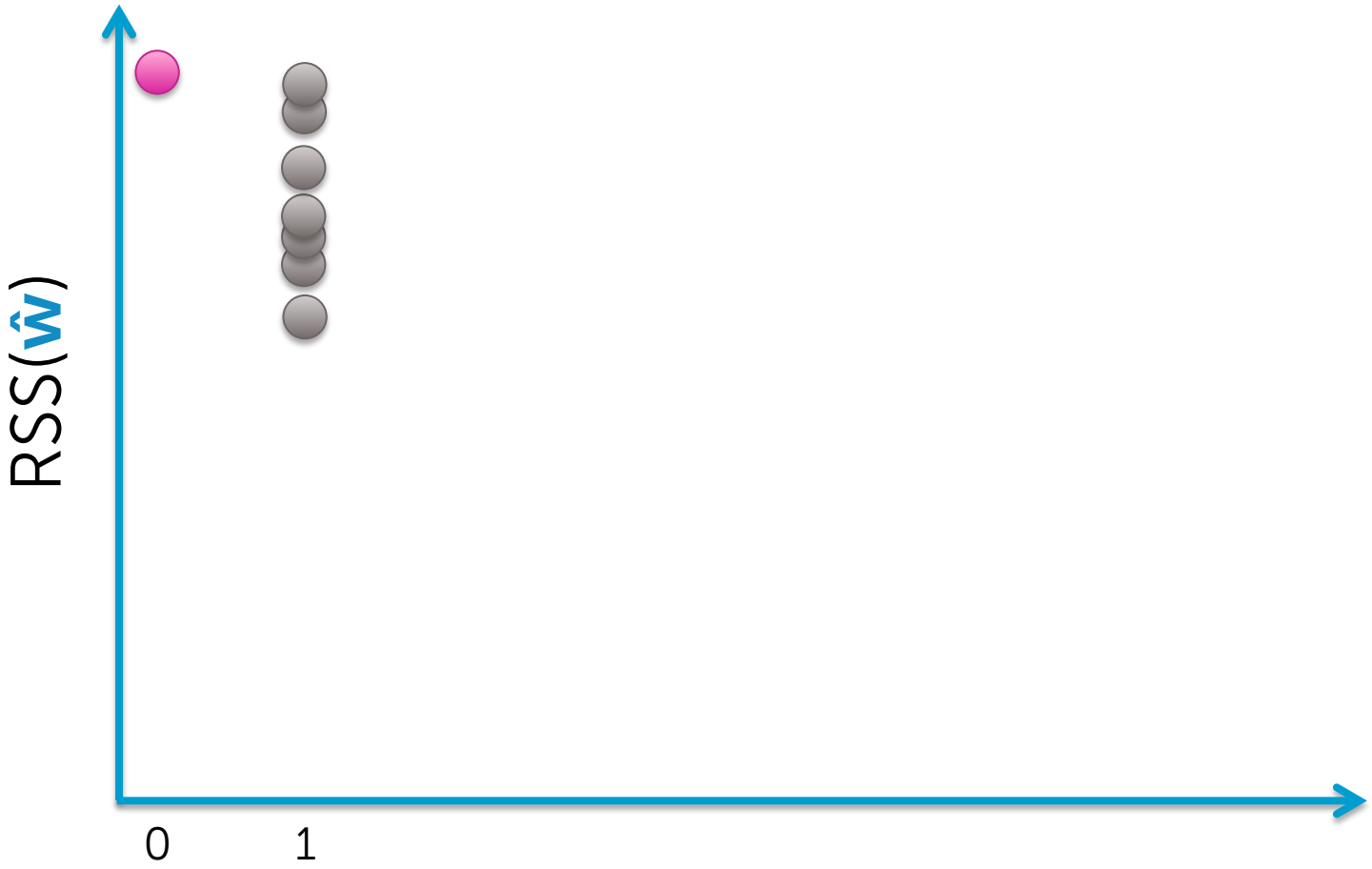
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



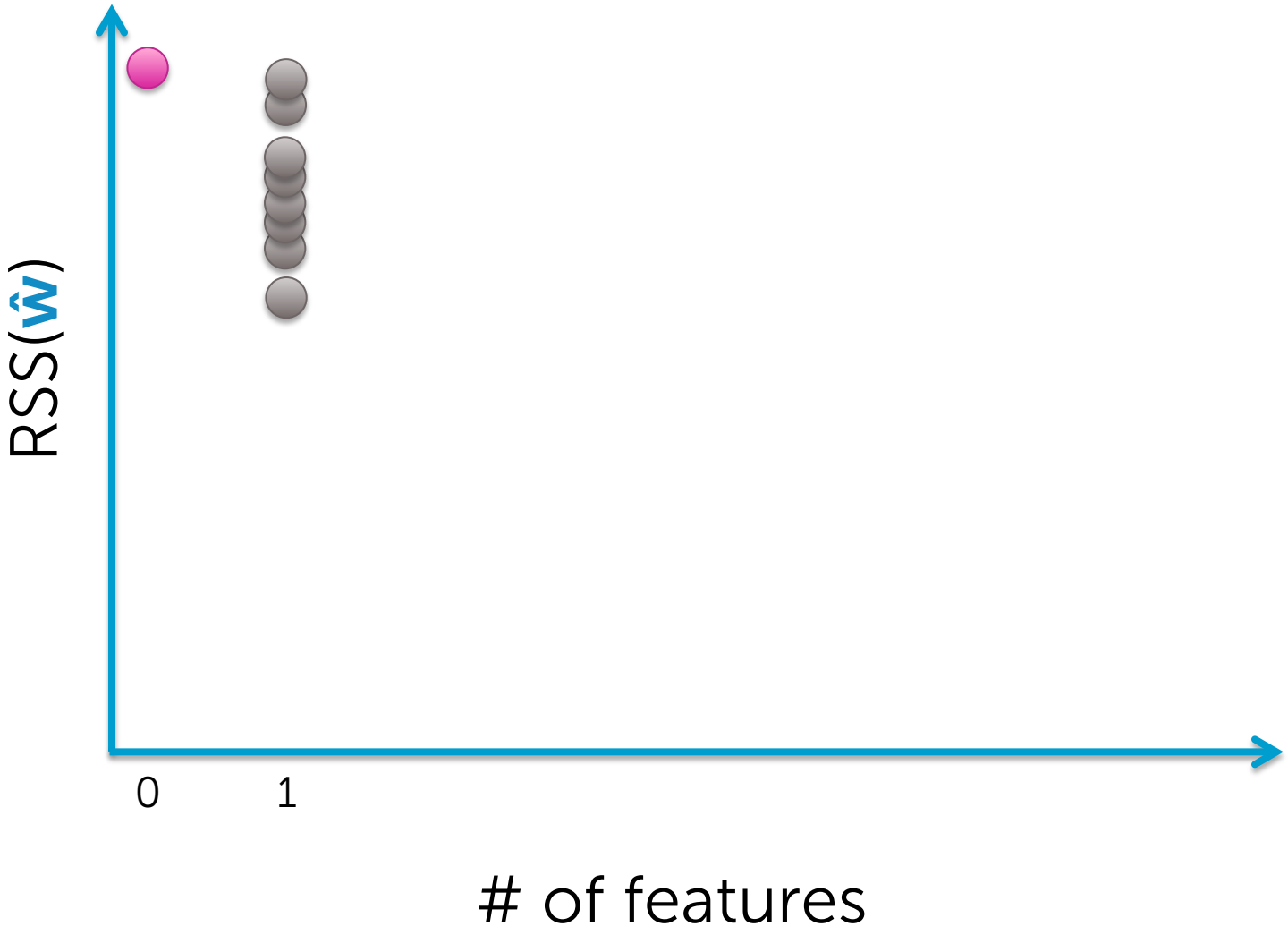
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



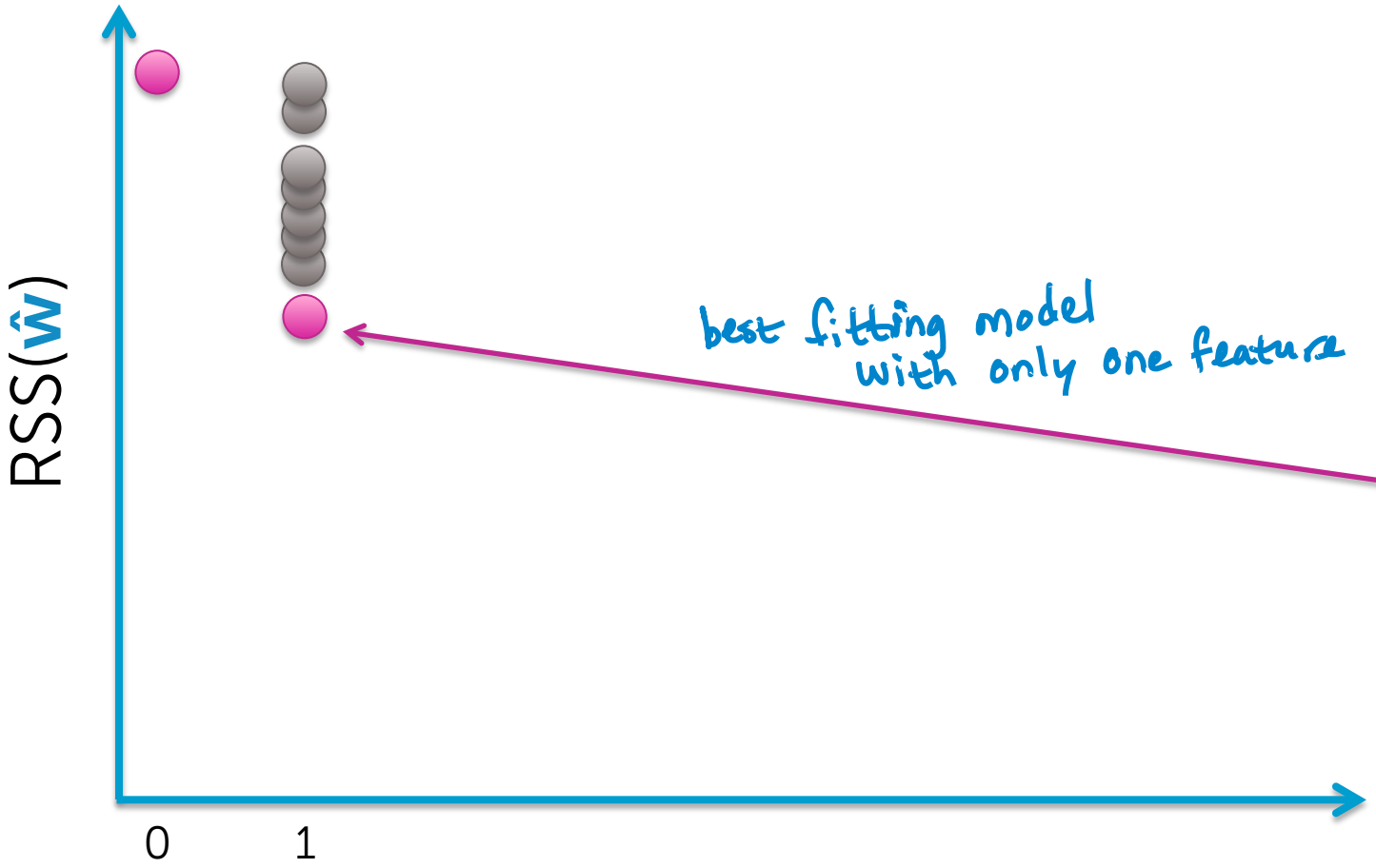
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



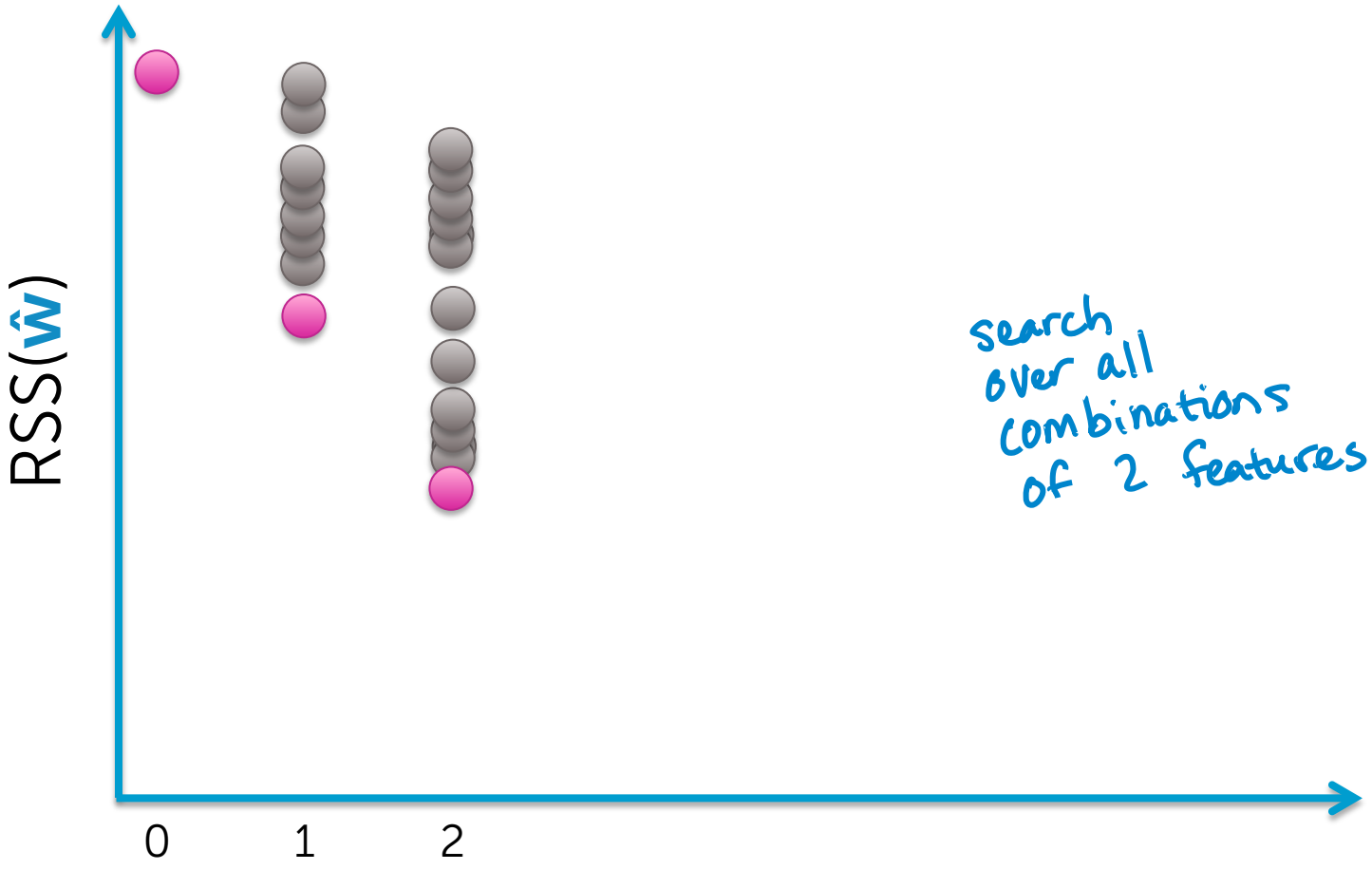
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1



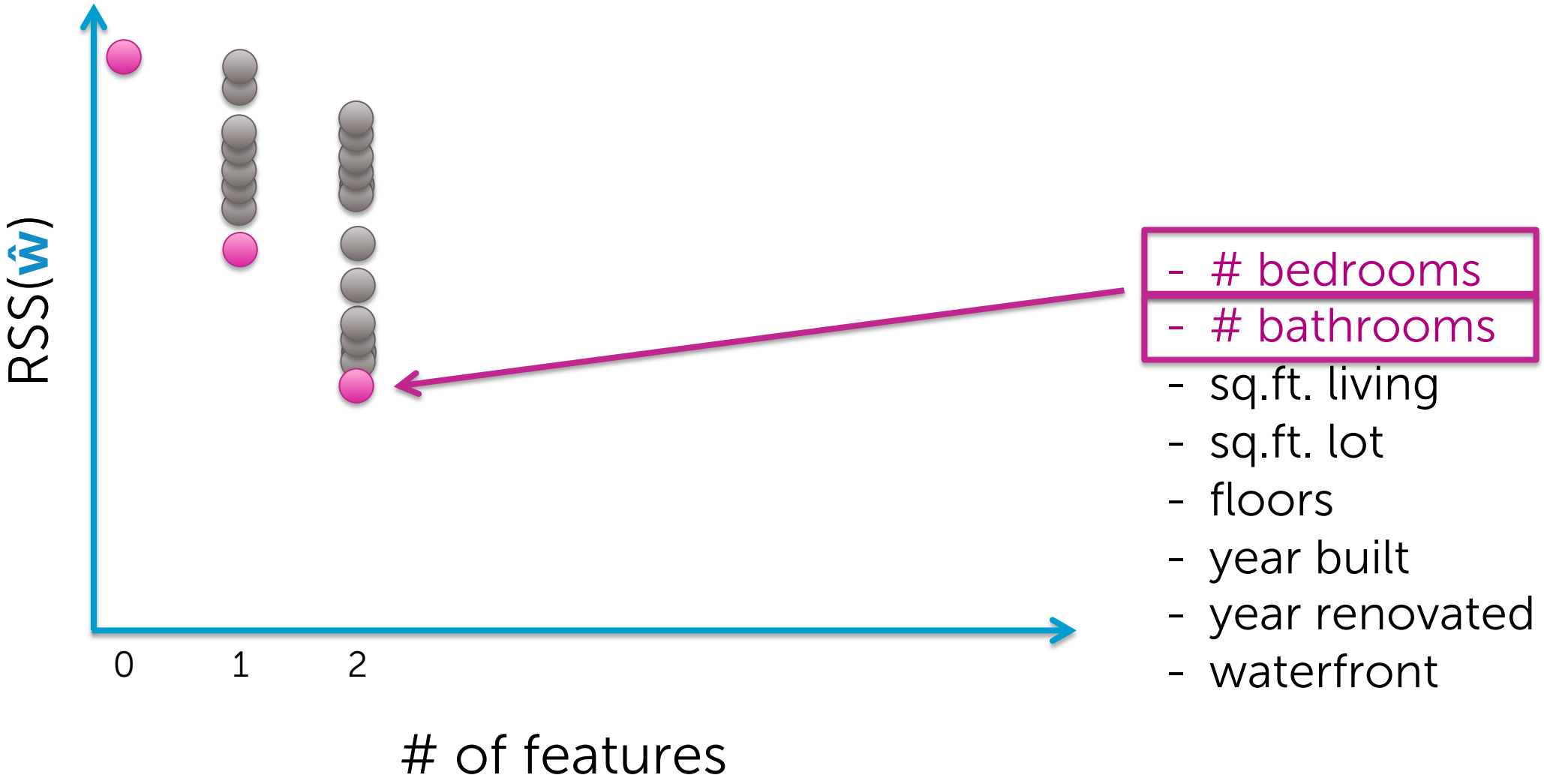
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 2

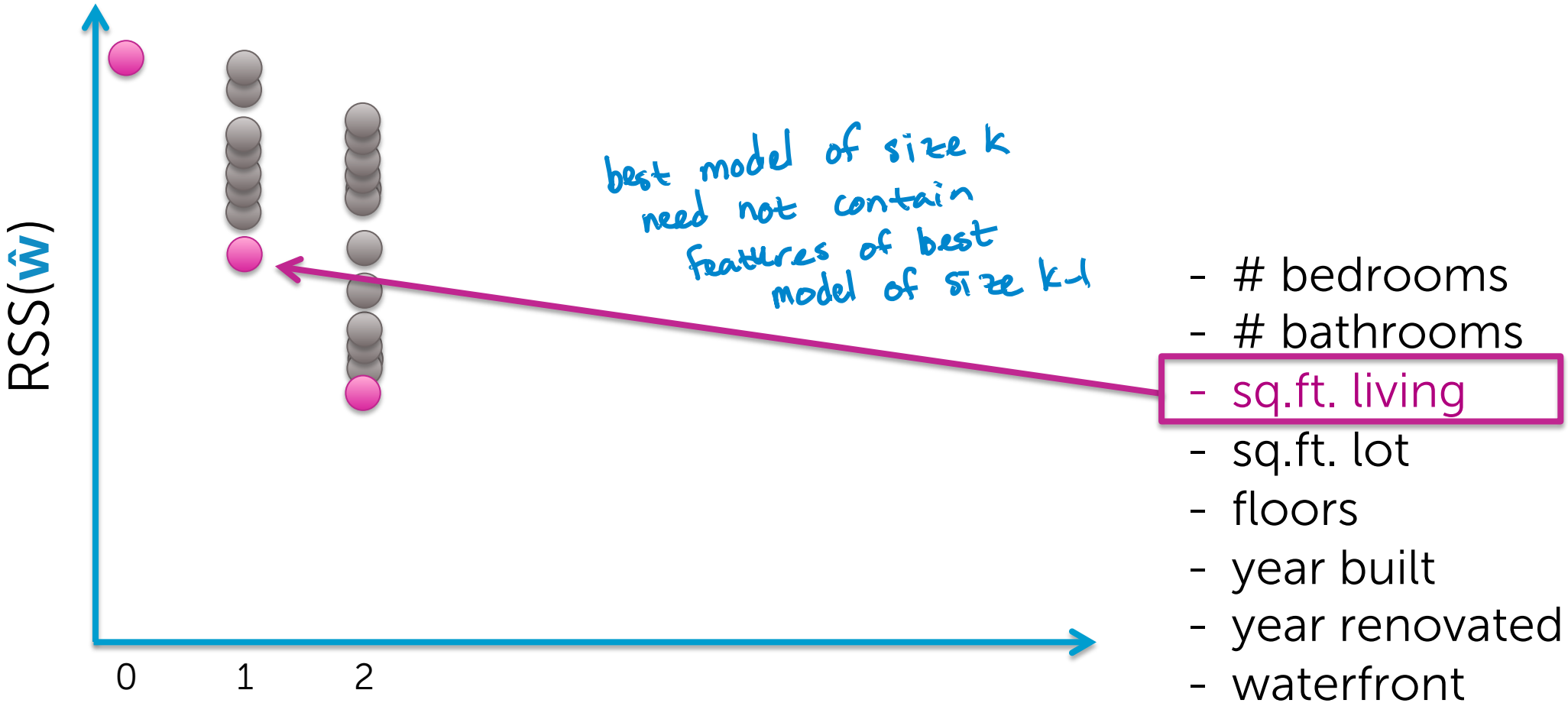


- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

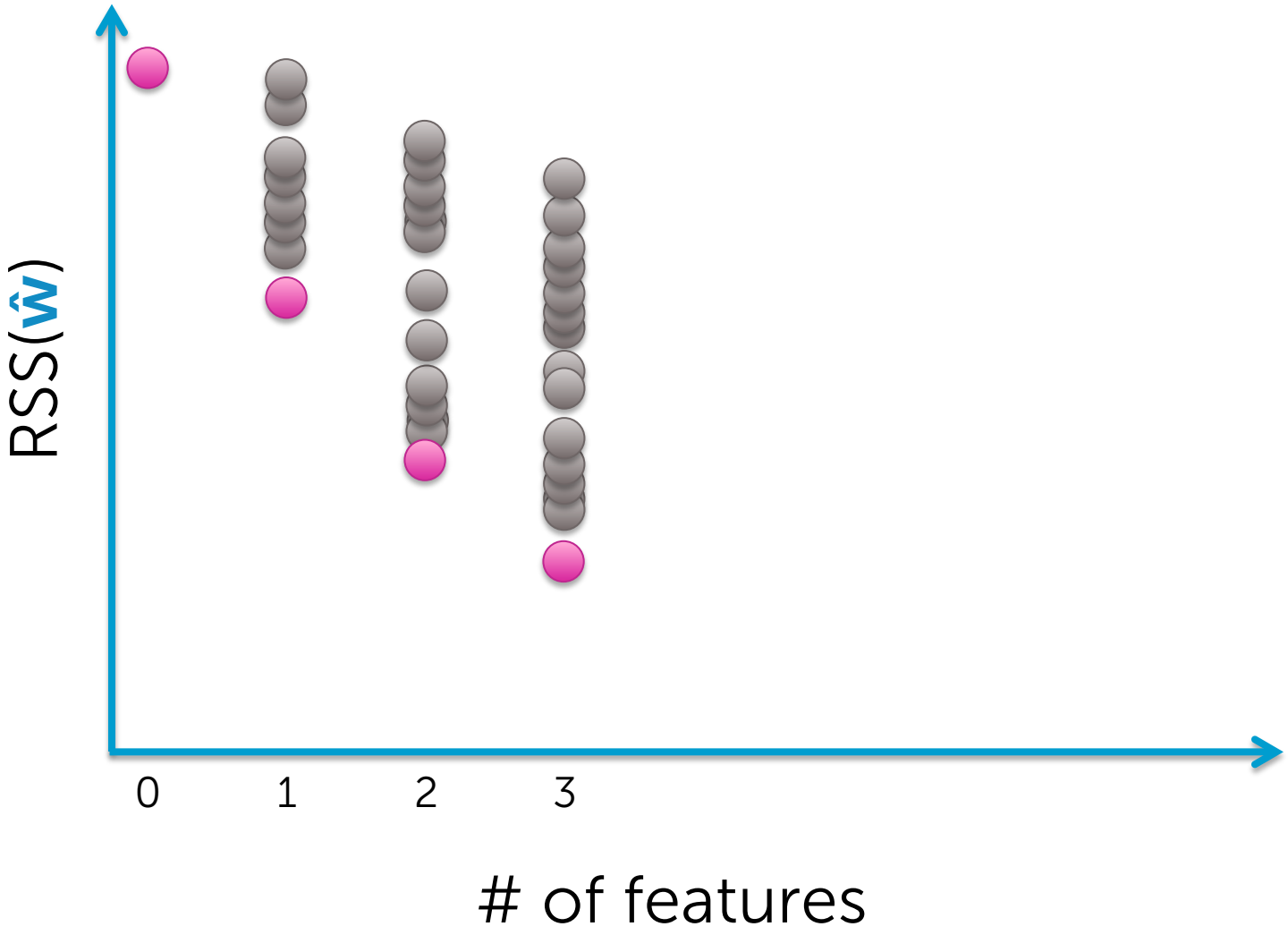
Note: Not necessarily nested!



Note: Not necessarily nested!

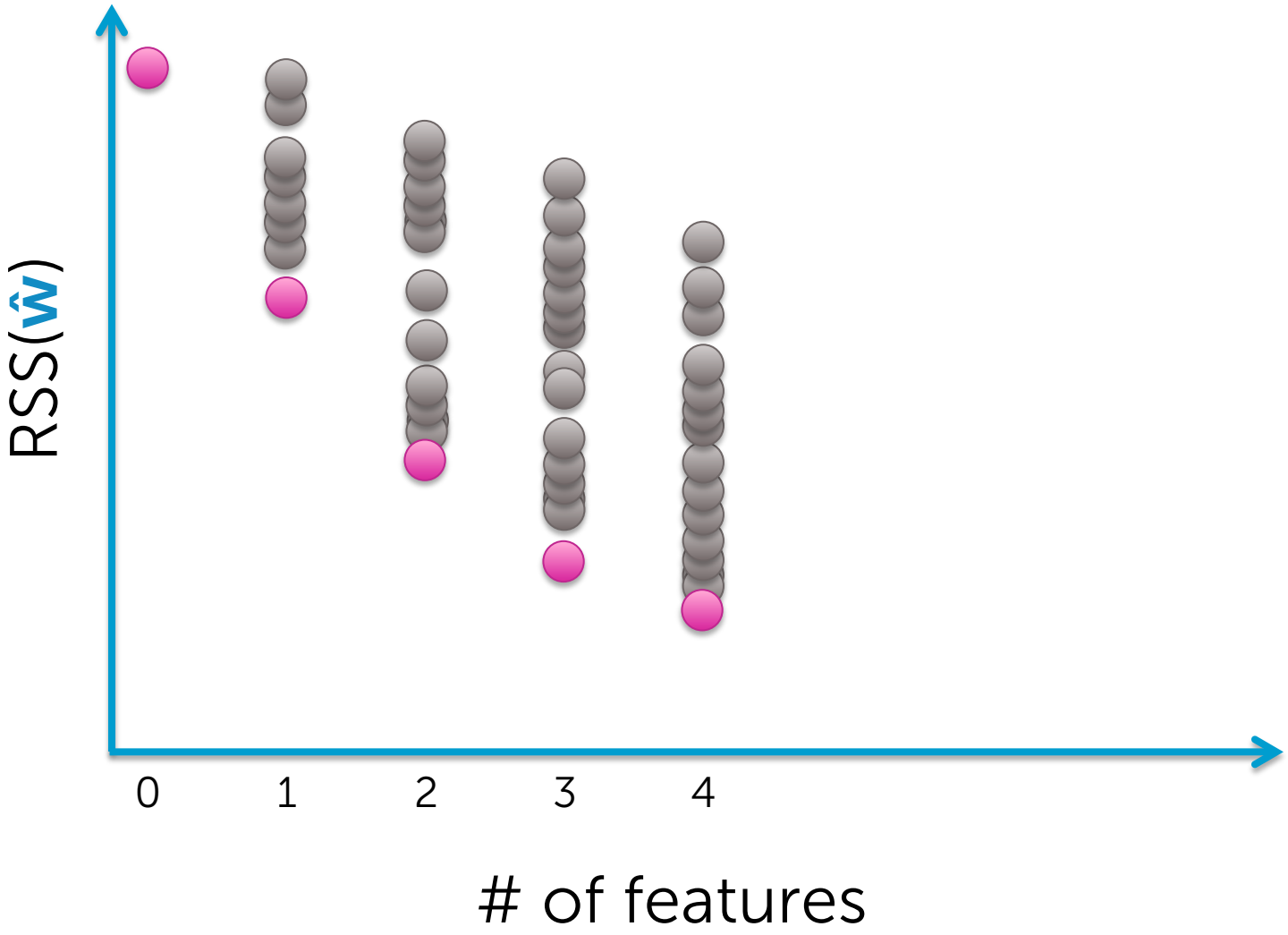


Find best model of size: 3



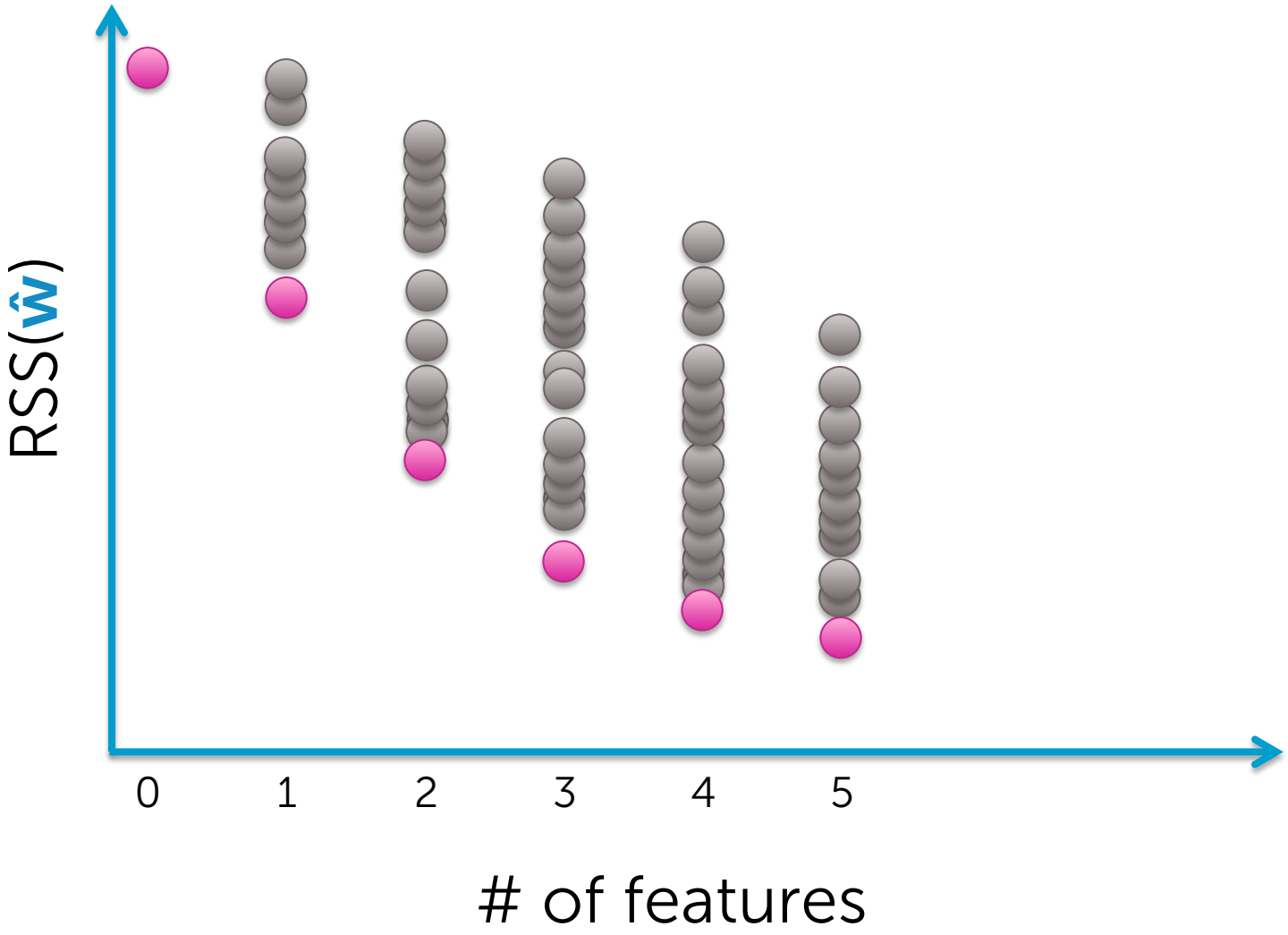
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 4



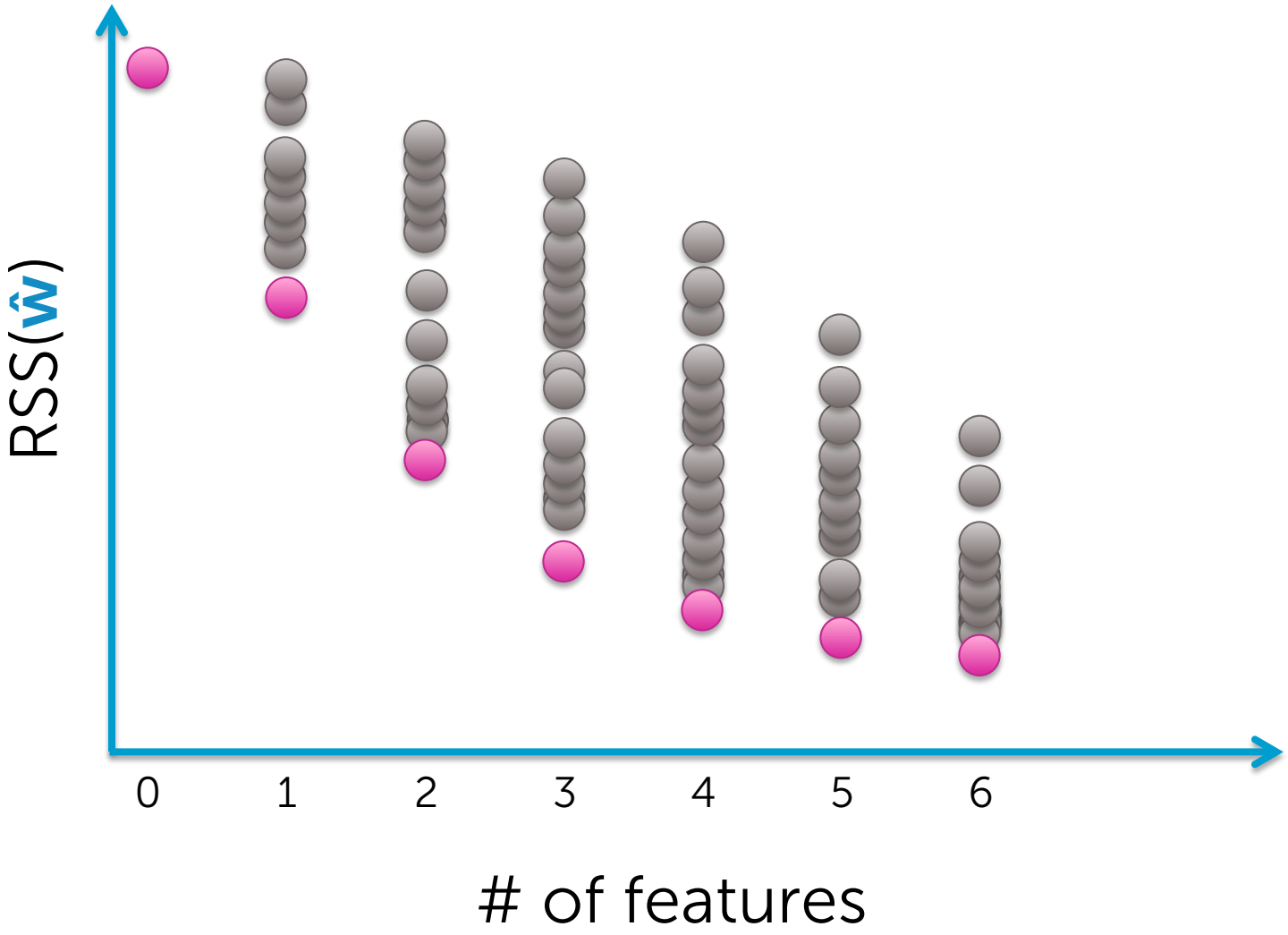
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 5



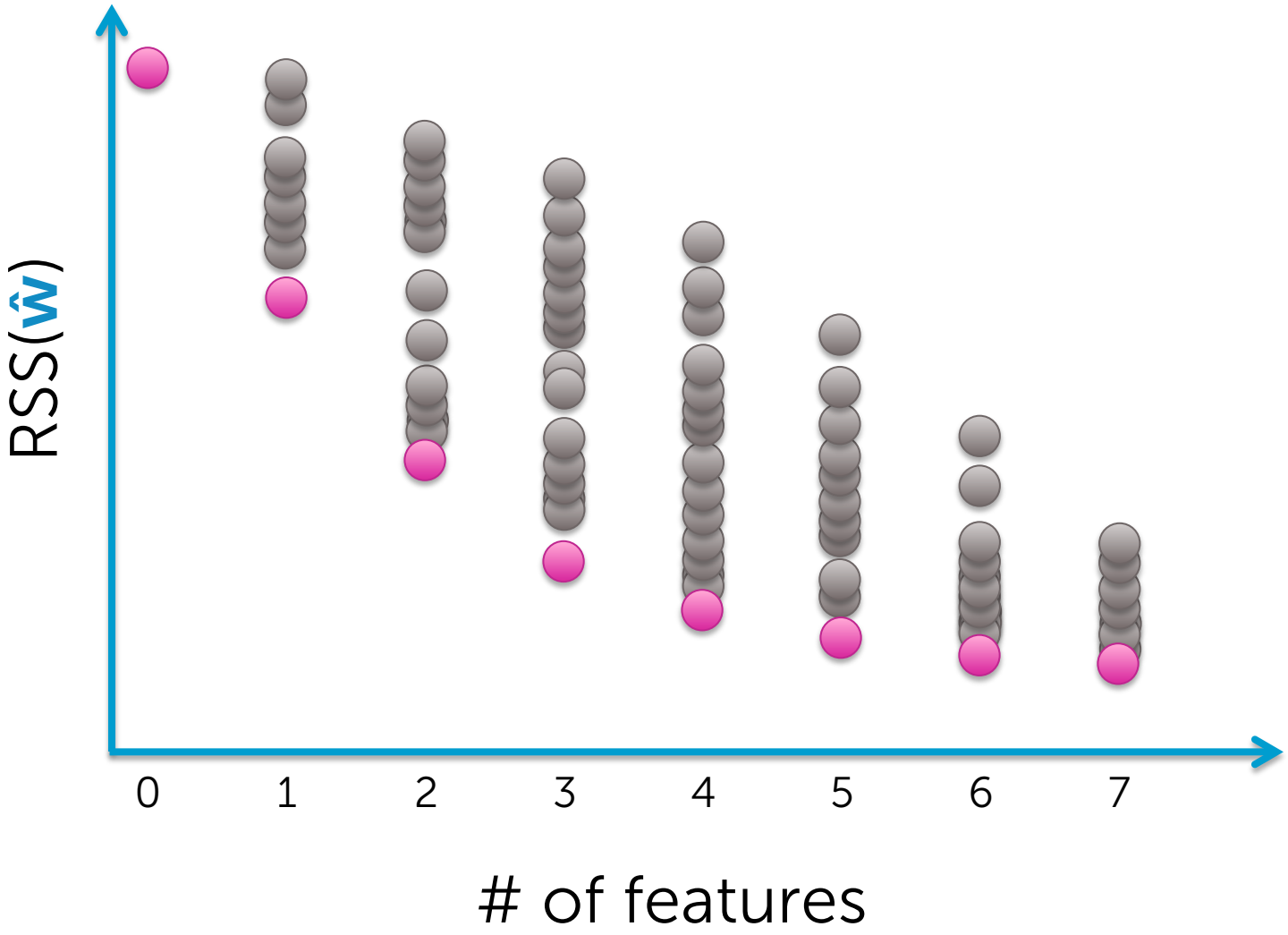
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 6



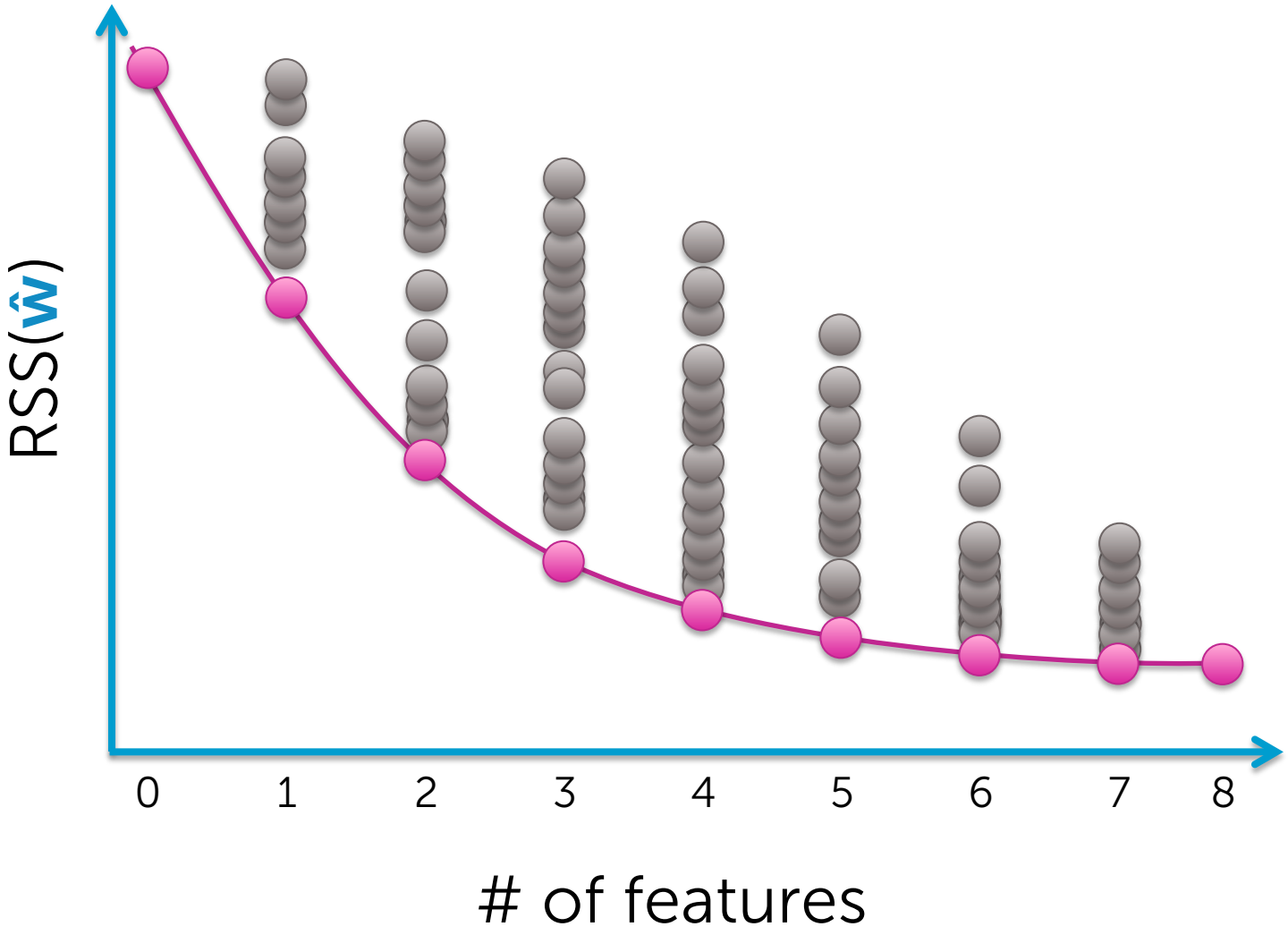
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 7



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

Find best model of size: 8



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

Choosing model complexity?

Option 1: Assess on validation set

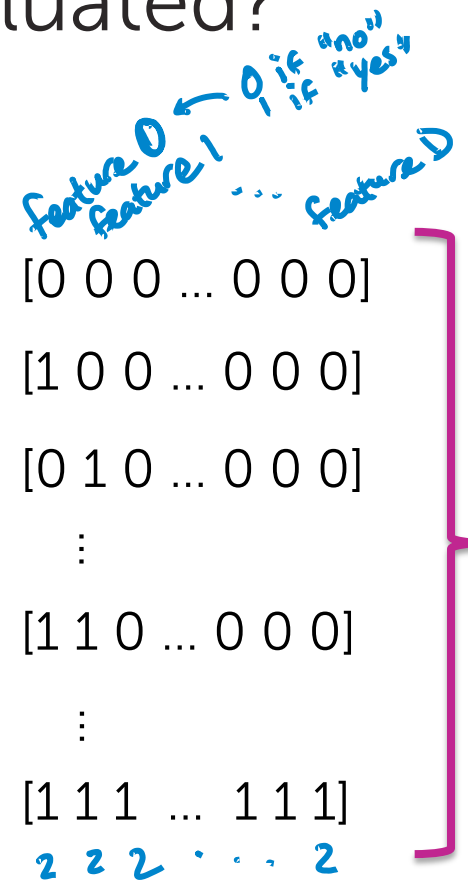
Option 2: Cross validation

Option 3+: Other metrics for penalizing model complexity like BIC…

Complexity of "all subsets"

How many models were evaluated?

– each indexed by features included



$y_i = \varepsilon_i$

$y_i = w_0 h_0(\mathbf{x}_i) + \varepsilon_i$

$y_i = w_1 h_1(\mathbf{x}_i) + \varepsilon_i$

⋮

$y_i = w_0 h_0(\mathbf{x}_i) + w_1 h_1(\mathbf{x}_i) + \varepsilon_i$

⋮

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

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

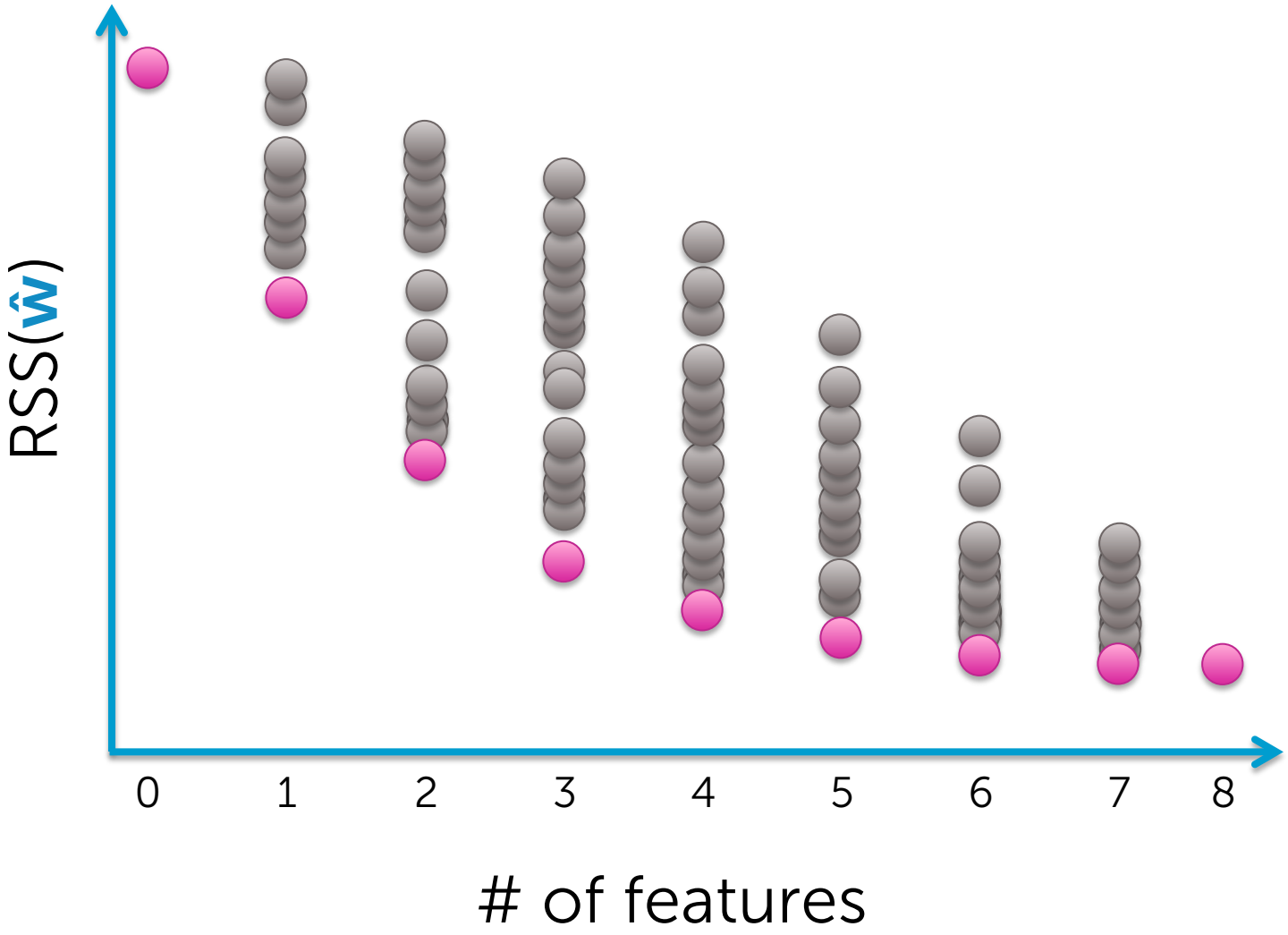
Option 2: Greedy algorithms

Forward stepwise 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 $h_0(\mathbf{x})=1$ → $y_i = w_0+\varepsilon_i$)
   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

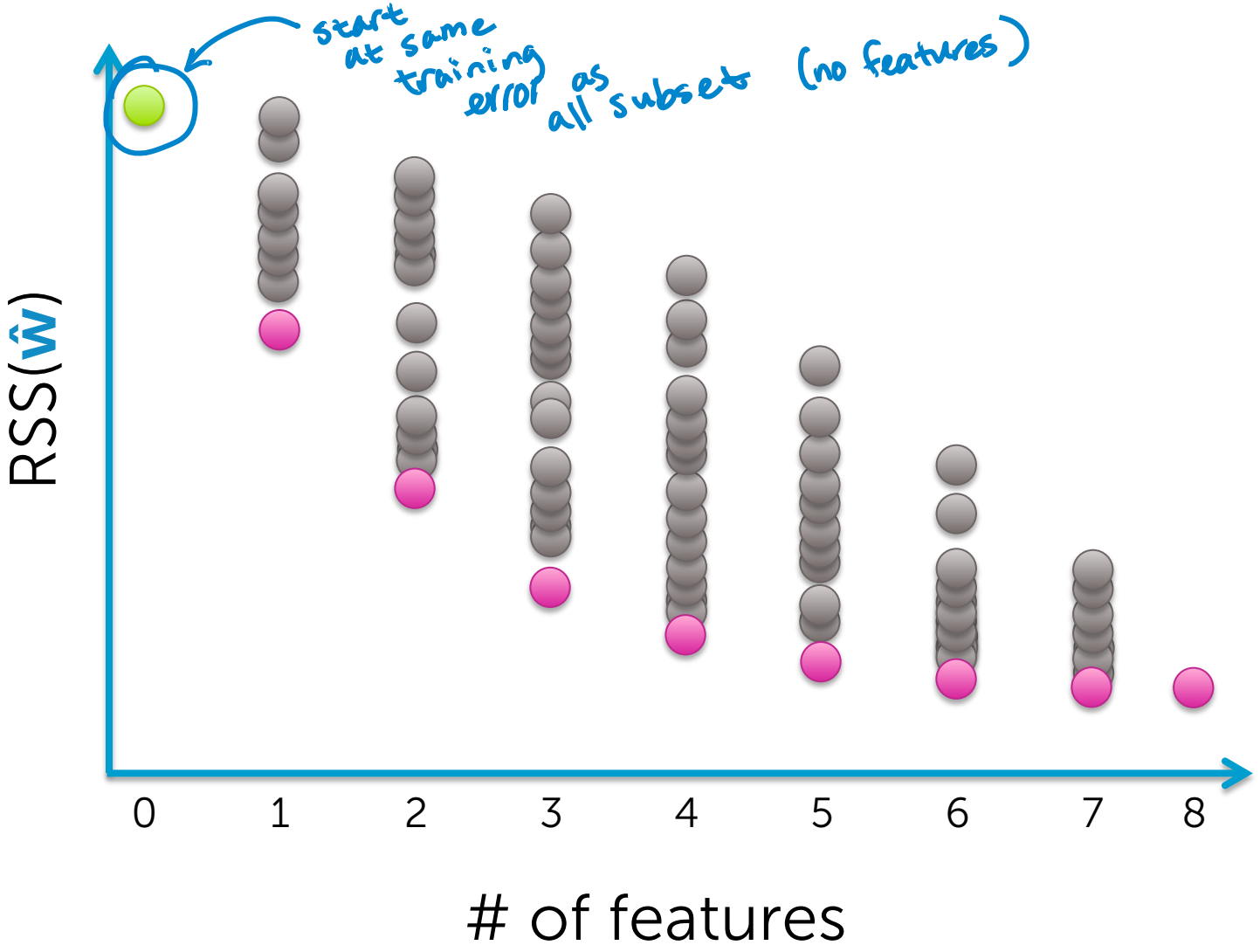
29

Visualizing greedy procedure



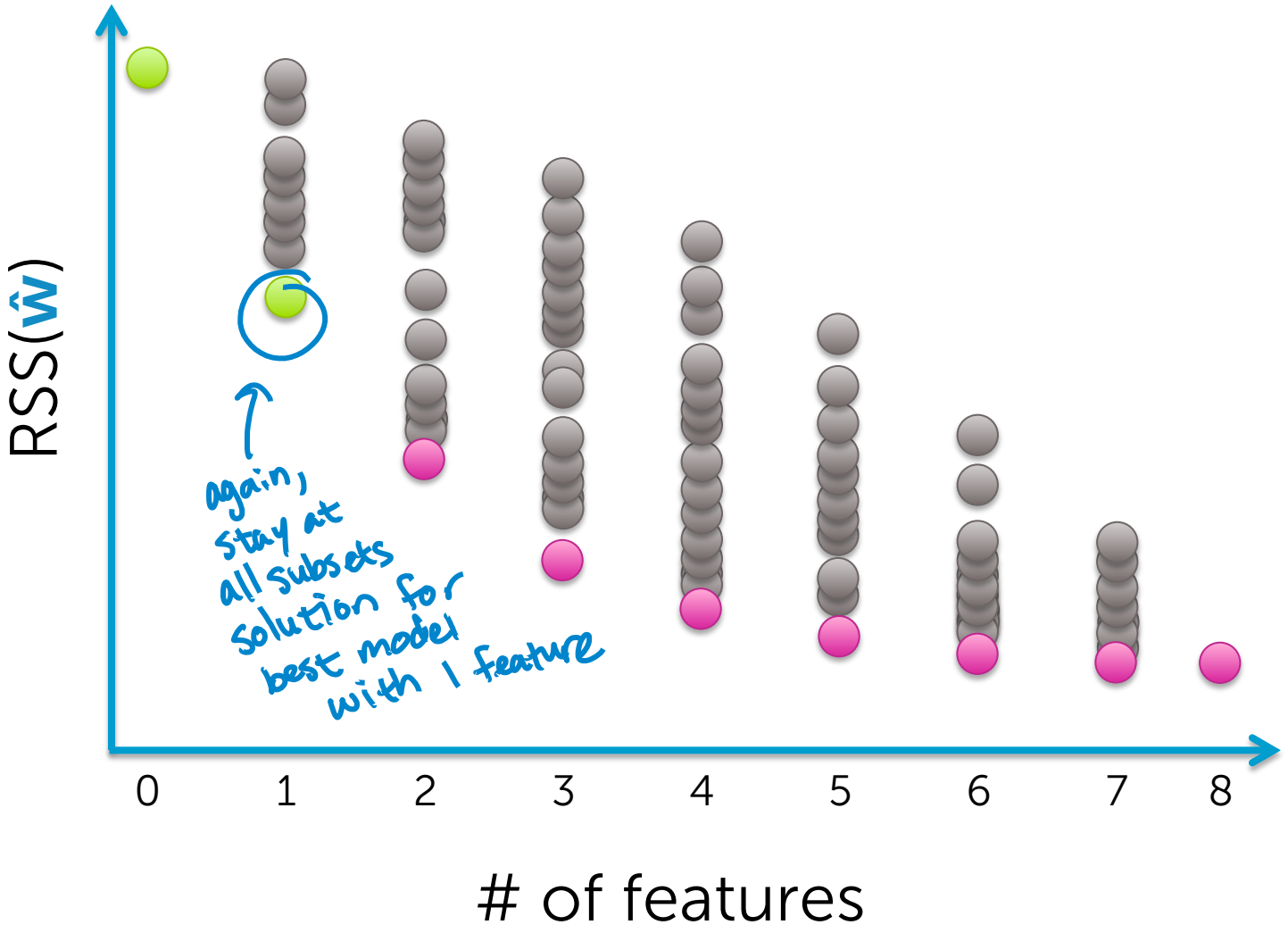
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Visualizing greedy procedure



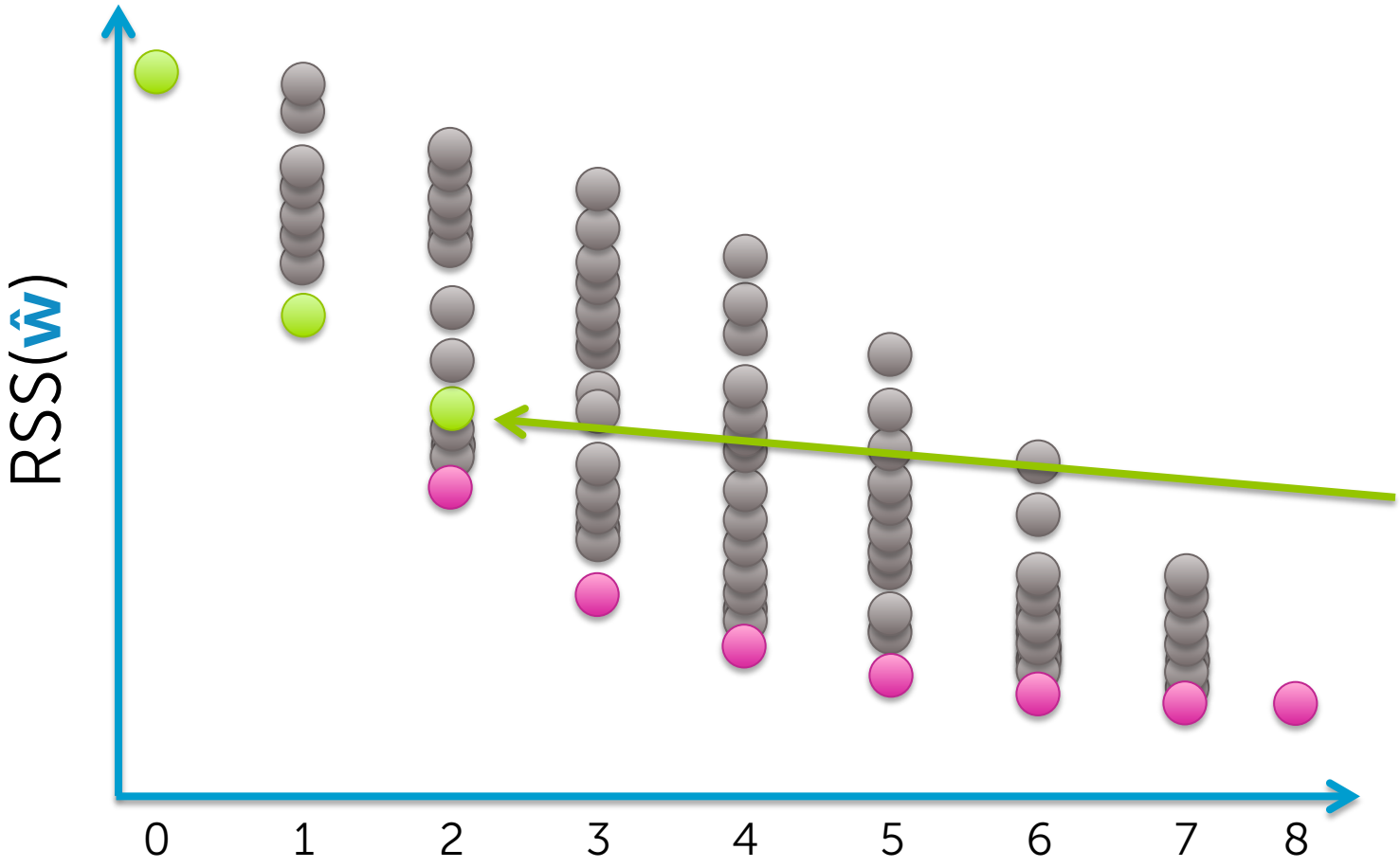
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Visualizing greedy procedure



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

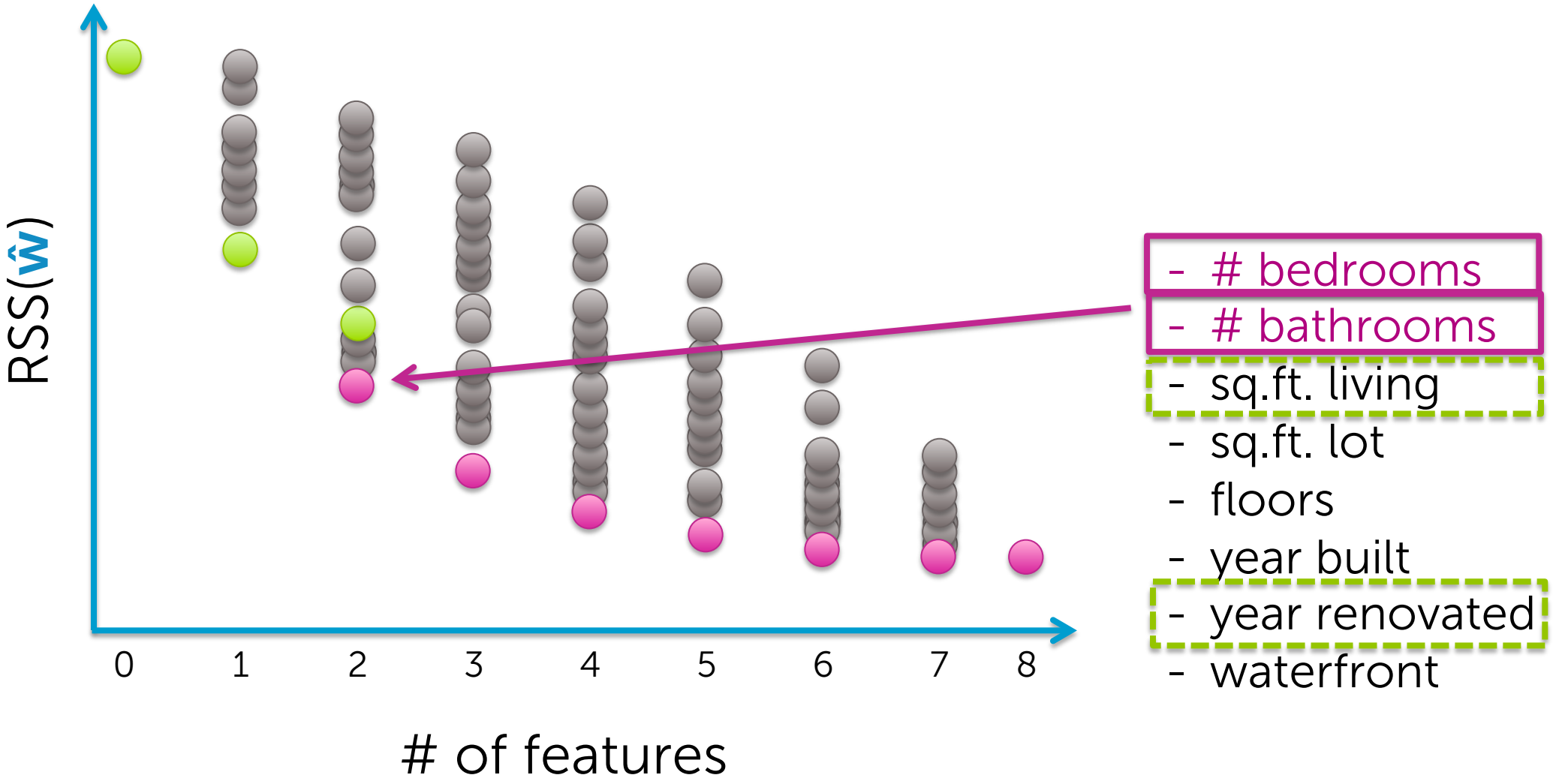
Visualizing greedy procedure



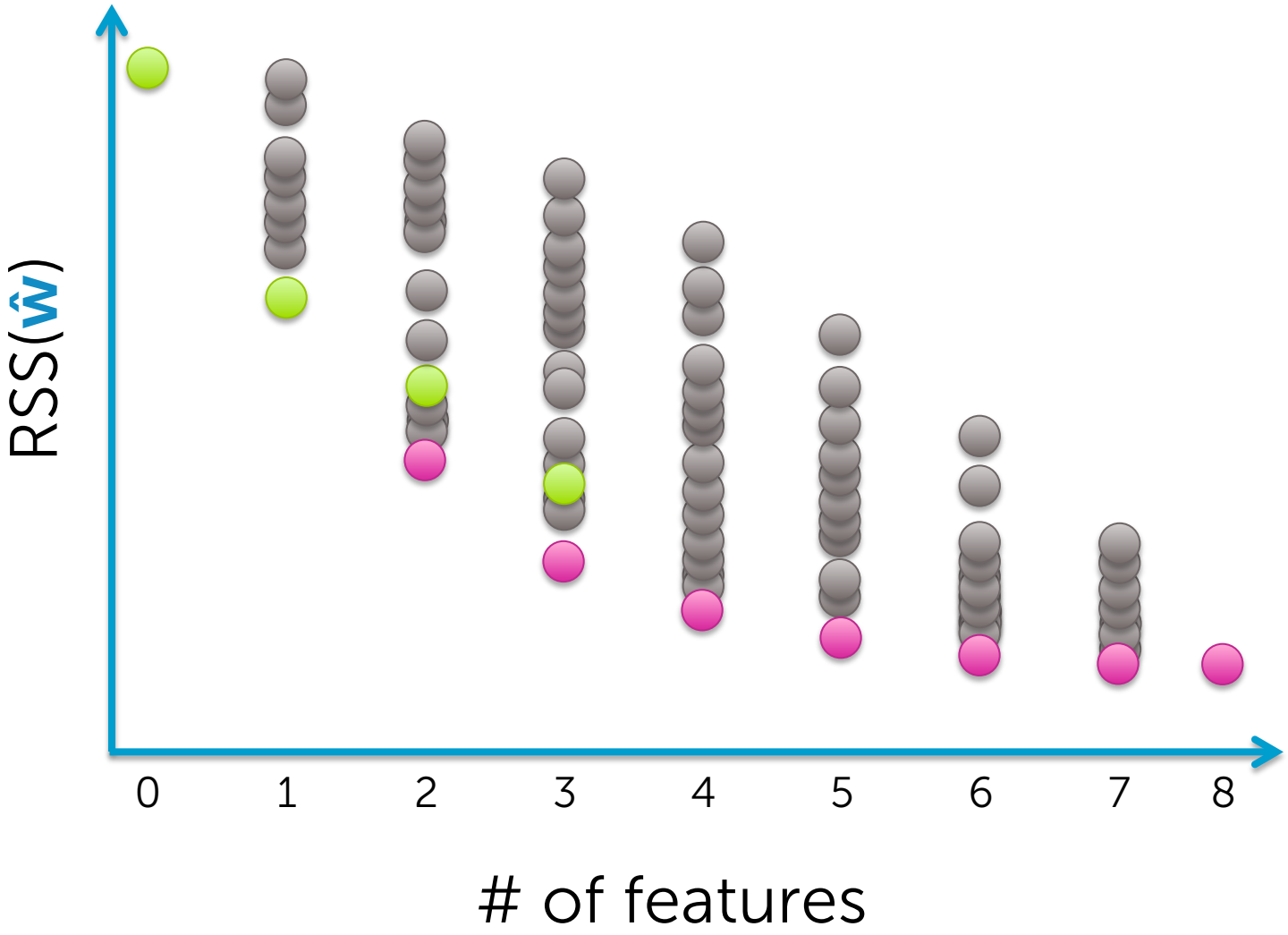
- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

forced to keep in model

Visualizing greedy procedure

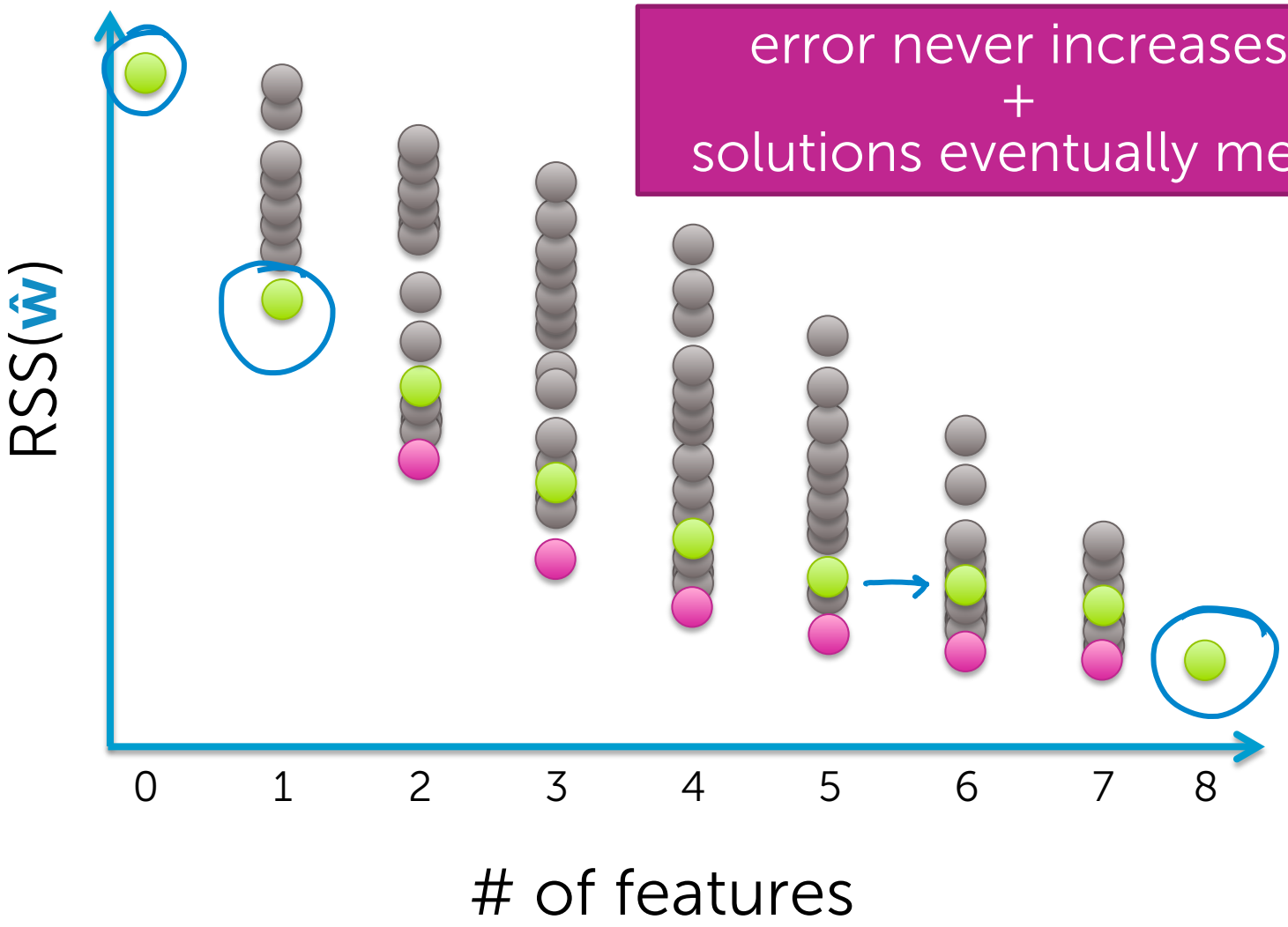


Visualizing greedy procedure



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

Visualizing greedy procedure



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

When do we stop?

When **training error** is low enough?

No!

When **test error** is low enough?

No!

Use validation set or cross validation!

37

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 1 feature out of D-2 possible)
- …

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

Lots of other variants, too.

39

Option 3: Regularize

Ridge regression:
L_2 regularized regression

Total cost =

measure of fit + λ measure of magnitude of coefficients

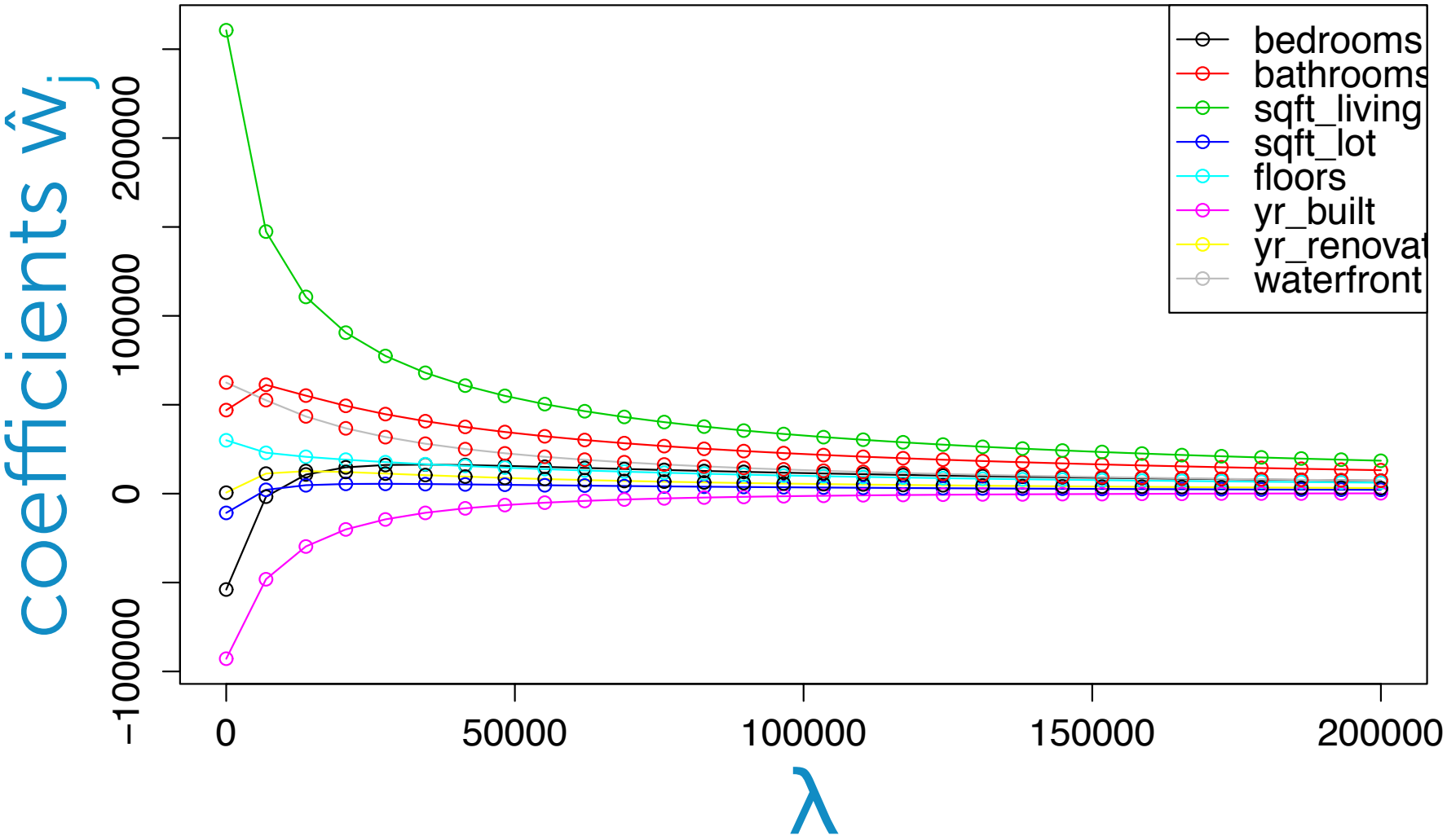
RSS($\mathbf{w}$)

$$||\mathbf{w}||_2^2 = w_0^2 + \ldots + w_D^2$$

Encourages **small weights**

but not exactly 0

Coefficient path – **ridge**



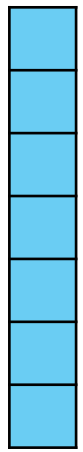
Recall sparsity (many $\hat{w}_j$=0) gives efficiency and interpretability

Efficiency:

- If size(**w**) = 100B, each prediction is expensive
- If **ŵ** sparse, computation only depends on # of non-zeros

many zeros

$$\hat{y}_i = \sum_{\hat{w}_j \neq 0} \hat{w}_j \, h_j(\mathbf{x}_i)$$



Interpretability:

- Which features are relevant for prediction?

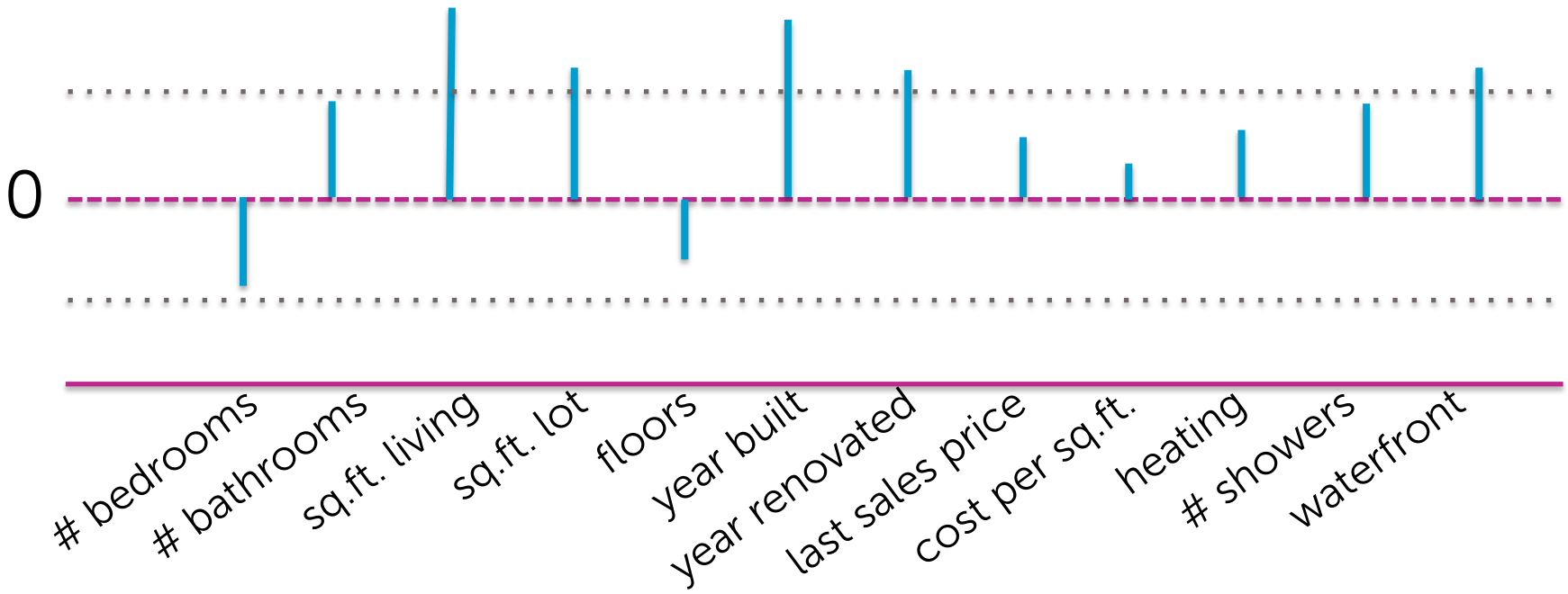
Using regularization for feature selection

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

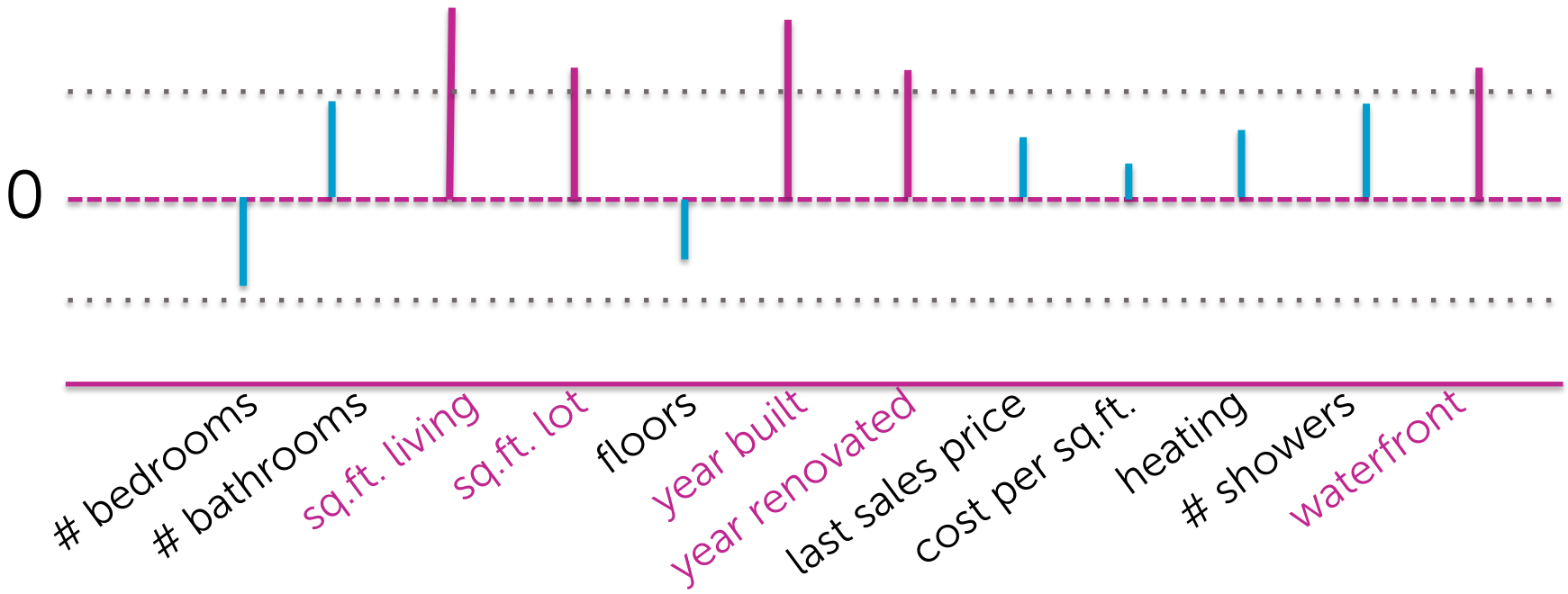
Thresholding ridge coefficients?

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



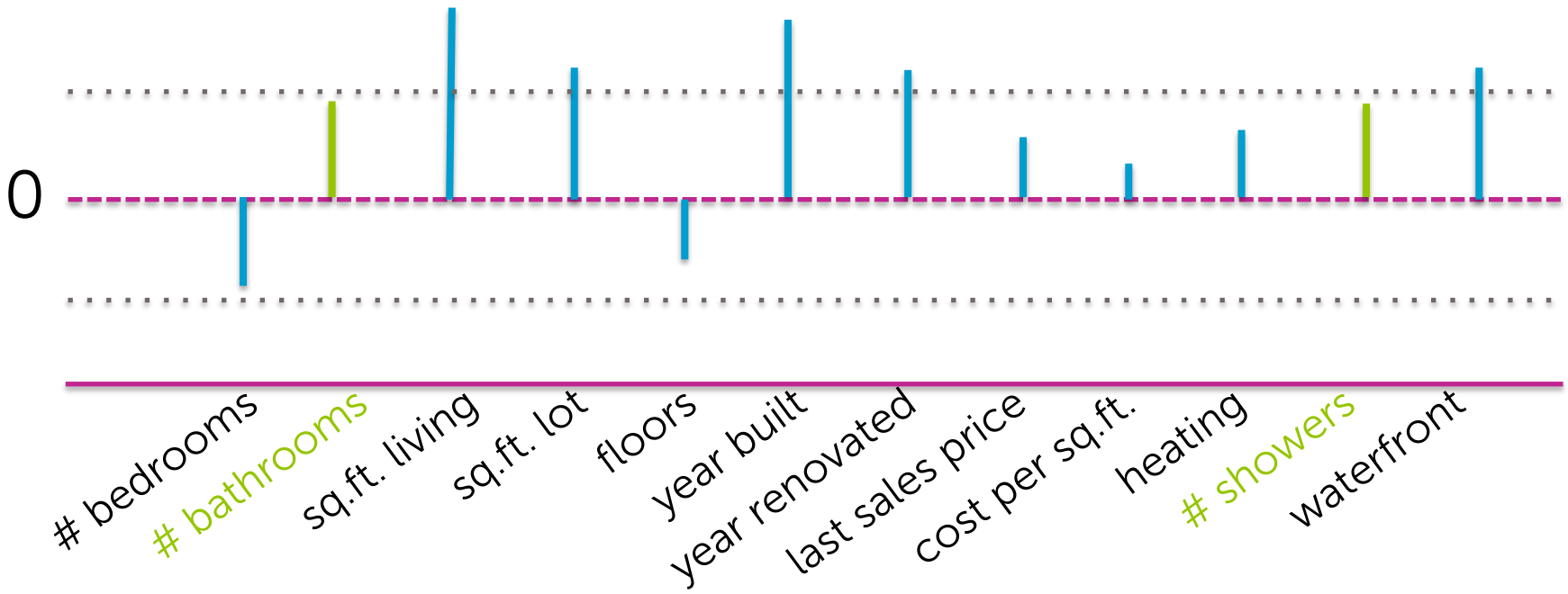
Thresholding ridge coefficients?

Selected features for a given threshold value



Thresholding ridge coefficients?

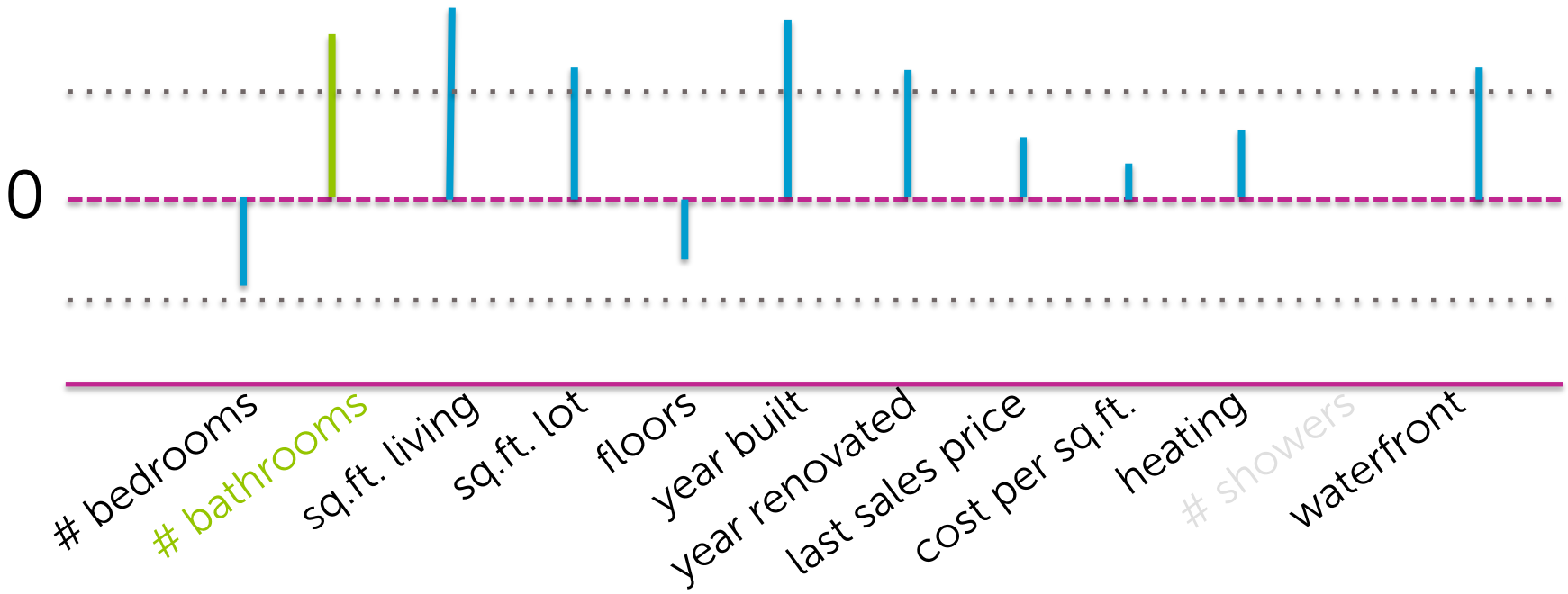
Let's look at two related features…



Nothing measuring bathrooms was included!

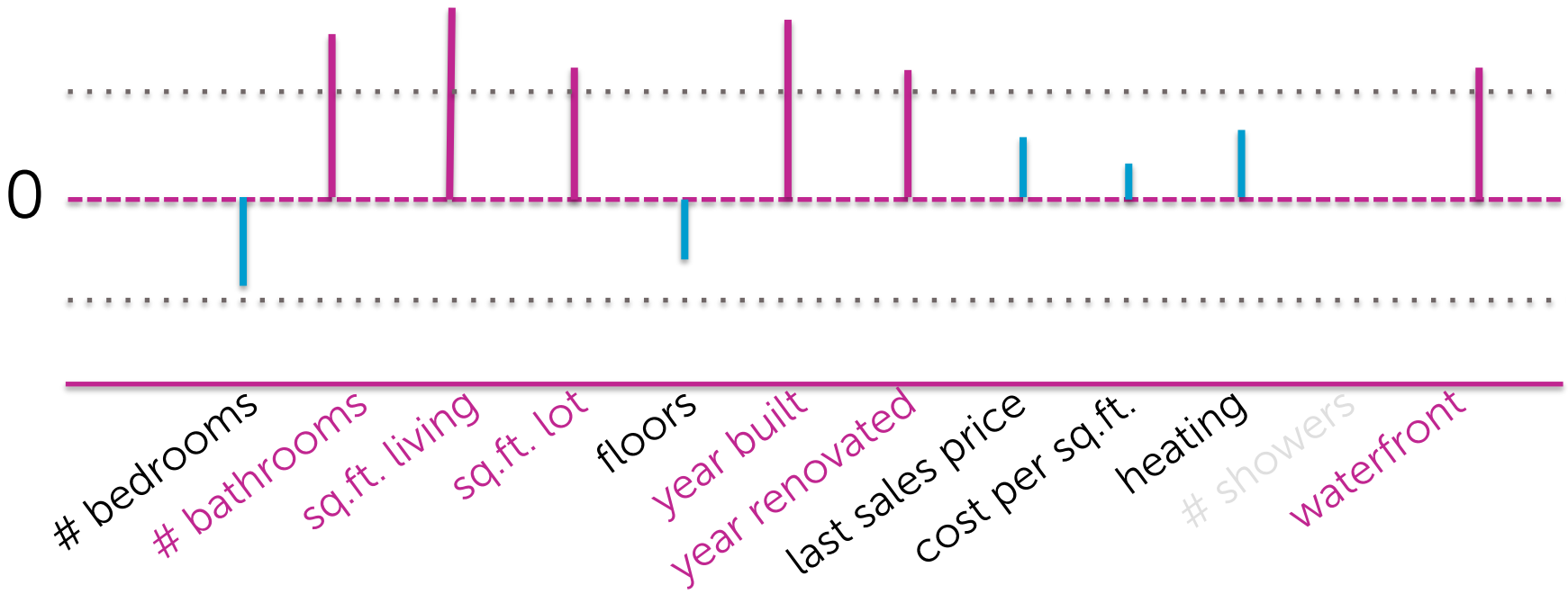
Thresholding ridge coefficients?

If only one of the features had been included…



Thresholding ridge coefficients?

Would have included bathrooms in selected model



Can regularization lead directly to sparsity?

Try this cost instead of ridge…

Total cost =

measure of fit + λ measure of magnitude of coefficients

RSS($\mathbf{w}$)

$||\mathbf{w}||_1 = |w_0| + \ldots + |w_D|$

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

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

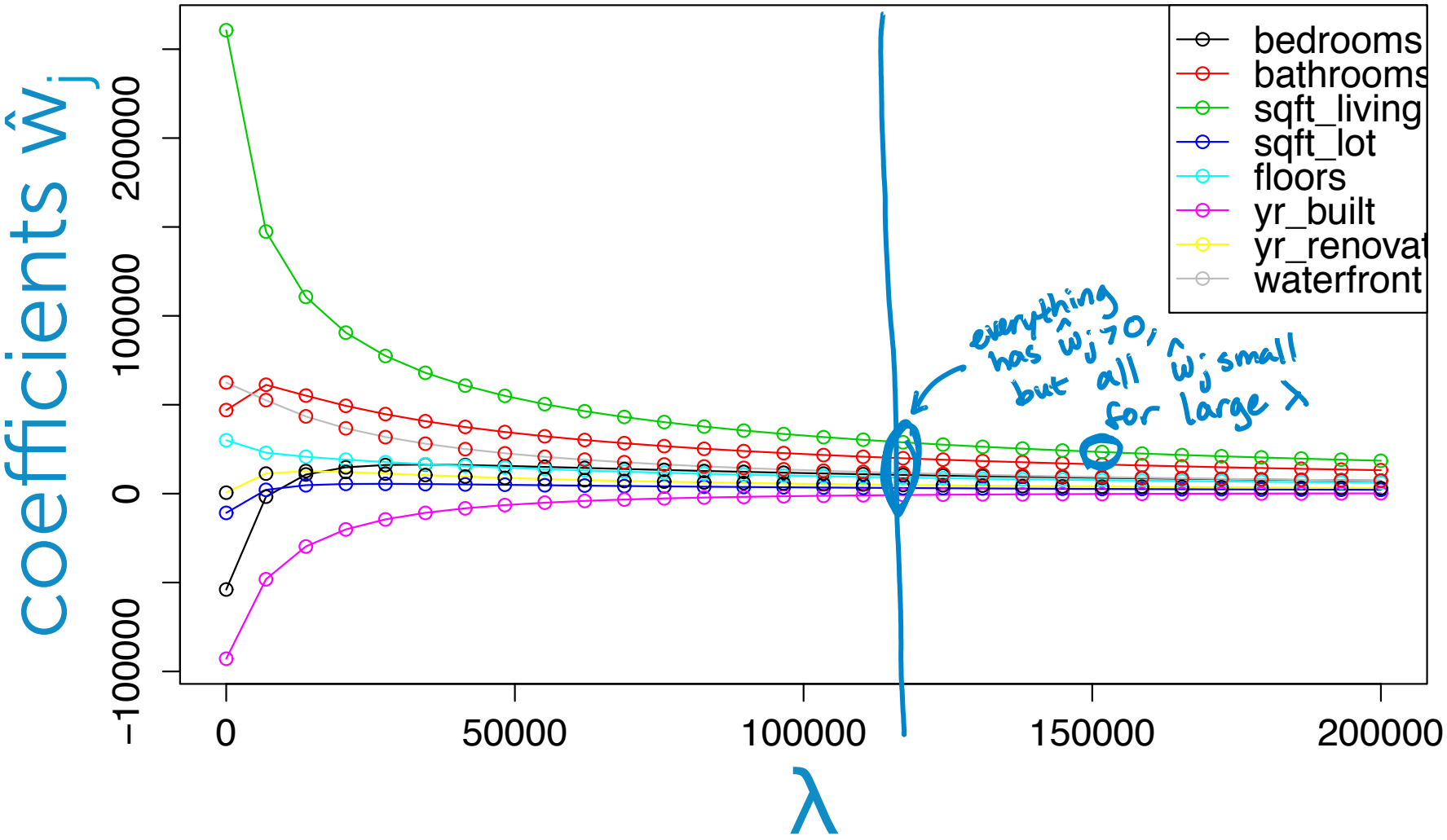
tuning parameter = balance of fit and **sparsity**

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

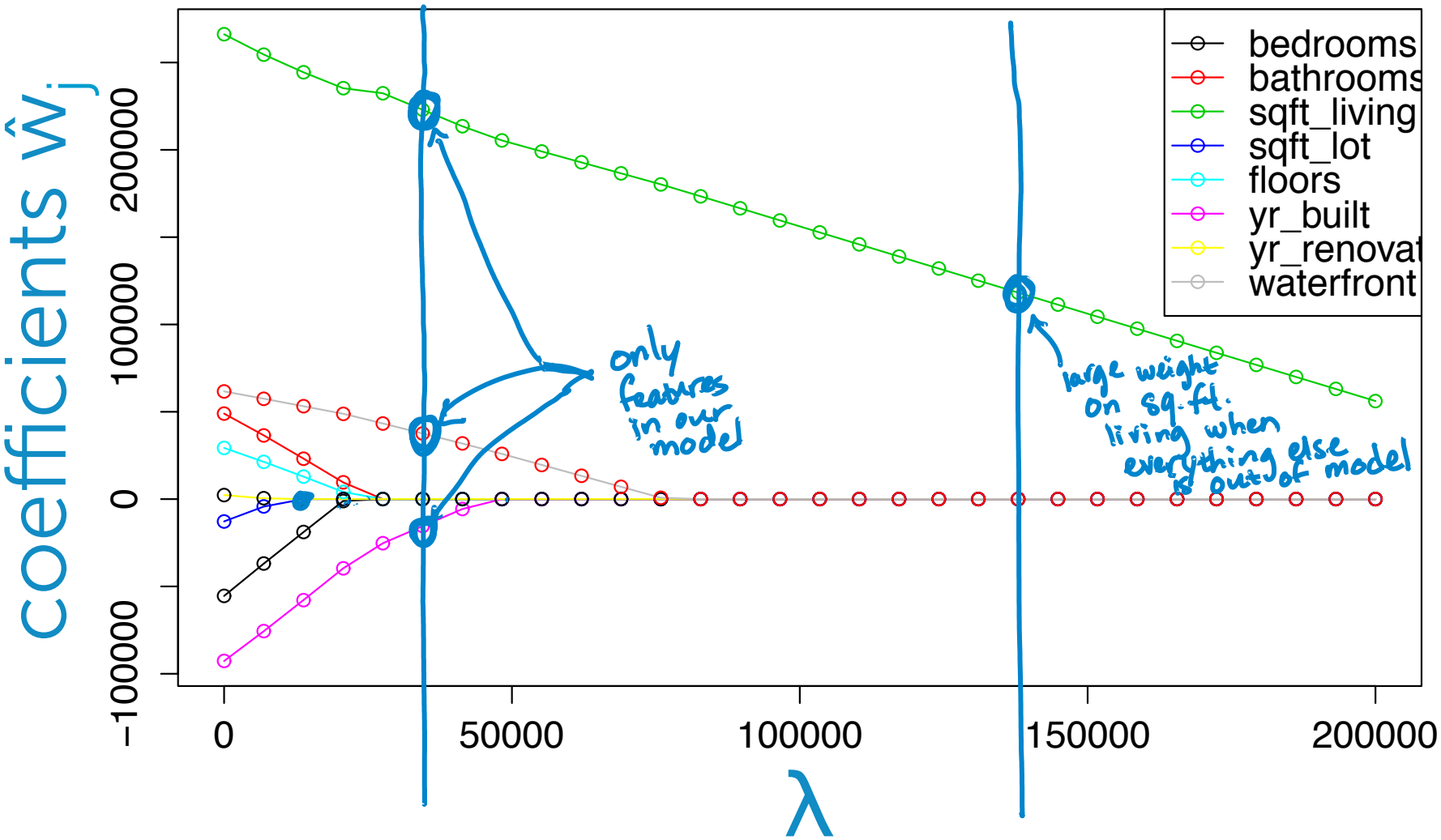
If $\lambda=\infty$: $\hat{w}^{lasso} = 0$

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

Coefficient path – **ridge**



Coefficient path – lasso

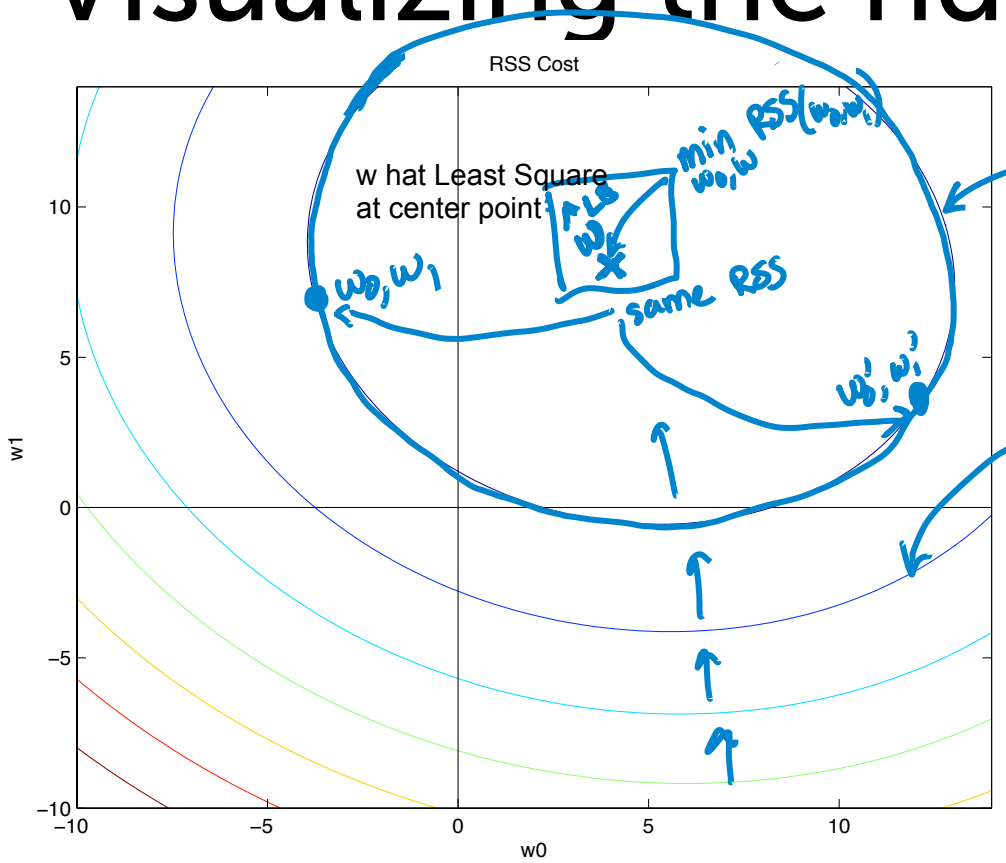


eventually sq.ft.living will be shrunken to exactly 0

Geometric intuition for sparsity of lasso solution

Geometric intuition for ridge regression

Visualizing the ridge cost in 2D



$RSS(w_0, w_1) = const_1$

$RSS(w_0, w_1) = const_2 > const_1$

Expand (y - w0h0 - w1h1)^2, get the following expression

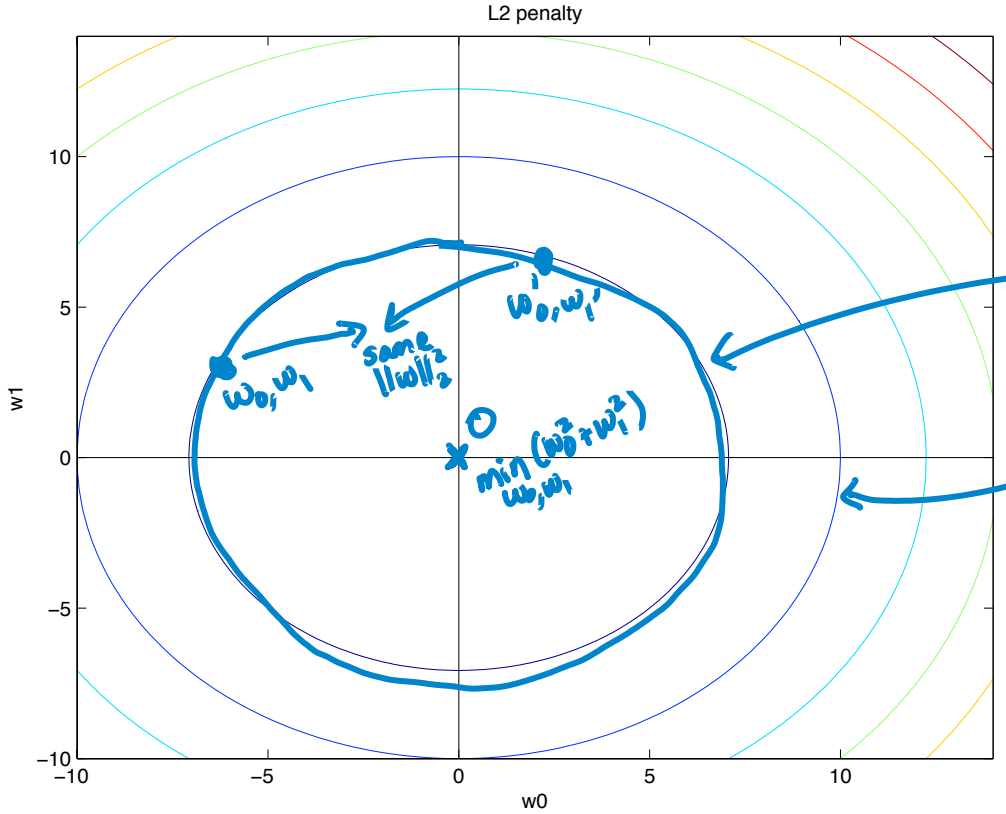
$\sum_i y^2 + w_0^2 \sum_i h_0^2 + w_1^2 \sum_i h_1^2 + \text{cross terms} = \text{constant}$

ellipse

2 features for visualization sake

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

Visualizing the ridge cost in 2D



$\|w\|_2^2 = \text{const}_1$

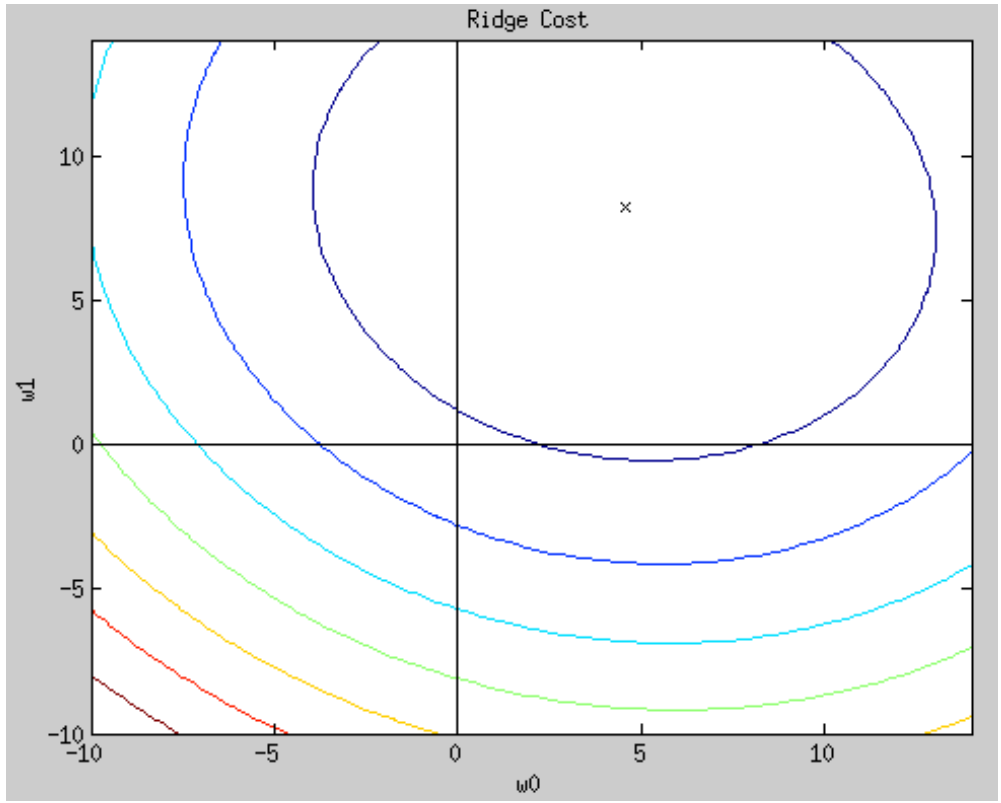
$\|w\|_2^2 = \text{const}_2 > \text{const}_1$

$w_0^2 + w_1^2 = \text{constant}$

circle

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

Visualizing the ridge cost in 2D



Add contour plots together (RSS plot and L2 norm plot)

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

ellipses ↑ circles

weighting

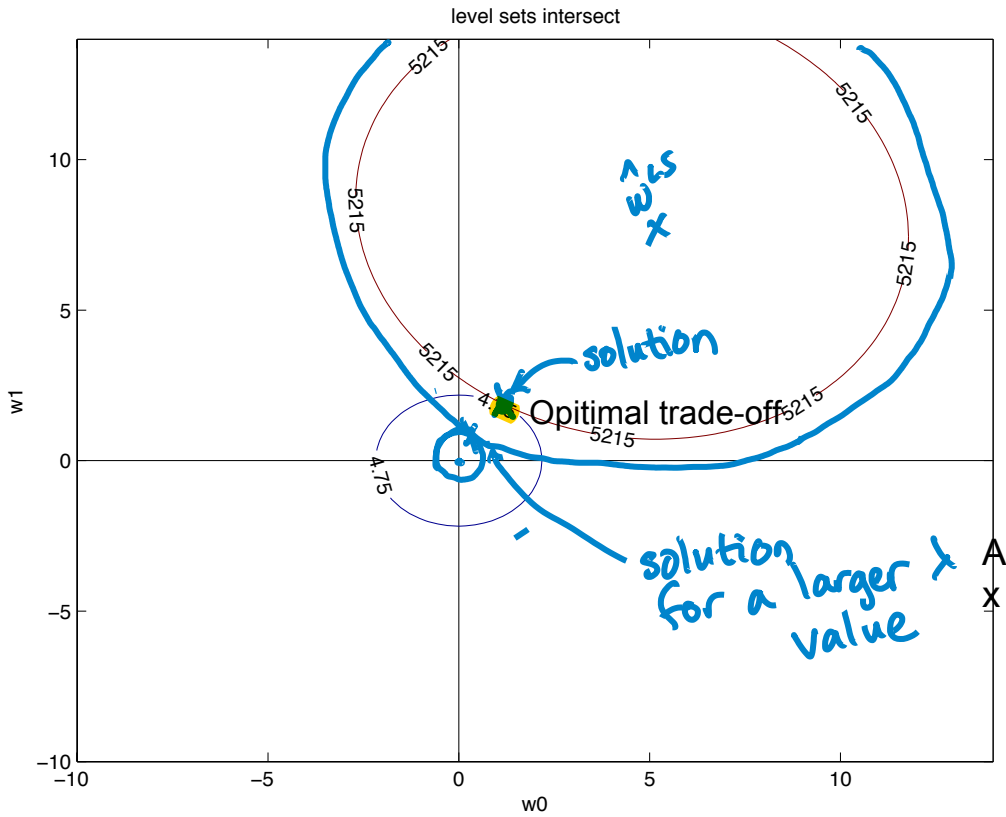
Movie: function of increasing λ

X mark optimal solution for a specific λ

When lambda=0, x starts at least squares solution.
As lambda -> infinity, coefficients shrink to 0.

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

Visualizing the ridge solution



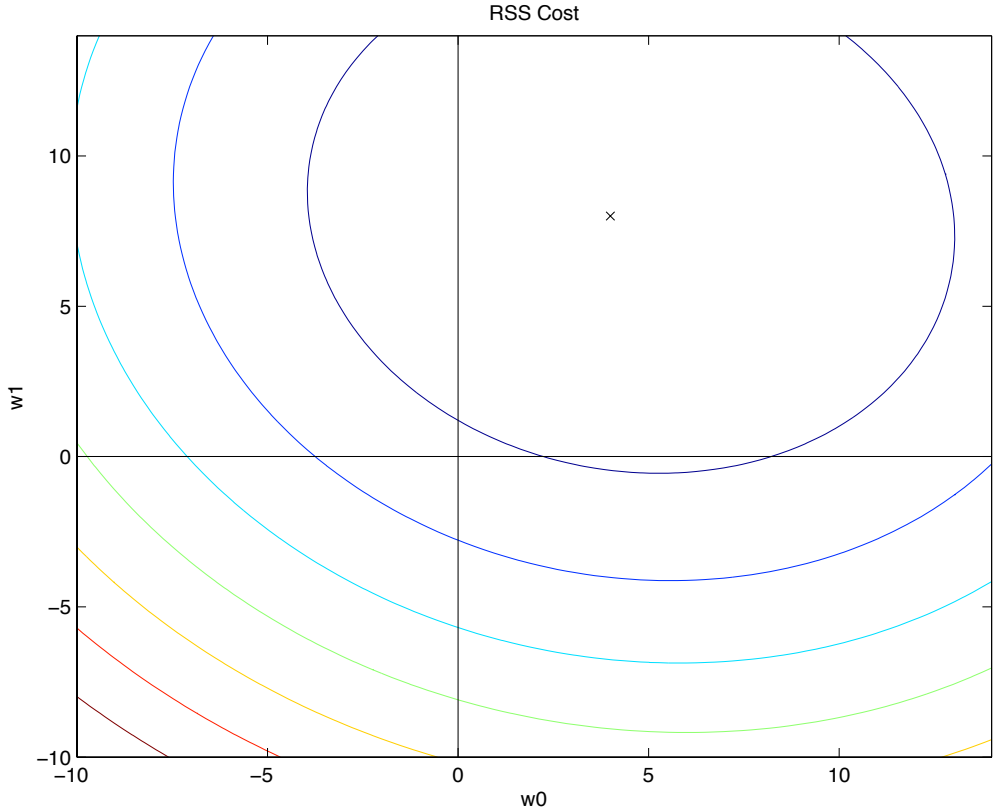
For a specific λ value, some balance between RSS and $\|w\|_2^2$

As lambda -> infinity, x is closer to the origin

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

Geometric intuition for lasso

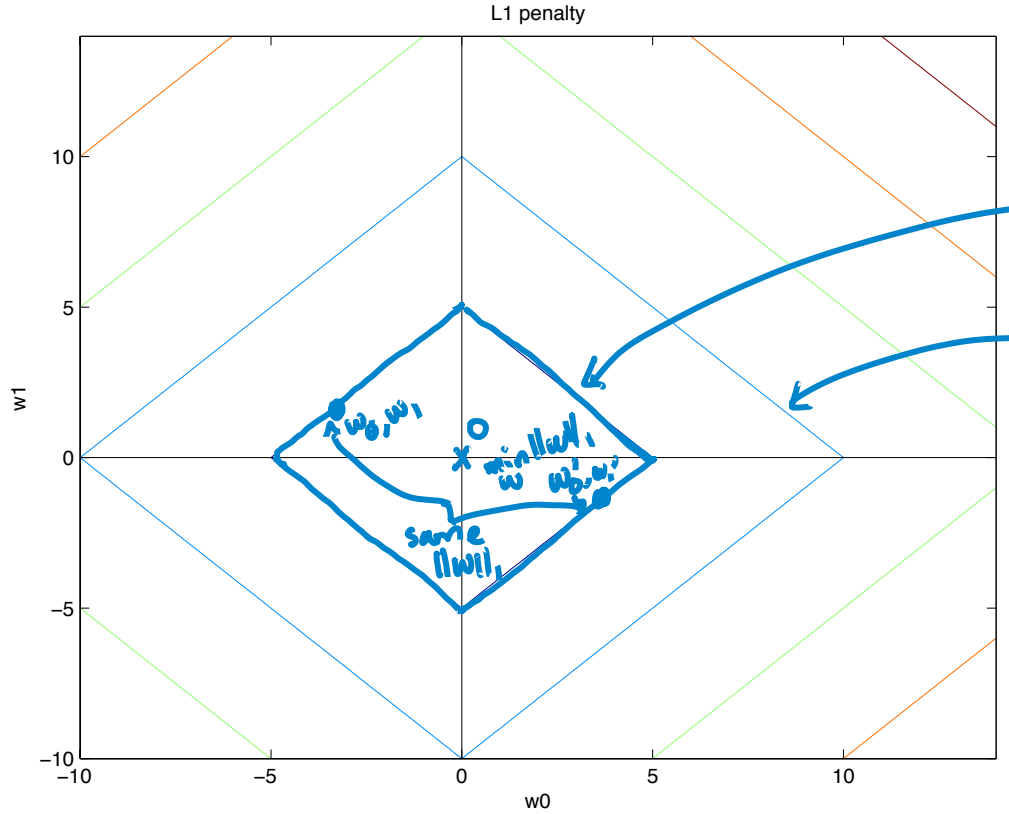
Visualizing the lasso cost in 2D



RSS contours for lasso are exactly the same as those for ridge!

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

Visualizing the lasso cost in 2D



$||w||_1 = const_1$

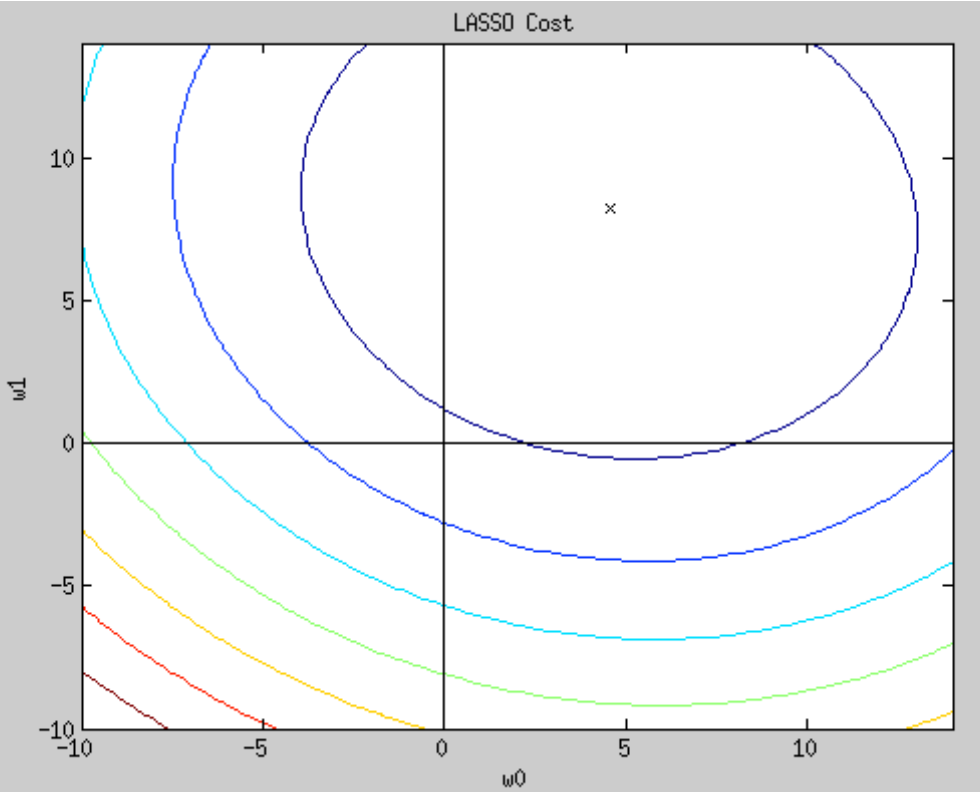
$||w||_1 = const_2 > const_1$

$|w_0| + |w_1| = constant$

diamond

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

Visualizing the lasso cost in 2D



Adding

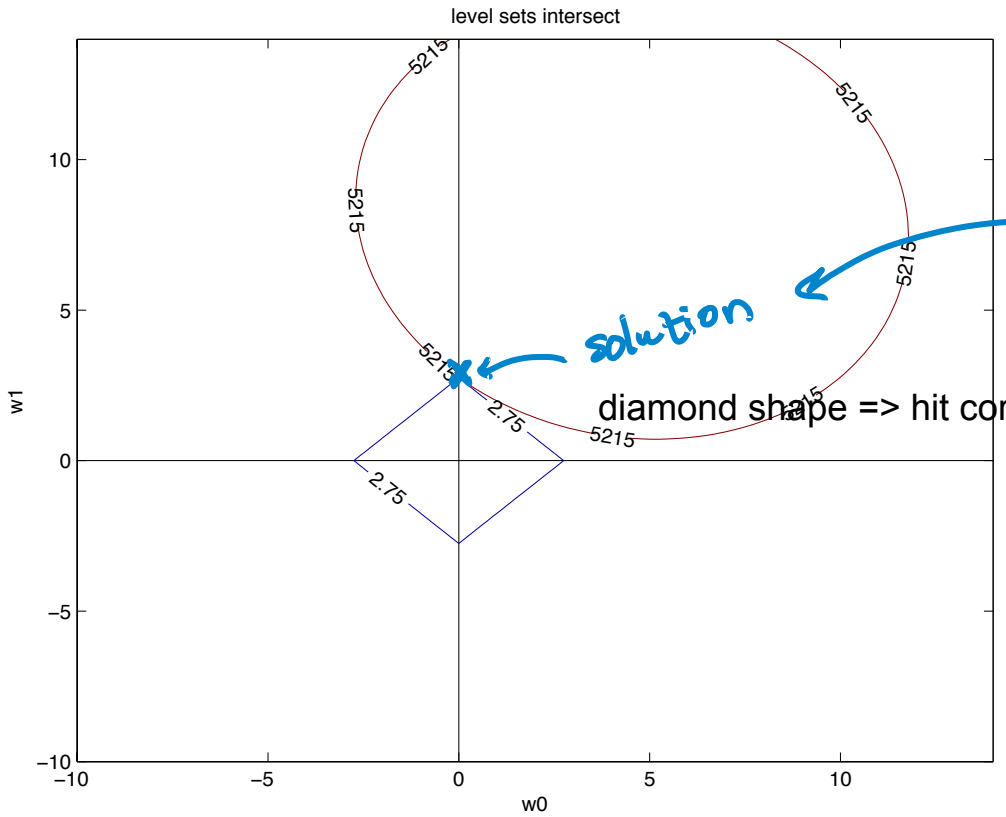
RSS + $\lambda \|w\|_1$

ellipses diamonds

$\times$ = optimal $\hat{w}$ for a specific λ

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

Visualizing the lasso solution



For a specific value of λ,

solution

diamond shape => hit corners => spare solutions

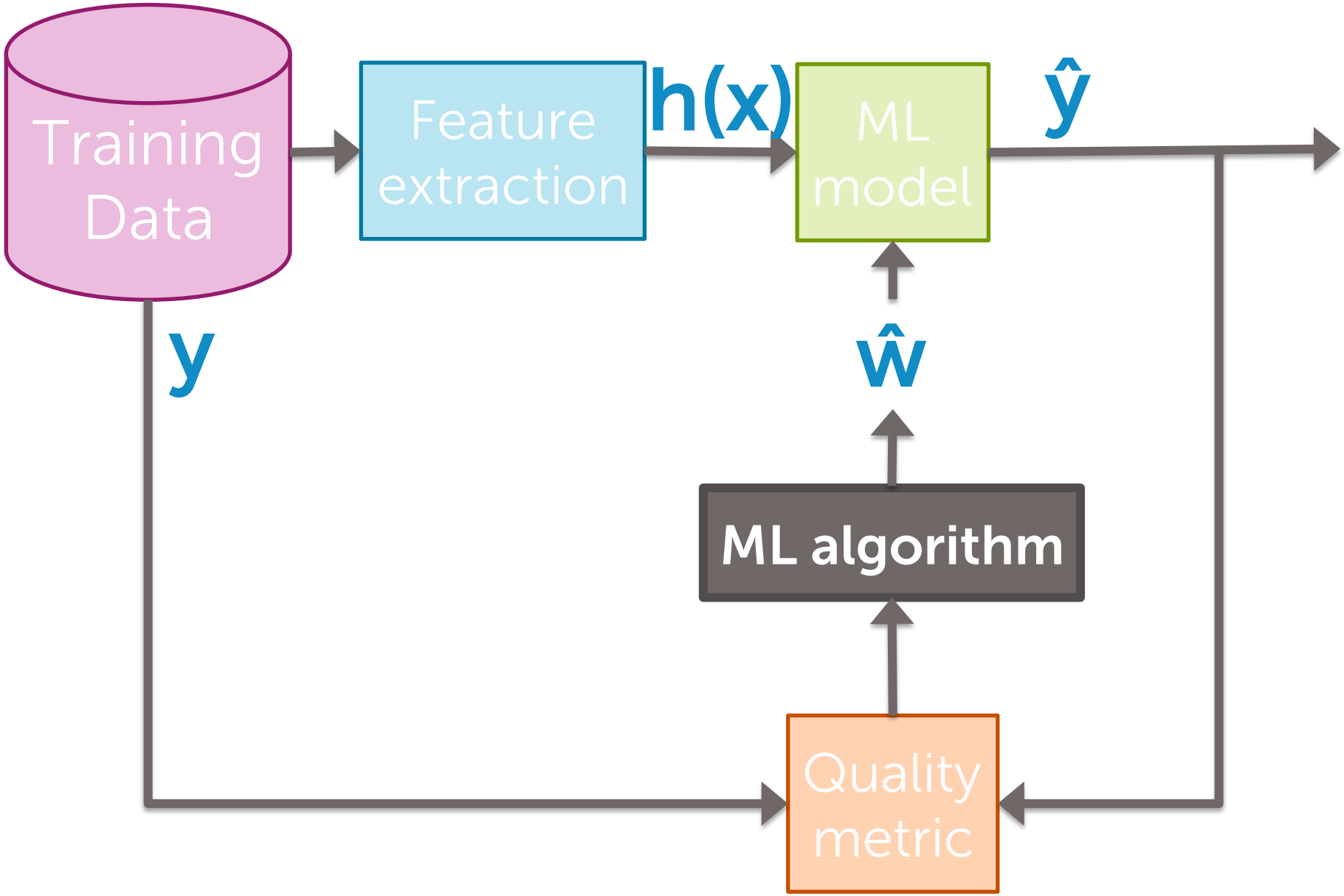
$$\text{RSS}(\mathbf{w}) + \lambda||\mathbf{w}||_1 = \sum_{i=1}^{N}(y_i - w_0 h_0(\mathbf{x}_i) - w_1 h_1(\mathbf{x}_i))^2 + \lambda\,(|w_0| + |w_1|)$$

Revisit polynomial fit demo

What happens if we refit our high-order polynomial, but now using **lasso regression**?

Will consider a few settings of λ …

Fitting the lasso regression model (for given λ value)



How we optimized past objectives

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

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

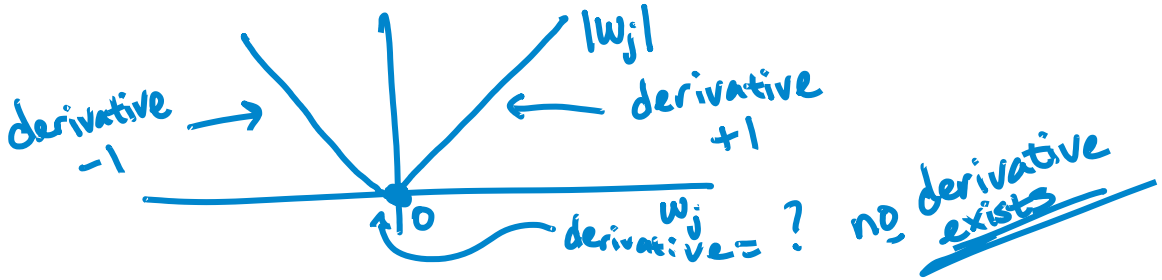
Optimizing the lasso objective

Lasso total cost: $\text{RSS}(\mathbf{w}) + \lambda ||\mathbf{w}||_1$

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

Issues:

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



gradients → subgradients

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

can use subgradient descent

Aside 1: Coordinate descent

Coordinate descent

Goal: Minimize some function g

$\min_{w} g(w)$

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

Often, hard to find minimum for all coordinates, but easy for each coordinate

when keeping others fixed

Coordinate descent:

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

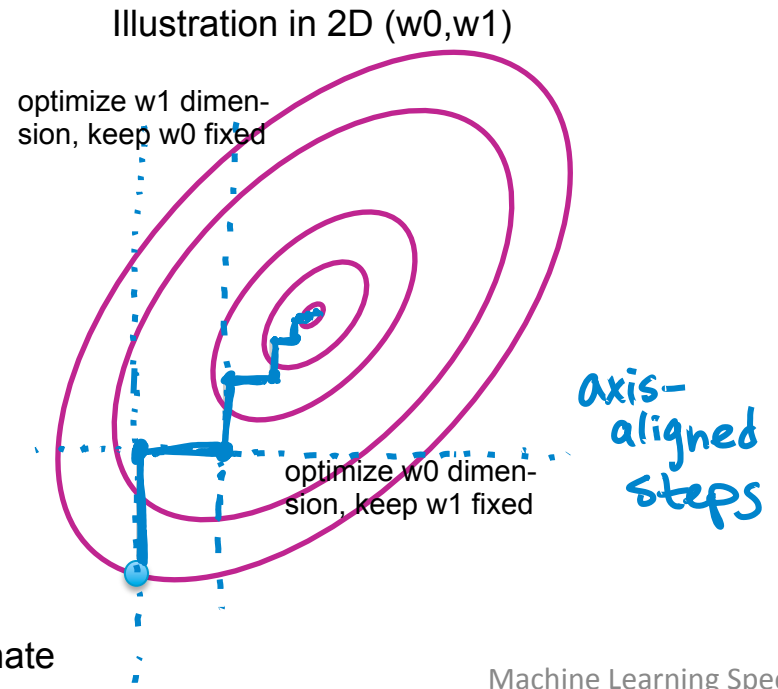
while not converged

  pick a coordinate j

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

values from previous iterations

just min over jth coordinate



objective: just minimize over jth coordinate

Comments on coordinate descent

How do we pick next coordinate?

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

No stepsize to choose! In contrast to gradient descent.

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:

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

Normalizer: Z_j

Apply same training scale factors to test data:

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

apply to test point

Normalizer: Z_j

summing over training points

e.g. housing application, a column might represent all instances of the number of square feet of houses in training data set.

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

unregularized => just RSS

Optimizing least squares objective one coordinate at a time

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

normalized features

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

$\mathbf{w}_{-j}$: all w_k for $k \neq j$

1d optimization: coordinate by coordinate

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

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

predicted value without jth feature

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

$\triangleq \rho_j$

by definition of normalized features, $= 1$

$$\Rightarrow -2\rho_j + 2w_j$$

Optimizing least squares objective one coordinate at a time

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

Set partial = 0 and solve

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

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

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

residual *without* feature j

prediction *without* feature j

Coordinate descent for lasso (for normalized features)

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

residual *without* feature j

prediction *without* feature j

Coordinate descent for lasso

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

while not converged

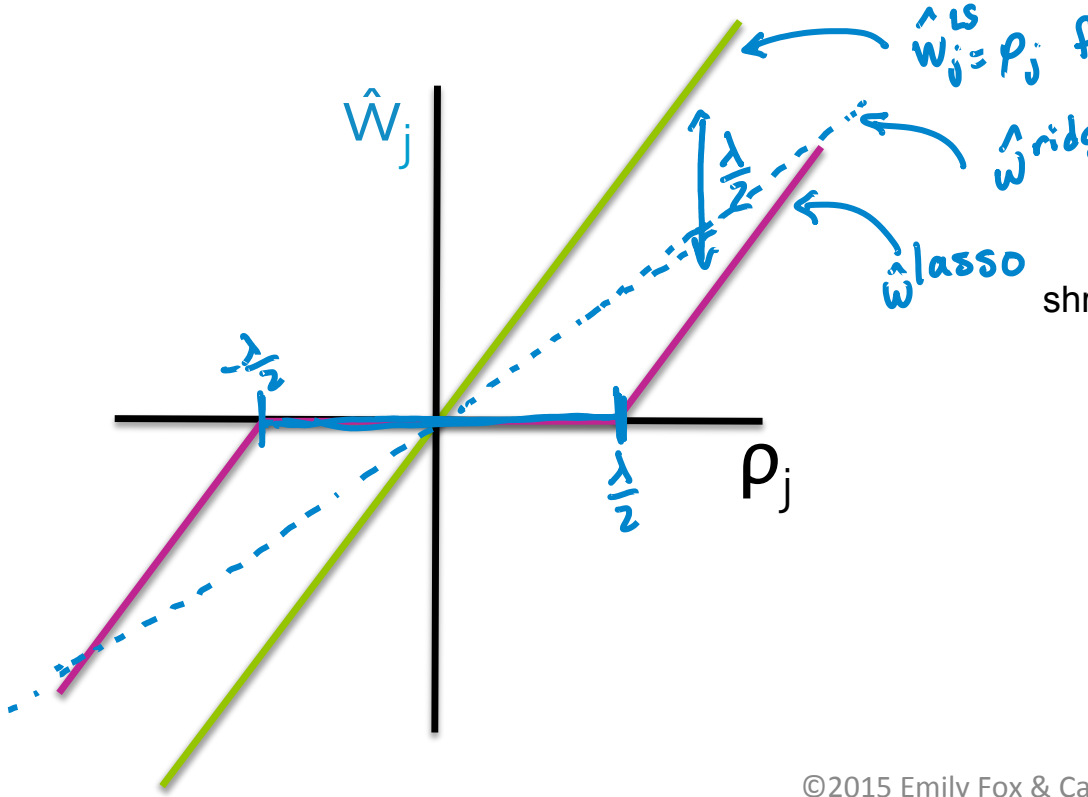
for j=0,1,…,D

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

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

Soft thresholding

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



(lambda = 0)

shrinks coefficient everywhere but never strictly to 0

shrink the weight relative to least square solution by lambda/2

How to assess convergence?

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

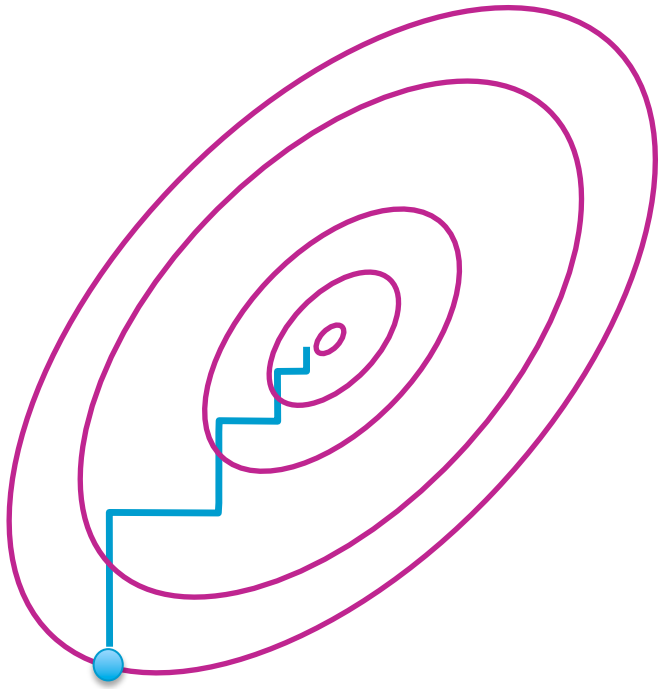
When to stop?

Gradient descent: stop when the magnitude of the gradient vector is below some tolerance epsilon.

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 < ε



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)

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

Coordinate descent for lasso with unnormalized features

Precompute: $z_j = \sum_{i=1}^{N} h_j(\mathbf{x}_i)^2$

zj is the normalizer.
Needs to be precomputed for each one of 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: $\hat{w}_j = \begin{cases} (\rho_j + \lambda/2)/z_j & \text{if } \rho_j < -\lambda/2 \\ 0 & \text{if } \rho_j \text{ in } [-\lambda/2, \lambda/2] \\ (\rho_j - \lambda/2)/z_j & \text{if } \rho_j > \lambda/2 \end{cases}$

Deriving the lasso coordinate descent update

Optimizing lasso objective one coordinate at a time

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

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

derive *without* normalizing features

Part 1: Partial of RSS term

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

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

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

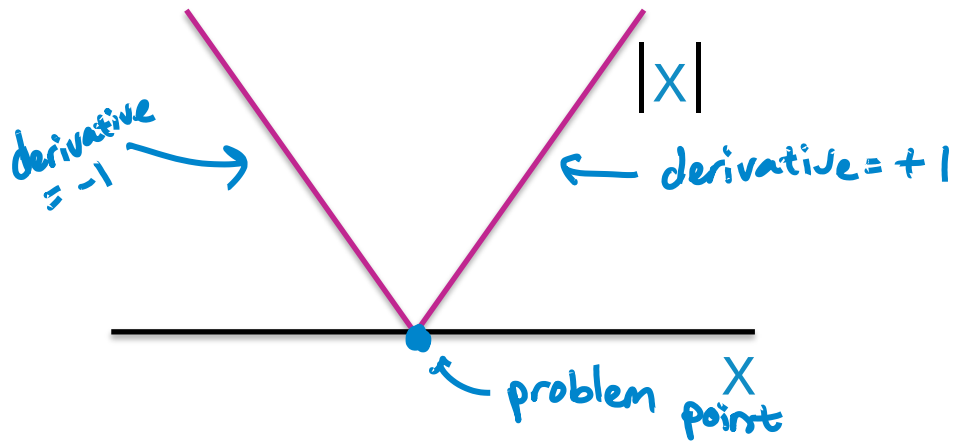
$$= \underbrace{-2\sum_{i=1}^{N} h_j(x_i)\left(y_i - \sum_{k\neq j} w_k h_k(x_i)\right)}_{\triangleq \rho_j} + 2 w_j \underbrace{\sum_{i=1}^{N} h_j(x_i)^2}_{\triangleq z_j}$$

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

Part 2: Partial of L_1 penalty term

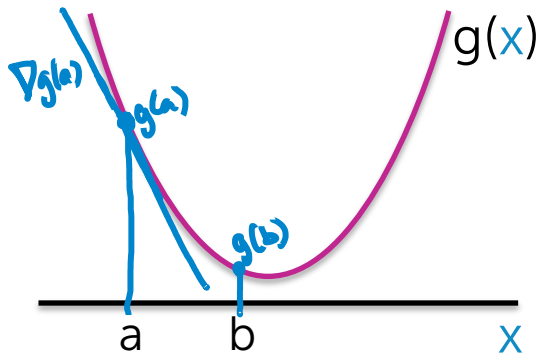
$$\text{RSS}(\mathbf{w}) + \lambda||\mathbf{w}||_1 = \sum_{i=1}^{N}(y_i - \sum_{j=0}^{D} w_j h_j(\mathbf{x}_i))^2 + \lambda \sum_{j=0}^{D} |w_j|$$

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



Subgradients of convex functions

Gradients lower bound convex functions:

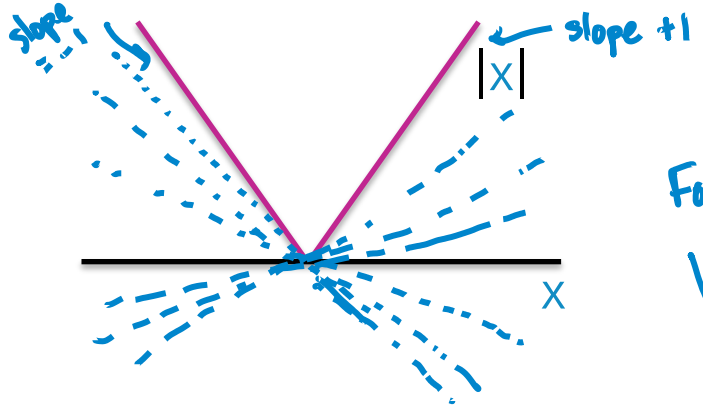


$$g(b) \geq g(a) + \nabla g(a)(b-a)$$

unique at x if function differentiable at x

Subgradients: Generalize gradients to non-differentiable points:

- Any plane that lower bounds function



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

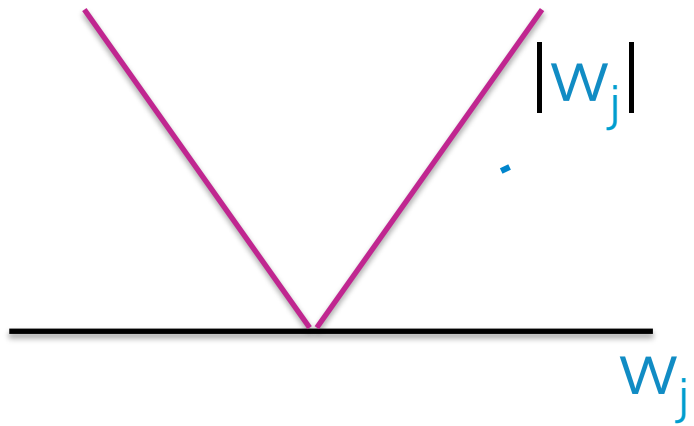
$v \in \partial g(x)$ subgradient of g at x if

$$g(b) \geq g(a) + v(b-a)$$

Part 2: *Subgradient* of L_1 term

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

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



Putting it all together…

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

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

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

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 $[-2\rho_j - \lambda,\ -2\rho_j + \lambda]$ to contain 0:

$-2\rho_j + \lambda \geq 0 \rightarrow \rho_j \leq \frac{\lambda}{2}$ $\quad -2\rho_j - \lambda \leq 0 \rightarrow \rho_j \geq -\frac{\lambda}{2}$

so that $\hat{w}_j = 0$ is an optimum

$-\frac{\lambda}{2} \leq \rho_j \leq \frac{\lambda}{2}$

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 $\rho_j > \frac{\lambda}{2}$

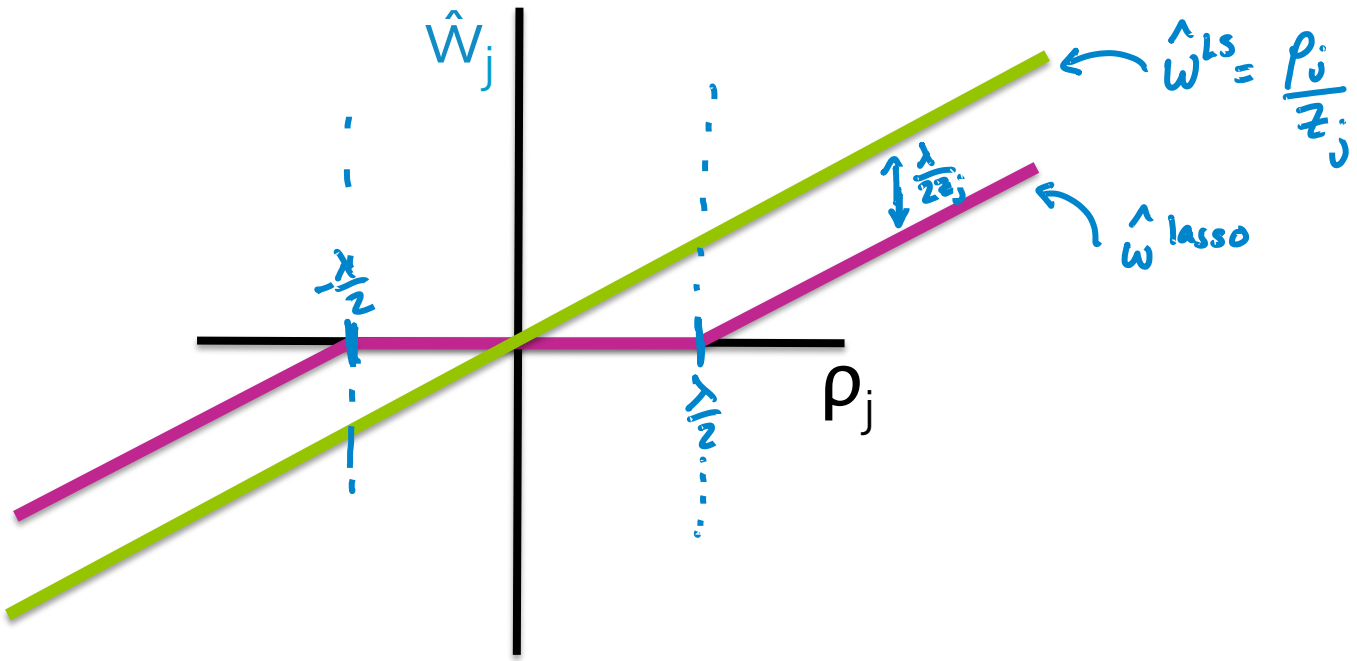
Optimal solution:
Set subgradient = 0

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

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

Soft thresholding

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



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

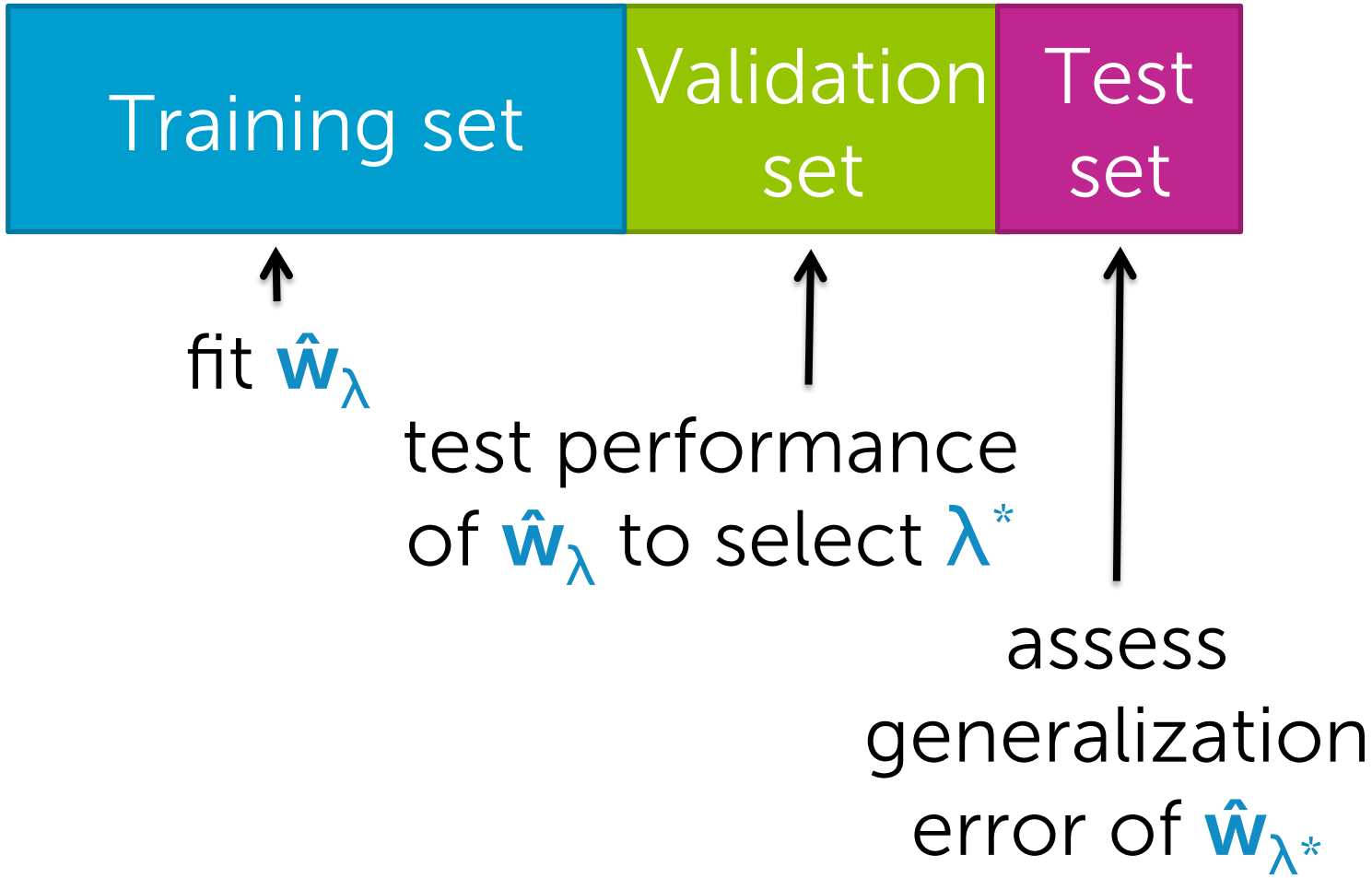
for j=0,1,…,D

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

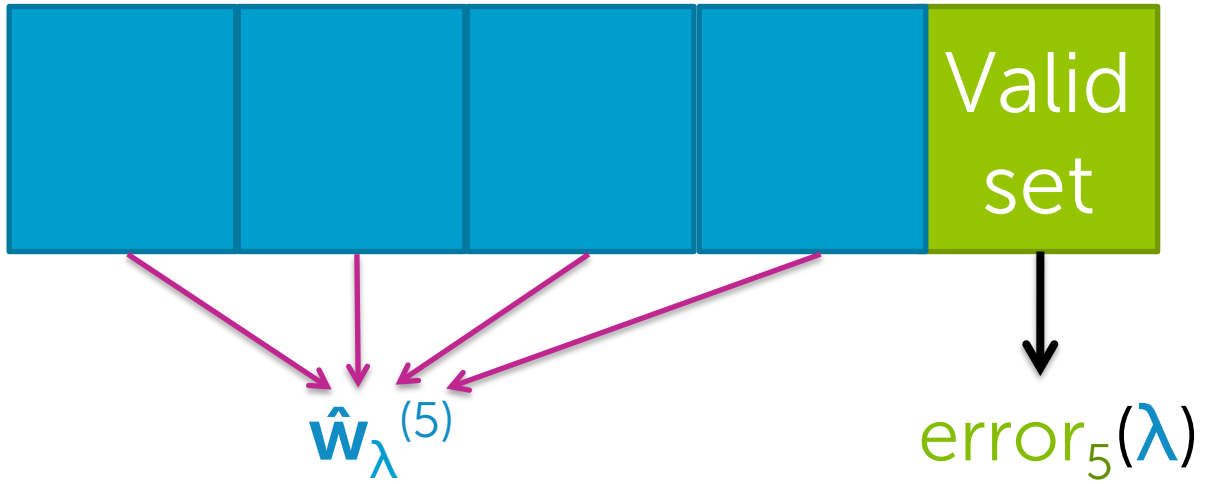
set: $\hat{w}_j = \begin{cases} (\rho_j + \lambda/2)/z_j & \text{if } \rho_j < -\lambda/2 \\ 0 & \text{if } \rho_j \text{ in } [-\lambda/2, \lambda/2] \\ (\rho_j - \lambda/2)/z_j & \text{if } \rho_j > \lambda/2 \end{cases}$

How to choose λ

If sufficient amount of data...



K-fold cross validation



For k=1,…,K

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

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

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

Debiasing the lasso solution

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 same reduced model

These plots show a little illustration of the benefits of debiasing

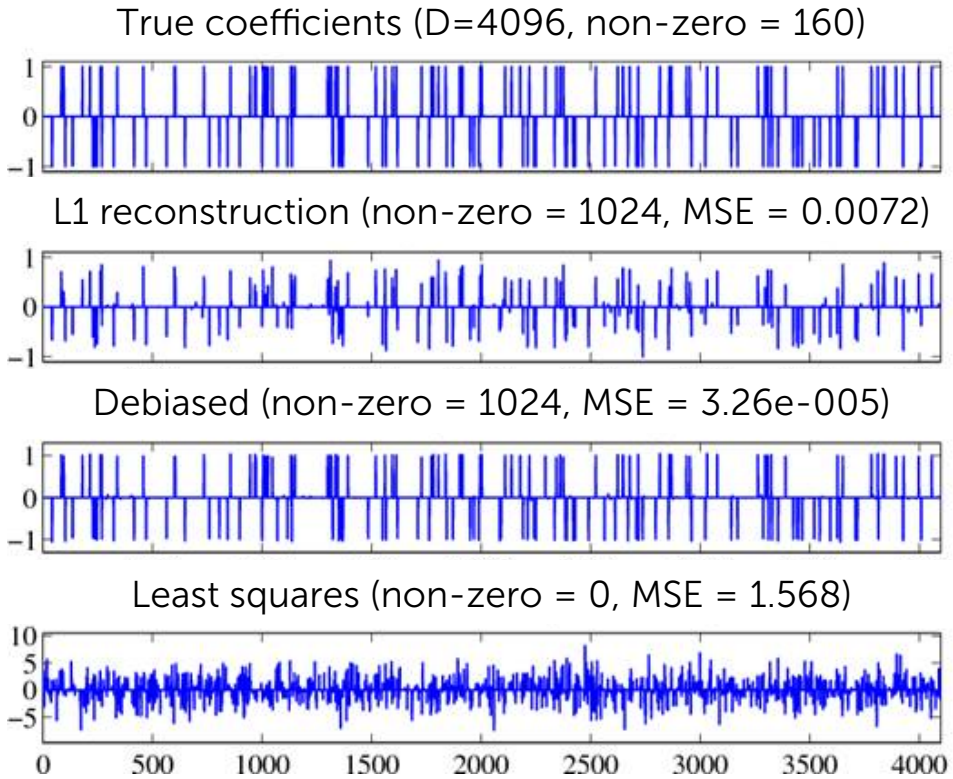


Figure used with permission of Mario Figueiredo
(*captions modified to fit course*)

Issues with standard lasso objective

1. With group of highly 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

Summary for feature selection and lasso regression

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

What you can do now…

- Perform feature selection using "all subsets" and "forward stepwise" algorithms
- Analyze computational costs of these algorithms
- Contrast greedy and optimal algorithms
- Formulate lasso objective
- Describe what happens to estimated lasso coefficients as tuning parameter λ is varied
- Interpret lasso coefficient path plot
- Contrast ridge and lasso regression
- Describe geometrically why L1 penalty leads to sparsity
- Estimate lasso regression parameters using an iterative coordinate descent algorithm
- Implement K-fold cross validation to select lasso tuning parameter λ