Feature selection, Regularization

2019-08-06

Feature Construction

Instead of $f(\mathbf{x})$, we write $f(\phi(\mathbf{x}))$, or $f(\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \phi_3(\mathbf{x}), \ldots, \phi_d(\mathbf{x}))$.

where ϕ_k are sometimes called *feature functions* or *basis functions* that define new *features* in terms of what we might call the “raw data”

$$\begin{tikzcd}
\text{Data Space} \arrow[r] & \text{Feature space} \\
\mathbf{x} & \phi(\mathbf{x})
\end{tikzcd}$$

- The Feature space is typically more high-dimensional than the data space
- The process is therefore sometimes called “basis expansion”

Example: Polynomial features

For a single variable:

Input: x

Transformation:

$$\phi_0(x) = 1, \phi_1(x) = x, \phi_2(x) = x^2, \ldots, \phi_d(x) = x^d$$

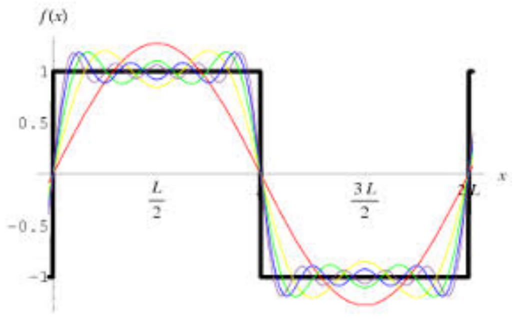
For multiple variables:

Input: $\mathbf{x} = [x_1, x_2, x_3]$

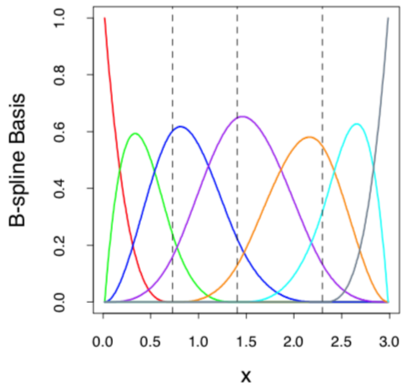
1st order: 3	x_1		x_2		x_3	
2nd order: 6	x_1^2	$x_1 x_2$	x_2^2	$x_2 x_3$	x_3^2	$x_1 x_3$
2nd order: 10	x_1^3	$x_1^2 x_2$	$x_1 x_2^2$	x_2^3	$\cdots$	$x_1 x_2 x_3$

$\cdots$

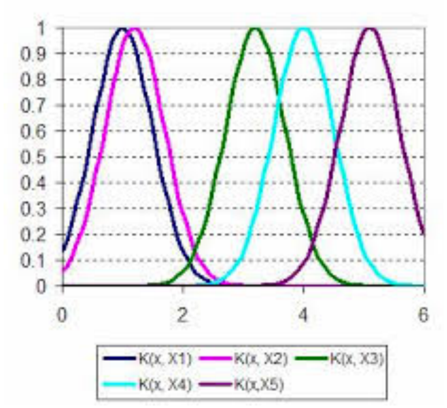
Other important bases



Fourier set



Splines

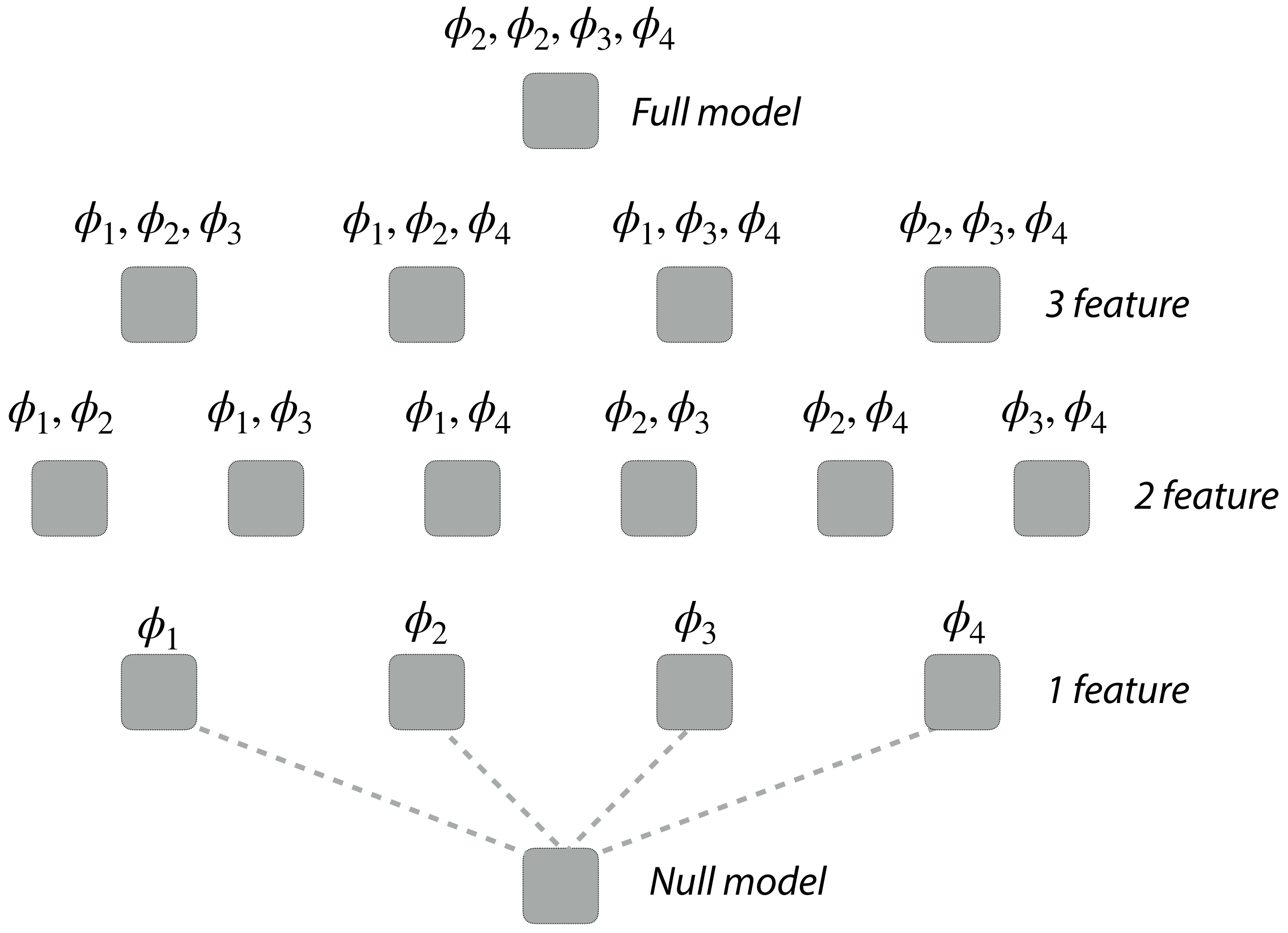


Radial basis functions

Feature Selection (*JWHT* 6.1)

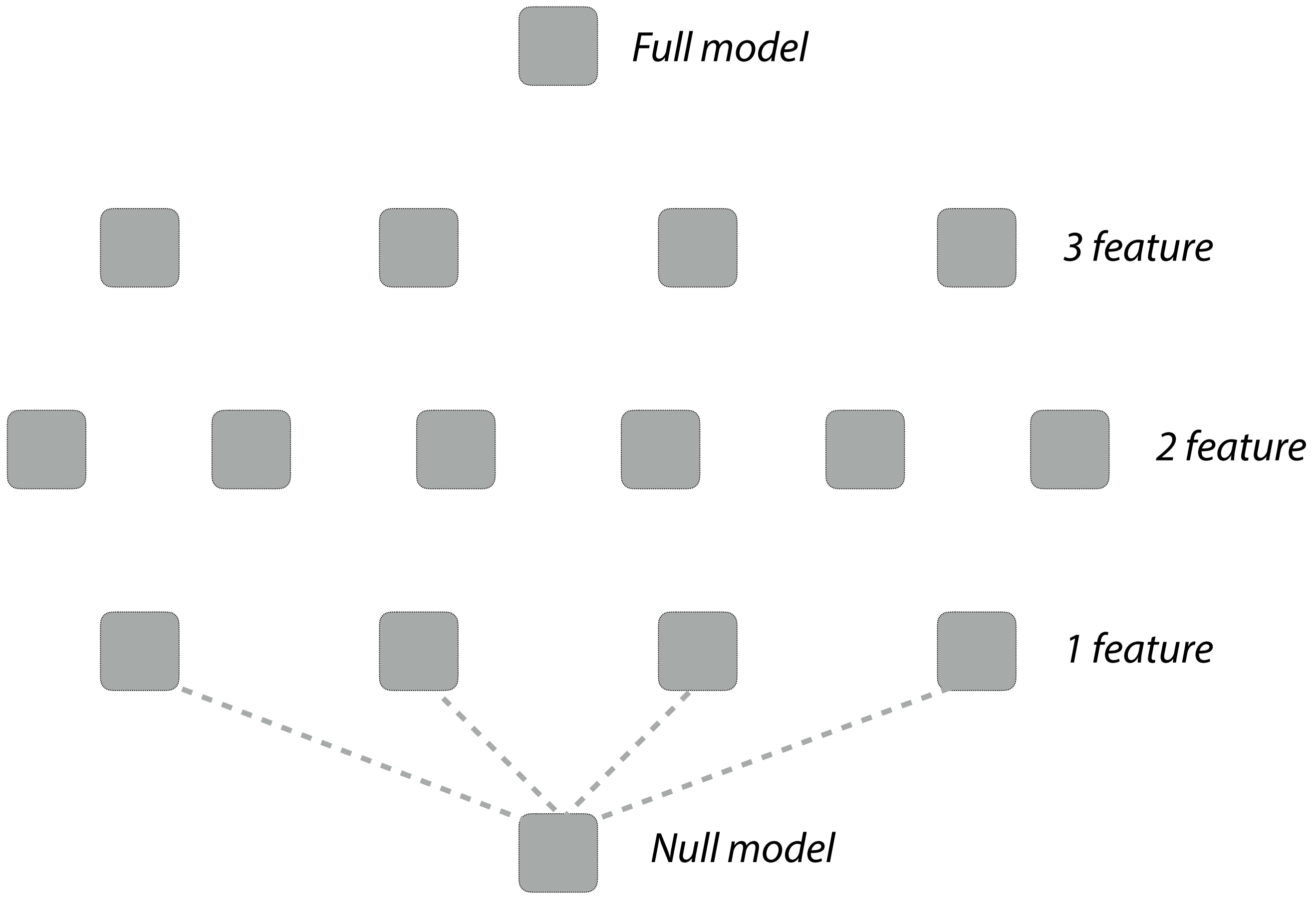
- Goal: select the best combination of features
- If we have K, we have 2^K ways of combining them.
- The full set of models can be visualized in a *model graph*

Feature selection



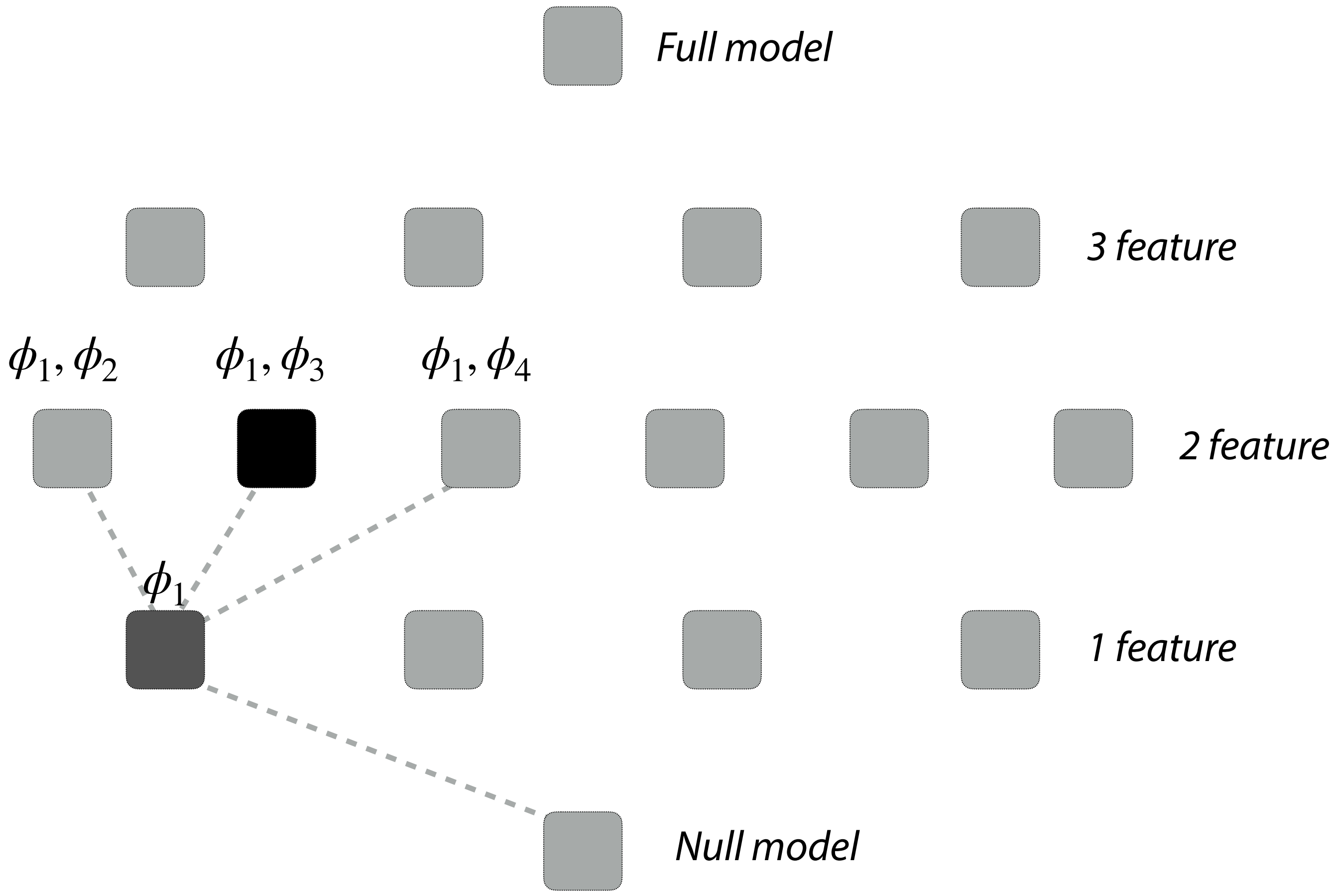
Example model graph with 4 features

Feature selection:forward



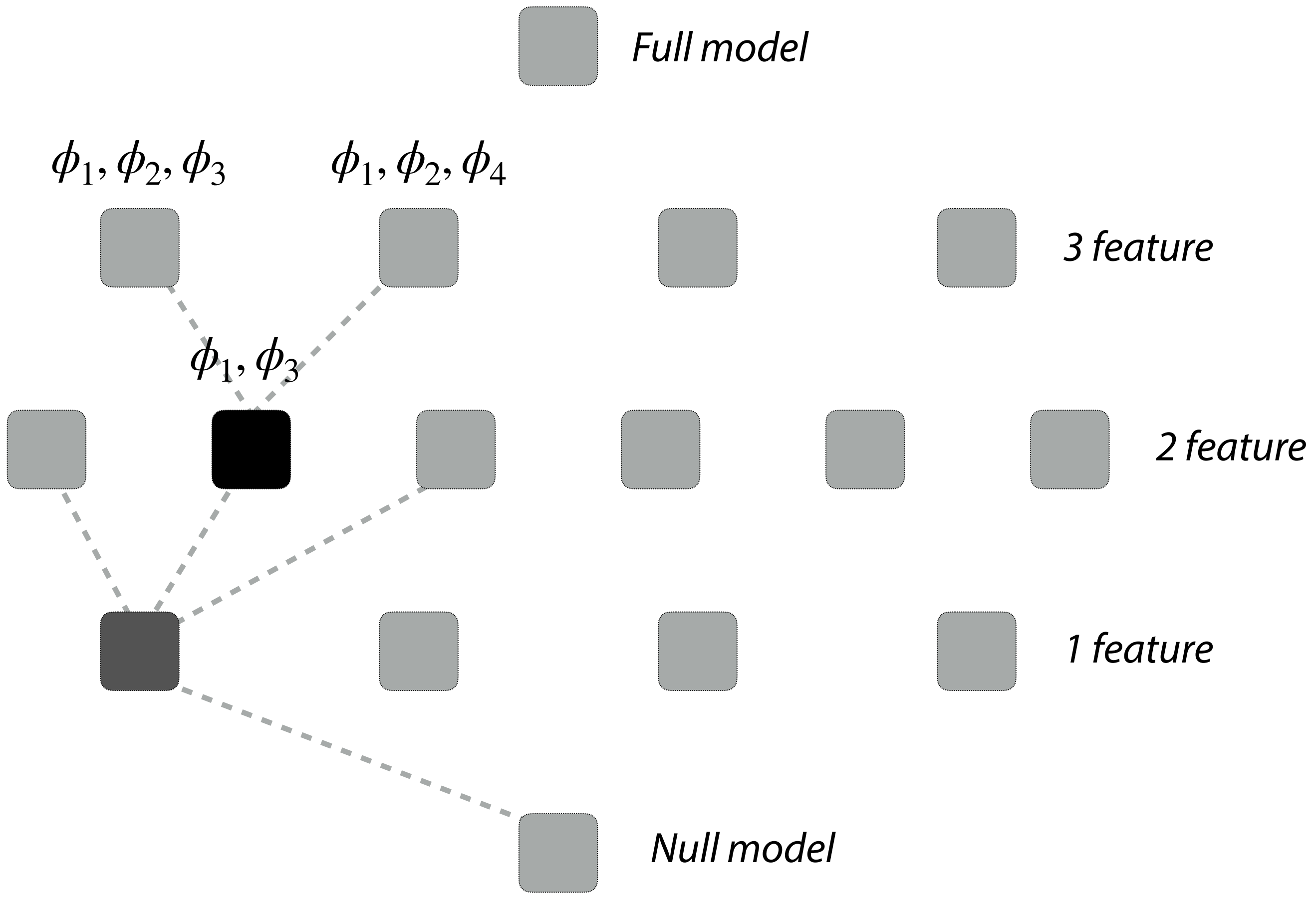
- Start with the null model
- Compare with models with 1 feature
- Choose the best one

Feature selection:forward



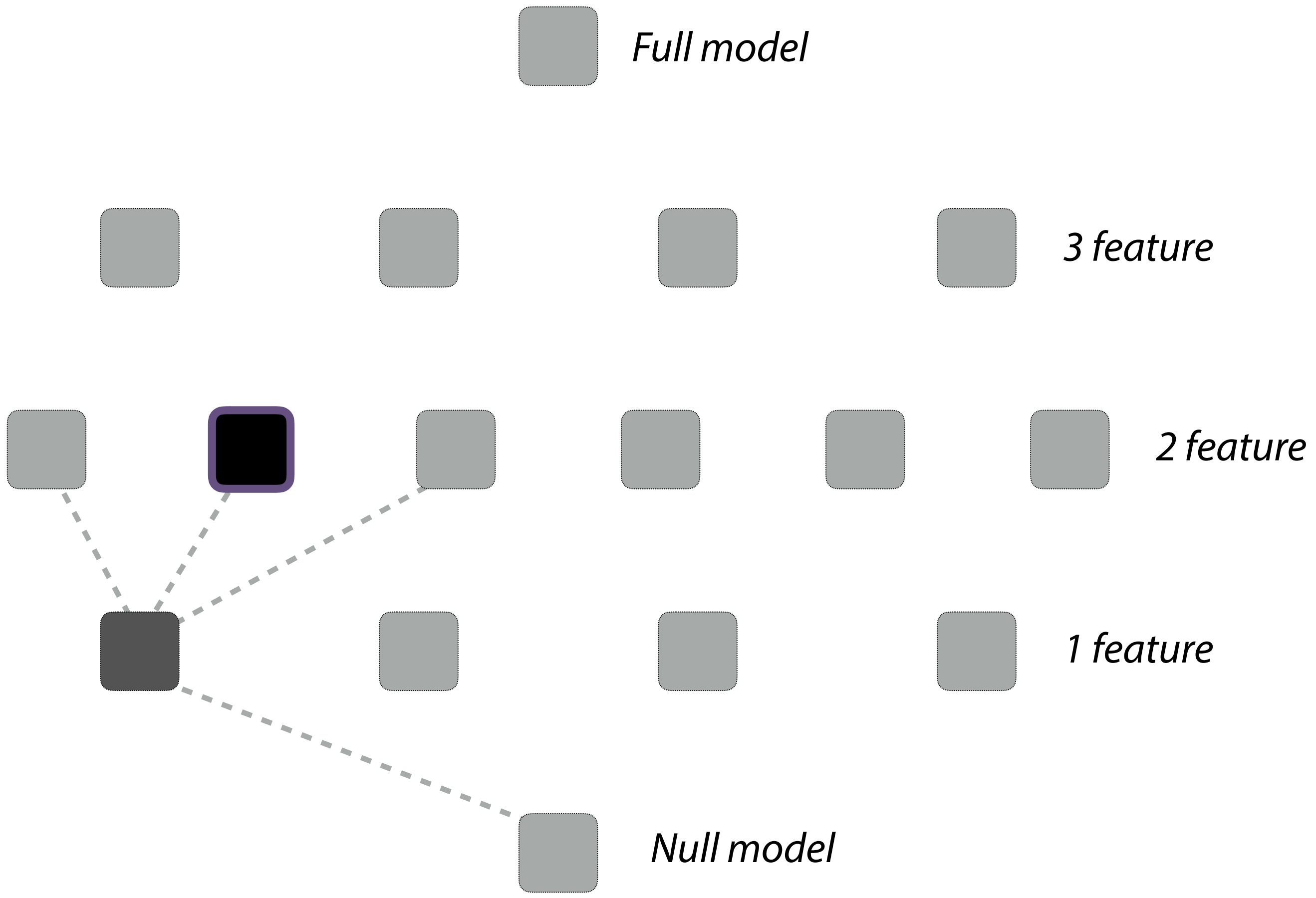
- Test only models that include already chosen feature
- Choose the best one

Feature selection:forward

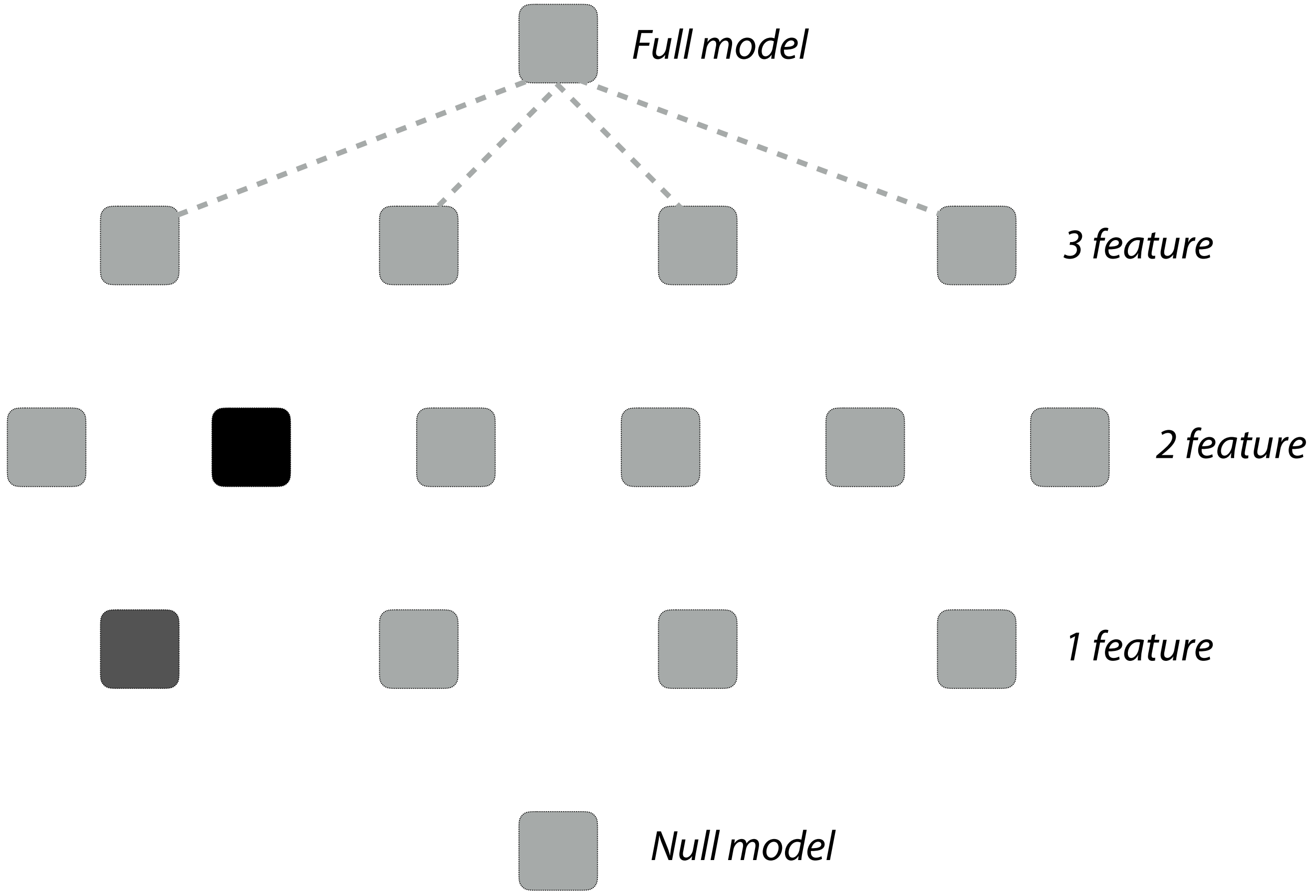


- Test only models that include already chosen features
- If the CV error does not decrease anymore, stop

Feature selection:forward

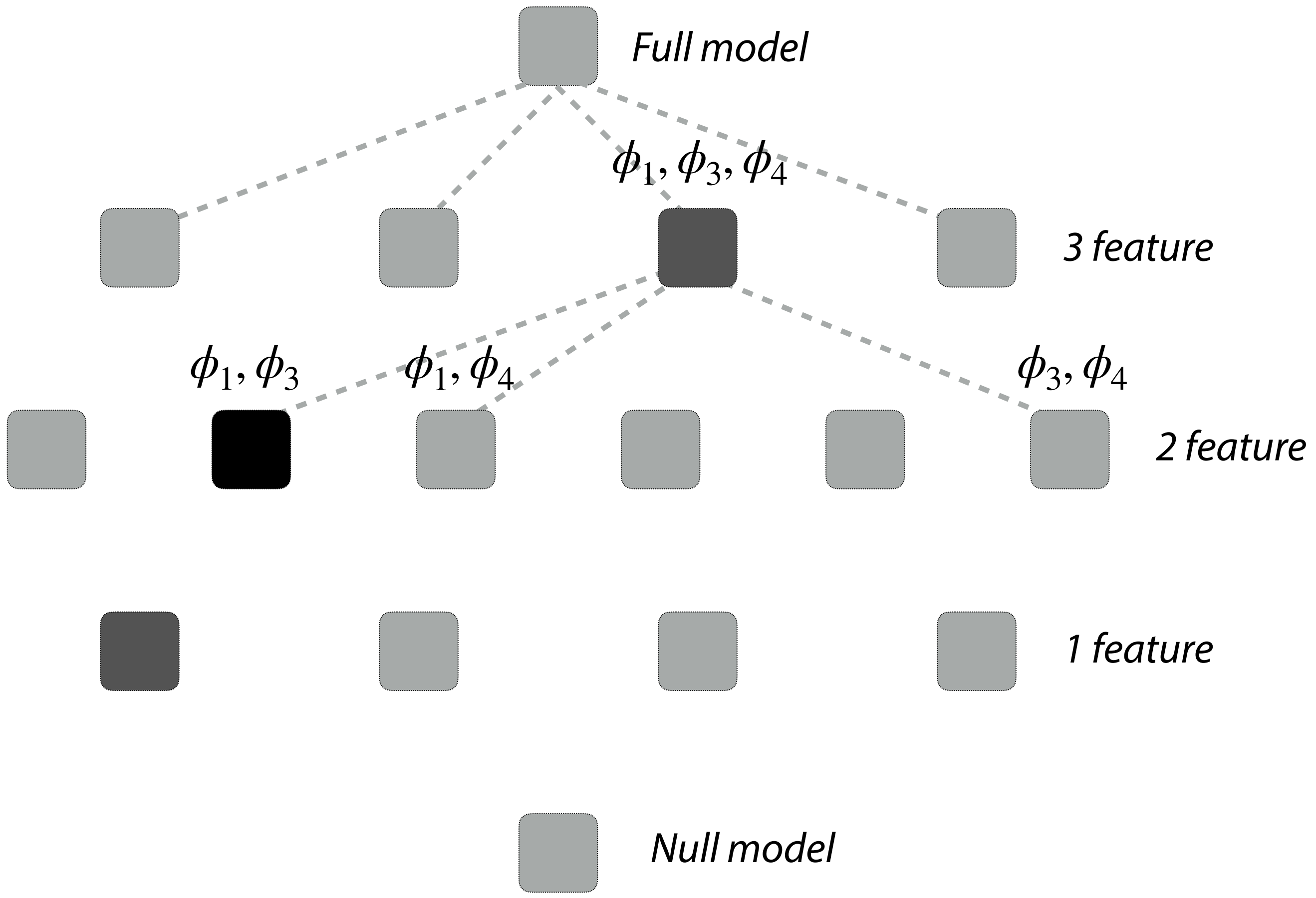


Feature selection:backward



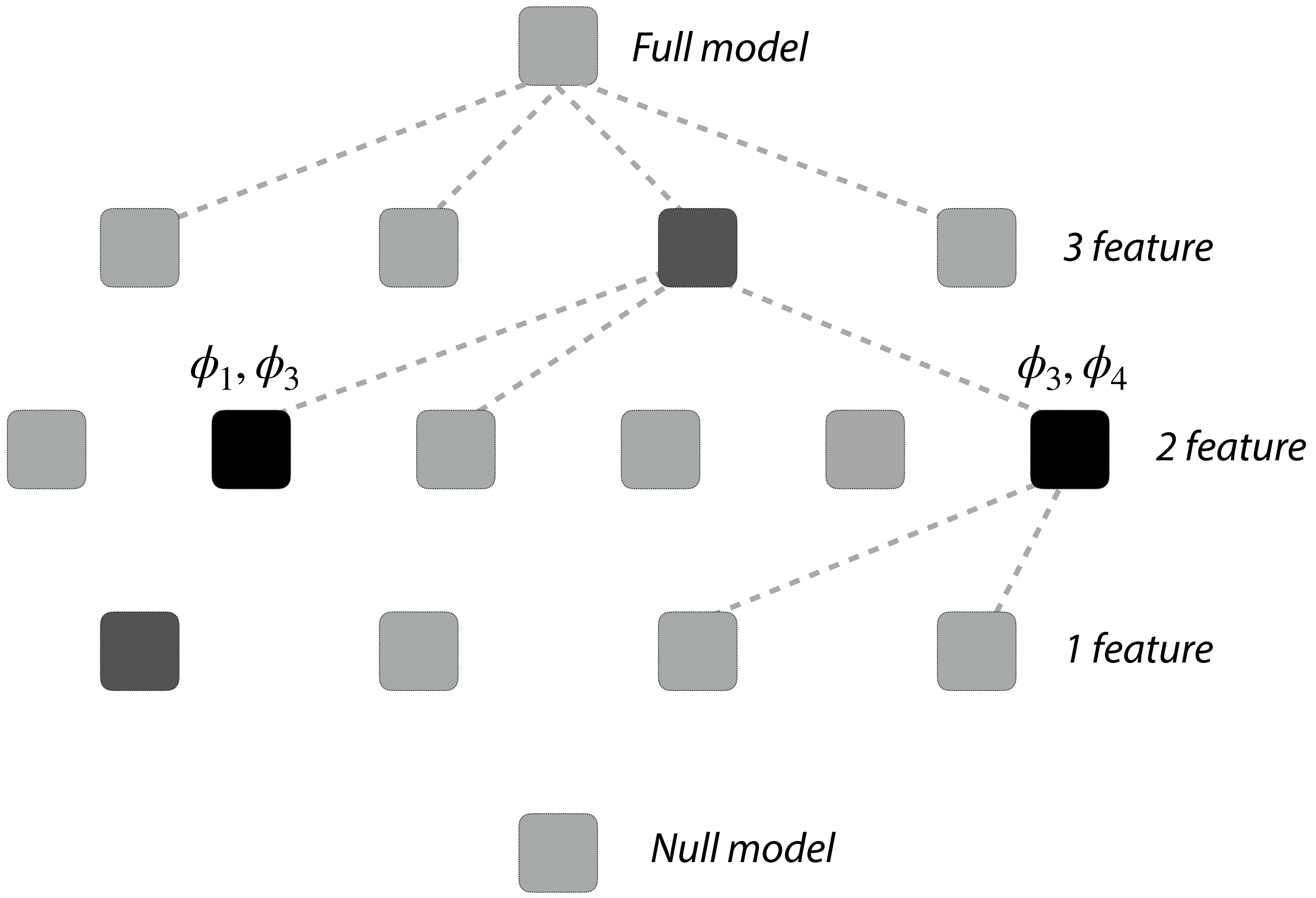
- Start with the full model
- Drop 1 feature at a time
- Choose the model that improves CV error most

Feature selection:backward



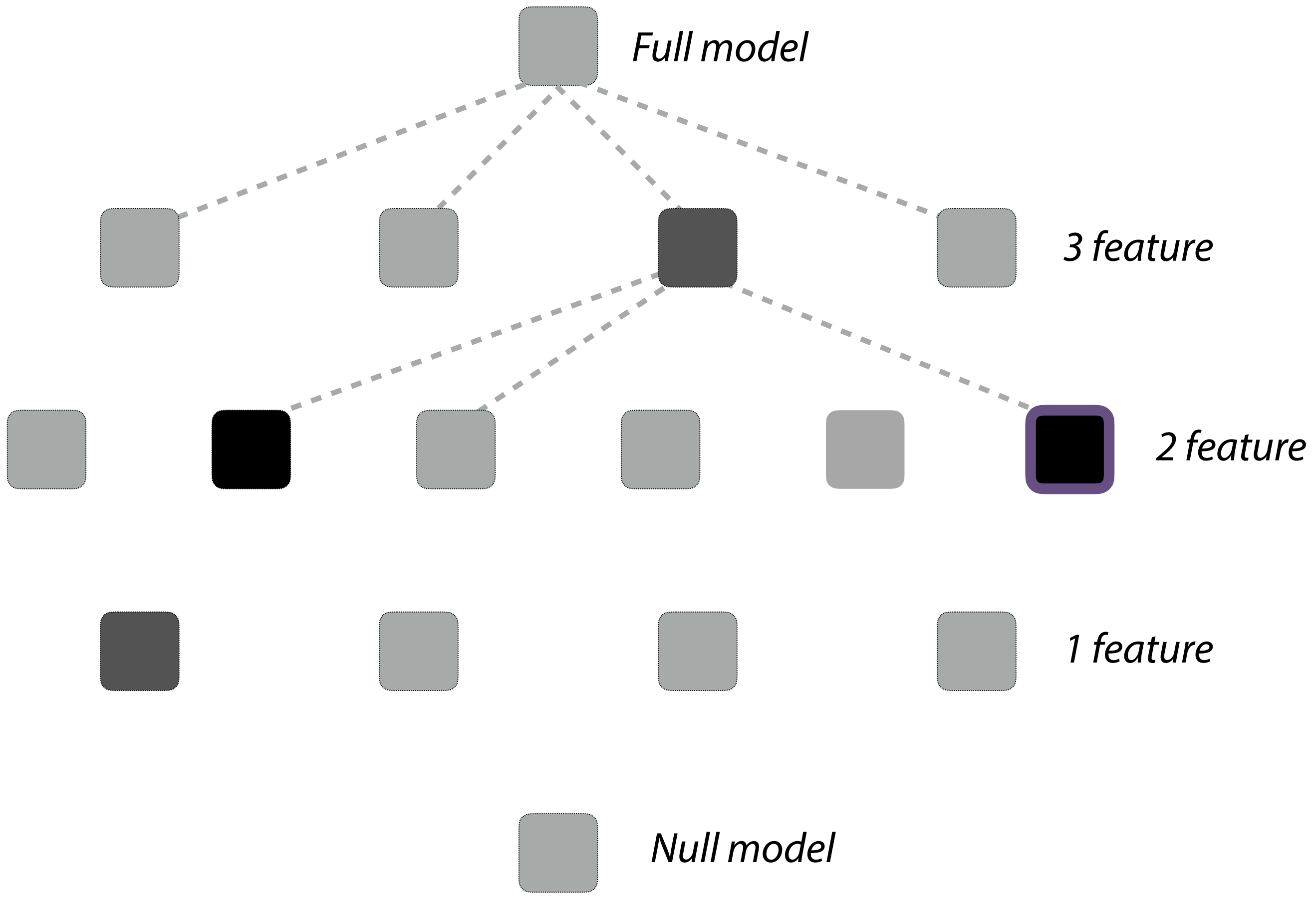
- Drop any of the next features
- Choose the model that improves CV error most

Feature selection:backward



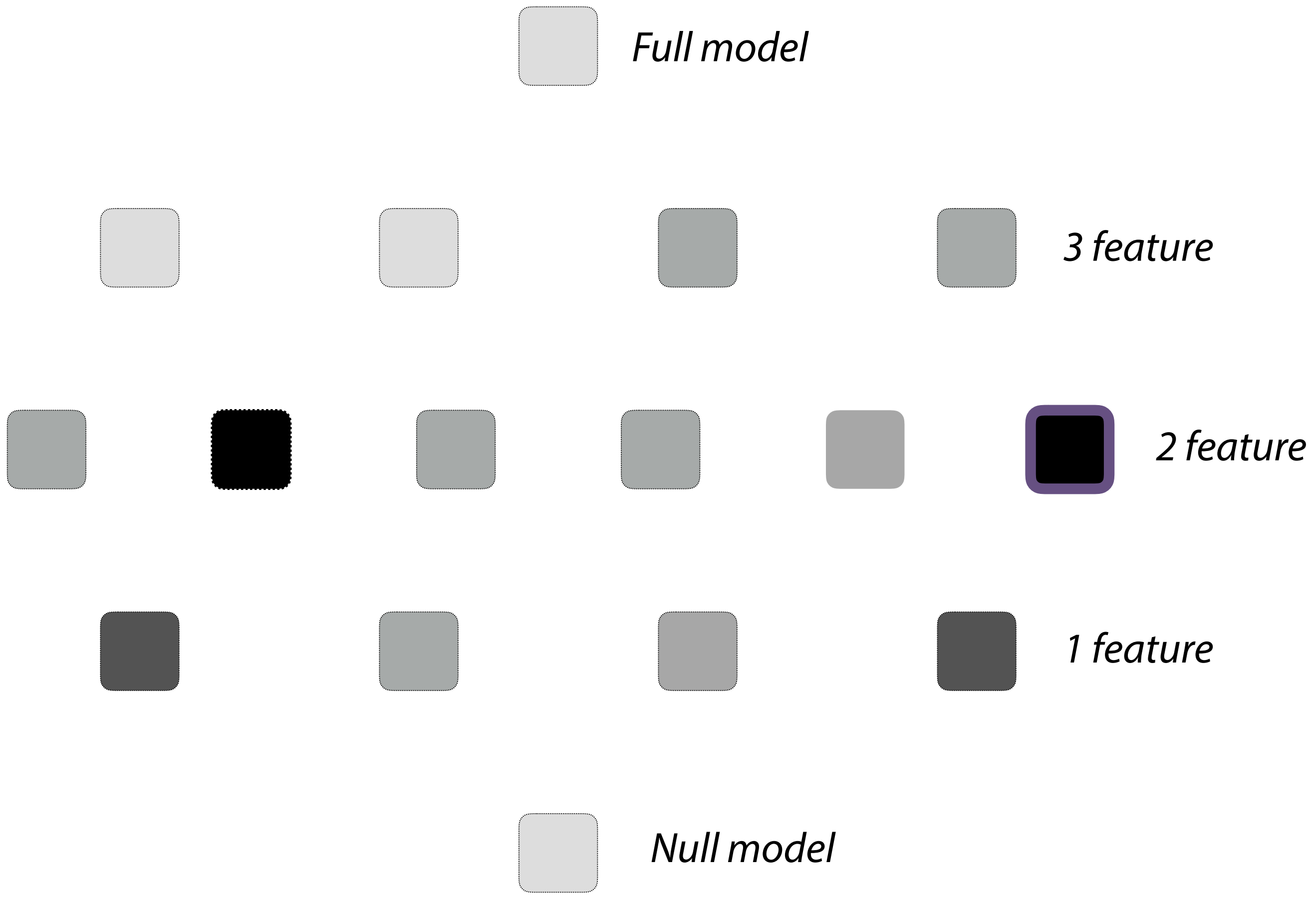
- Drop any of the next features
- Stop if CV error starts increasing for all features

Feature selection:backward



- Backwards and forward selection can give different answers
- For high-dimensional problems they often do

Exhaustive model search



- Fit all the models and choose the best one
- In a probabilistic setting you can take model uncertainty into account (model averaging)

Feature Selection (*JWHT* 6.1)

- Goal: omit features that are not helpful for prediction.
- If we have K, we would have to test 2^K models (exhaustive model search).
- Forward selection: Start with no features, try adding each one, measure performance. Keep the best features. Then try to add the next one, until all are included (or the validation error does not decrease anymore).
- Backward selection: Start with all features, try removing each one (separately), keep the best model that has had one feature removed. Repeat until all features are removed.
- Bayesian model search (Clyde, 1999) is a probabilistic version of this that provide more stable results.

Feature Selection

- There are many methods.
  - Books, paper, and software are often imprecise: “We are using backward selection…” Always investigate to determine *exactly* what they are doing, report precisely in your own work.
- Forward and backward selection can be used with *any* supervised learning method. (Sometimes called **wrapper** methods.)
- There are other feature selection techniques that are specific to individual learning methods.

REGULARIZATION

Regularization

- Situation
  - Not practical to have too many parameters for the given amount of data we have.
  - But we are not sure which are important - and all could have a relatively small influence
  - Want to consider them all together

Regularization

- Strategy
  - Include all features in the model
  - *Change our optimization objective* to prefer "simple models"
  - i.e. ones with coefficients near to zero
    - Why?

Regularization

- Rather than explicitly trying different model spaces, we try one model space but change the optimization criterion
- Objective = Loss + Regularizer
- (Previously, Objective = Loss)

Regularization Example: Ridge
HTF 3.4

$$J_{\text{ridge}} = \sum_{i=1}^{n} \left(y_i - \beta^{\mathsf{T}} \mathbf{x}_i\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

$\{y_i, \mathbf{x}_i\}$: *n* observations

J: Objective function (L: Loss function)

β: Vector of *j* regression parameters to optimize

λ: Ridge coefficient

Regularization Example: Ridge
HTF 3.4

$$J_{\text{ridge}} = \sum_{i=1}^{n} \left(y_i - \beta^{\mathsf{T}} \mathbf{x}_i\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Solutions have coefficients nearer to 0 than OLS on same data.

More stable solutions with limited data.

We can even fit models that we otherwise cannot optimize

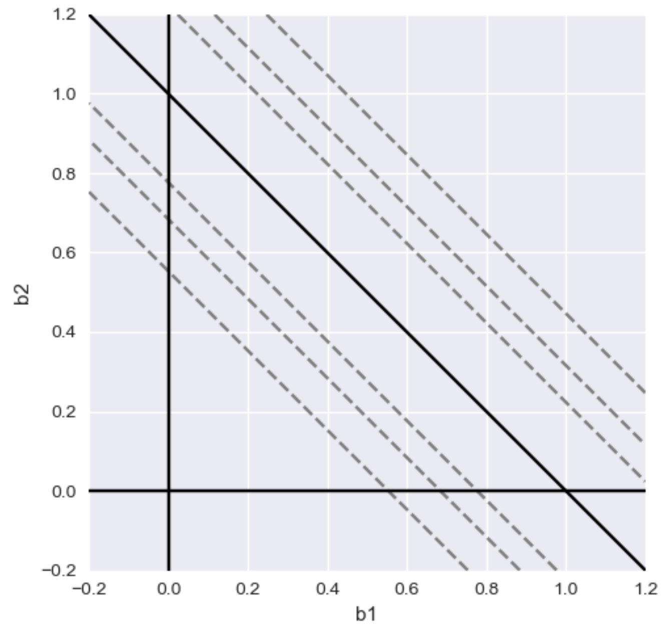
Ridge regression

$\hat{y} = b_1 x_1 + b_2 x_2$

Let's take a degenerative example with only one data point

$y = 1, x_1 = 1, x_2 = 1$ $\hat{y} = b_1 + b_2$

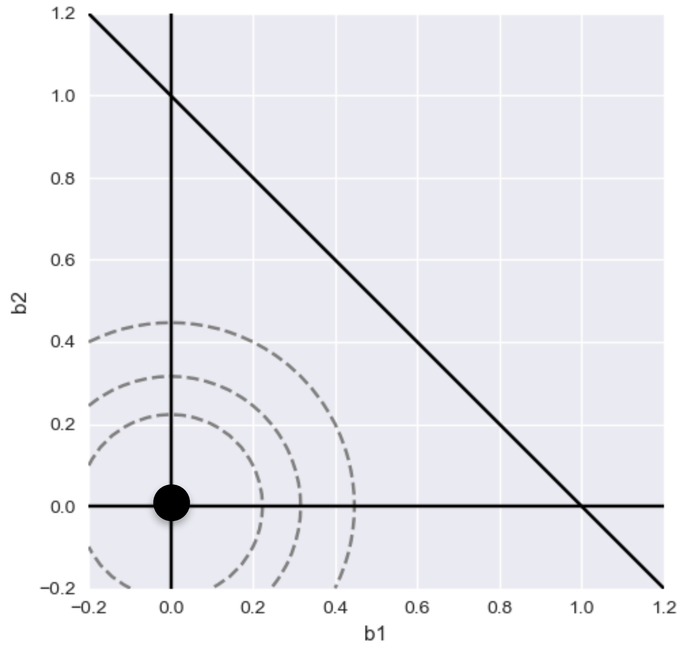
Squared Loss



$L = (b_1 + b_2 - 1)^2$

The cost is minimized anywhere along the black line

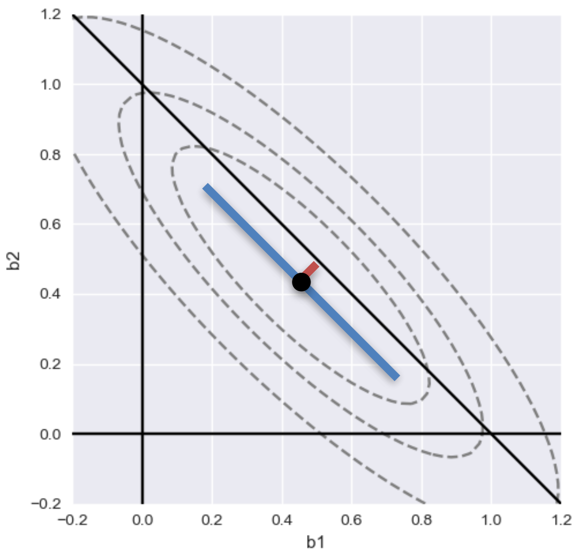
Regularization term



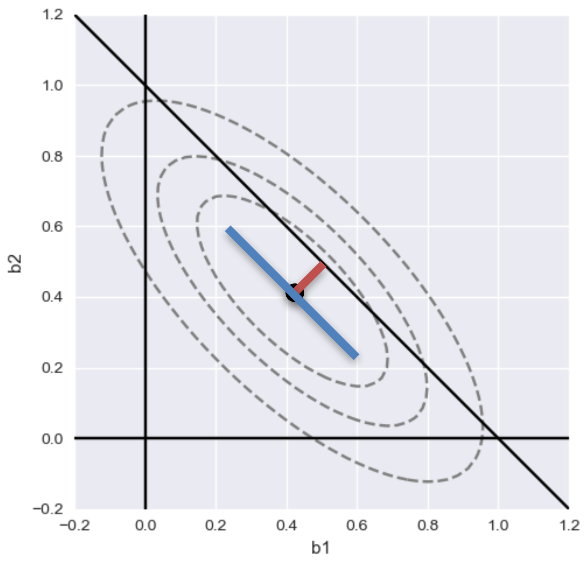
$\lambda(b_1^2 + b_2^2)$

Objective: Penalized loss function

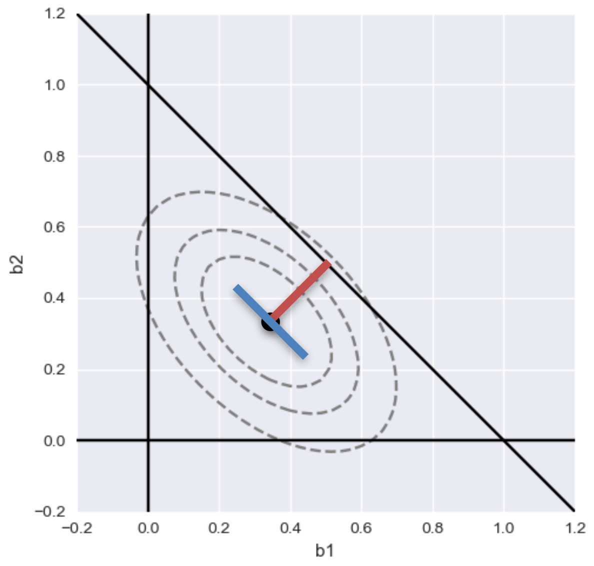
$$J = (b_1 + b_2 - 1)^2 + \lambda(b_1^2 + b_2^2)$$



$\lambda = 0.2$



$\lambda = 0.4$



$\lambda = 1$

🟥 Bias

🟦 Variance

- Regularization gives saturated model unique solutions
- Regularization reduces the variance of parameters + prediction
- Regularization increases bias

Bias-Variance tradeoff: *Regularization makes models simpler*

Ridge regression

a. $\hat{y} = b_1 x_1 + b_2 x_2$

b. $\hat{y} = b_1 x_1 + 2b_2 x_2$

For un-regularized regression, these two models are equivalent

Why?

a. $\hat{y} = b_1 x_1 + b_2 x_2$

b. $\hat{y} = b_1 x_1 + 2b_2 x_2$

When we regularize, the solution depends on the scaling of the regressors (x)

Why?

For $y = 1, x_1 = 1, x_2 = 1$ and small λ, we get the solution for model a.

$$b_1 \approx 0.5$$

$$b_2 \approx 0.5$$

For $y = 1, x_1 = 1, x_2 = 1$ and small λ, we get the solution for model b.

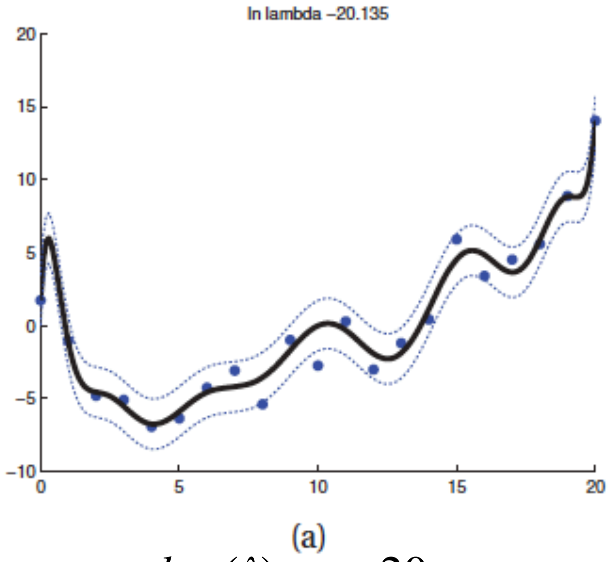
$$b_1 \approx 1/3$$

$$2b_2 \approx 2/3$$

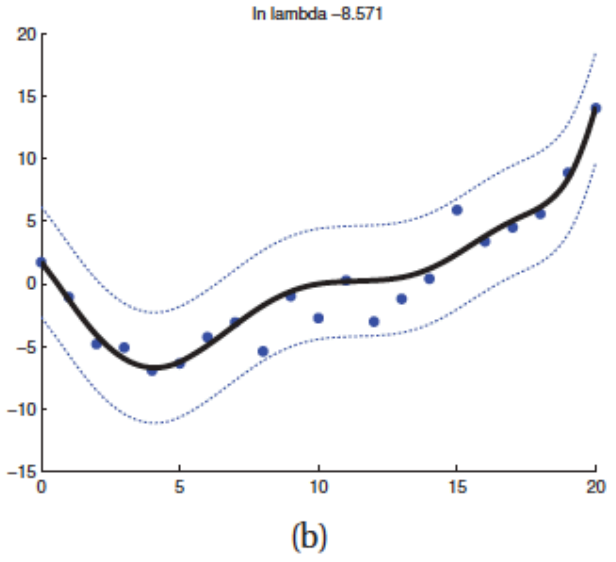
Regularization

- Since scaling changes the result, one strategy is to z-standardize regressors before submitting them to the model $x = \frac{x - mean(x)}{std(x)}$
- Another possibility is to have different regularization coefficients (λ) for different regressors (sometimes discussed as Tikhonov regularization)

Example of Ridge regression



$log(\lambda) = -20$



$log(\lambda) = -8$

Example shows N=21 data points, fit with a polynomial of degree 14

The dotted line show the predictive density, not the confidence interval!

posterior predictive density: $p(\hat{f}(x) + \epsilon \,|\, data)$

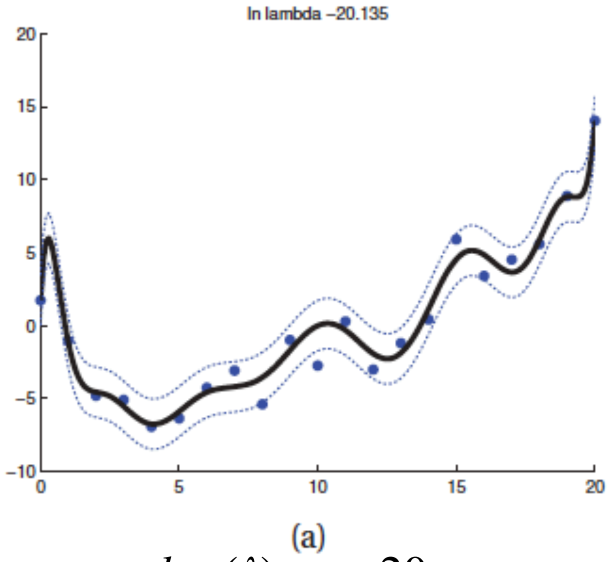
uncertainty of prediction: $p(\hat{f}(x) \,|\, data)$

Predictive density also includes the estimate of the irreducible error $\hat{\sigma}_\epsilon^2$

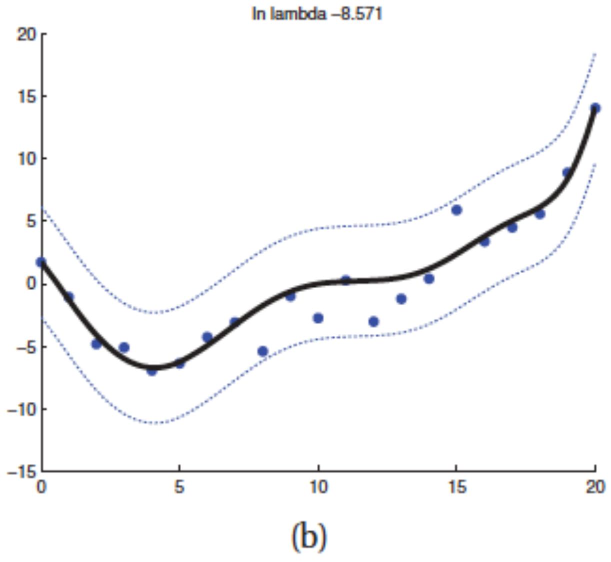
So why is the predictive interval larger for the simpler model?

Murphy, 7.5

How do we choose λ?



$log(\lambda) = -20$



$log(\lambda) = -8$

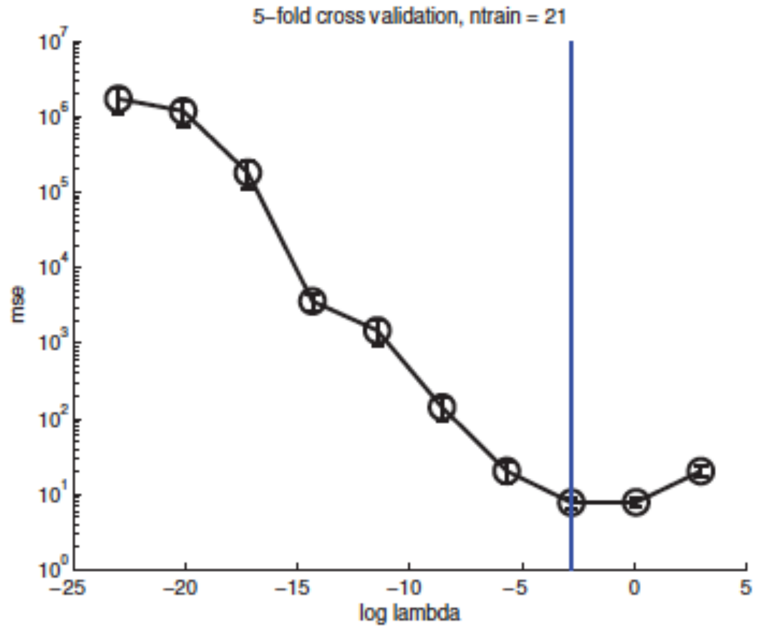
Example shows N=21 data points, fit with a polynomial of degree 14

Which model is the more appropriate one?

Murphy, 7.5

How do we choose λ?

- Setting λ is a model selection problem (larger λ lead to simpler model)
- We can use the training-validation-test strategy (possibly with cross-validation) that we learned last lecture



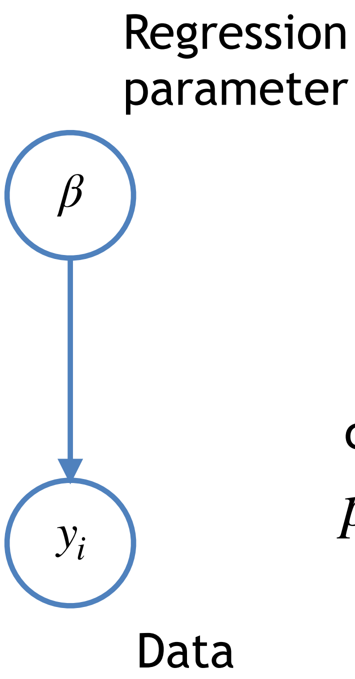
We can conduct a grid search, trying out different values for λ and measuring validation error for each value.

Once we determine the best λ, what is then our estimate of our test error?

How would we use nested CV for determining the test error?

Murphy, 7.5

Probabilistic view of regularization



	prior	Normal example	negative log-likelihood
	$p(\beta)$	$N(0,\sigma_\beta^2)$	$-\ell = \dfrac{\lVert\beta\rVert^2}{2\sigma_\beta^2} + \ldots$
	conditional $p(y_i \mid \beta, x)$	$N(\beta^T x, \sigma_\epsilon^2)$	$-\ell = \dfrac{\sum (y_i - \beta^T x_i)^2}{2\sigma_\epsilon^2} + \ldots$

$$N(\underbrace{\beta^T x}_{\text{Predicted value}}, \underbrace{\sigma_\epsilon^2}_{\text{Irreducible error}})$$

Complete log-likelihood

$$log p(\beta) p(y \mid \beta, x)$$

$$J = \frac{\sigma_\epsilon^2}{\sigma_\beta^2} \lVert\beta\rVert^2 + \sum (y_i - \beta^T x_i)^2$$

- Ridge regression is equivalent to imposing a normal prior on β
- The regularization factor weights our measurement vs. parameter uncertainty $\lambda = \sigma_\epsilon^2 / \sigma_\beta^2$

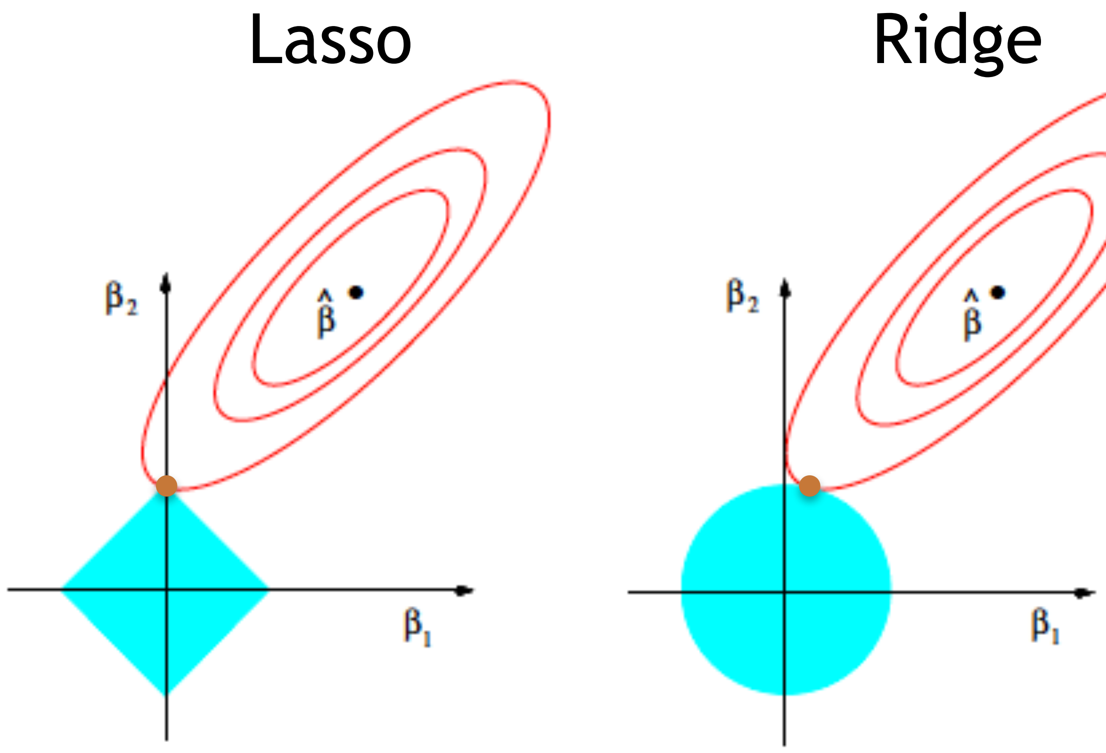
L_1 Regularization: LASSO *HTF* 3.4

We can also penalize with the L1-norm:

$$J_{\text{Lasso}} = \sum_{i=1}^{n} \left(y_i - \beta^{\mathsf{T}} \mathbf{x}_i\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

- LASSO: least absolute shrinkage and selection operator
- In the probabilistic setting, what type of prior would this correspond to?
- Coefficient will be again be closer to zero (bias) and have less variance
- But how does the solution differ from Ridge regression?

L_1 Regularization: LASSO



$\hat{\beta}$: OLS Estimate

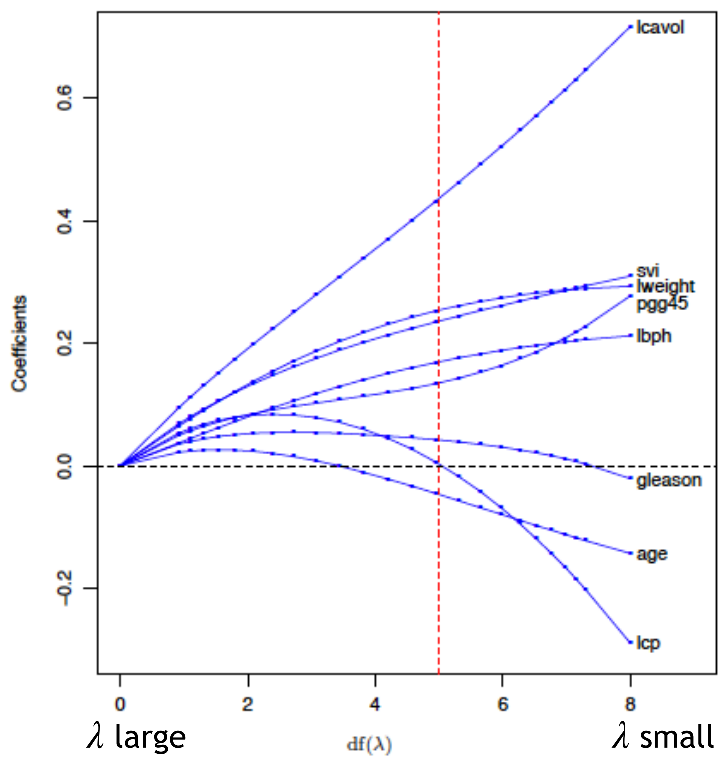
Red Contours: Squared loss

Blue shapes: Regularization penalty

Lasso tends to reduce specific coefficient to zeros

Prostate cancer example

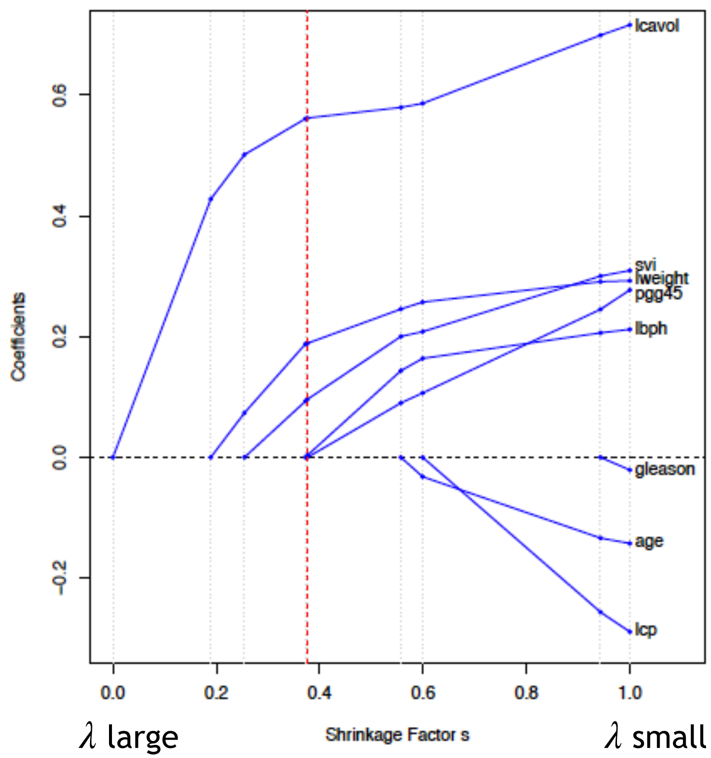
Ridge regression



λ large λ small

- Ridge moves all parameters together down to zero
- Note how some can change sign? Why is this?

Lasso



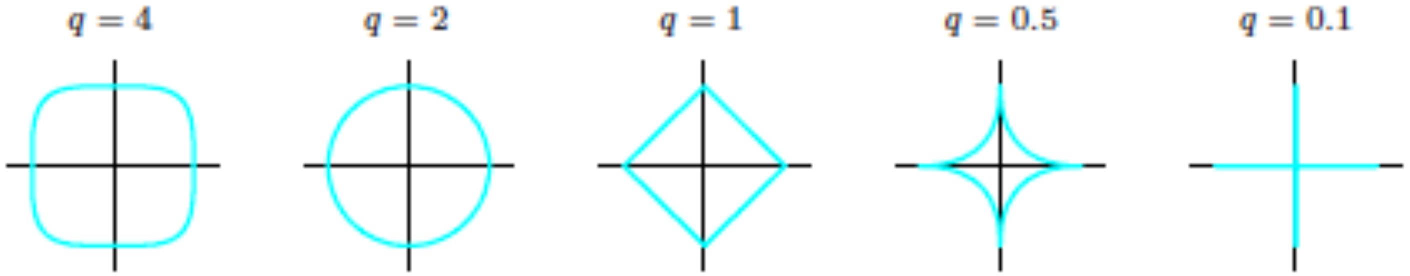
λ large λ small

- Lasso drops out one parameter at a time.
- Lasso has an efficient algorithm to find these points.

HTF 3.4

Regularization

In general, we can regularize by: $||\beta||^q$



Distributed — Sparse

When should we use a distributed (Ridge-like) and when a sparse (Lasso-like) model?

HTF p 72

Best of both worlds: Elastic net

$$J_{\text{ridge}} = \sum_{i=1}^{n} \left(y_i - \beta^{\mathsf{T}} \mathbf{x}_i\right)^2 + \lambda \left(\alpha \|\beta\|_1 + (1-\alpha)\|\beta\|_2^2\right)$$

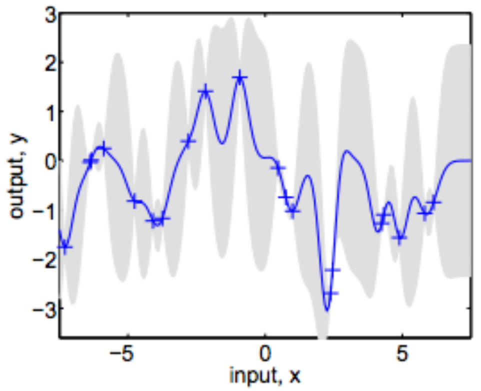
λ: Regularization strength

α: L1-ratio

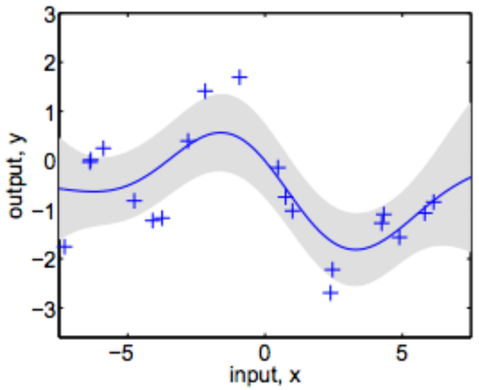
- Elastic net contains both Ridge ($\alpha = 0$) and Lasso ($\alpha = 1$) regression
- Both parameters can be tuned through cross-validation (grid search)

Advanced topic: Gaussian process regression

- Imaging we use a radial basis function for every data point: $\phi_i(x)$
- Because we use L_2 regularization, we still can estimate the model
- The width of the radial basis function will determine the smoothness of the fit



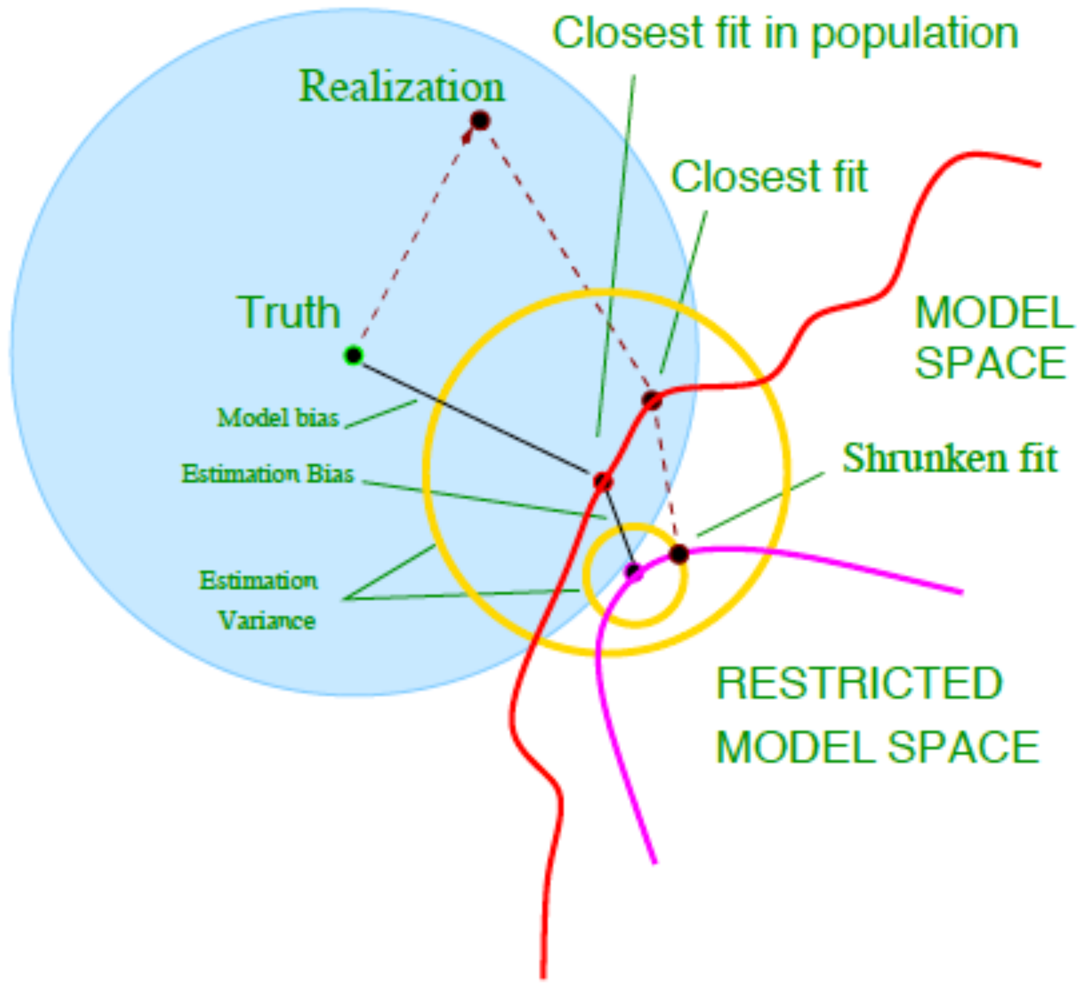
(b), $\ell = 0.3$



(c), $\ell = 3$

This, in a nutshell, is Gaussian Process Regression. I highly encourage you to read this book: Rassmussen & Williams: Gaussian Process regression for Machine Learning. http://www.gaussianprocess.org/gpml/chapters/RW.pdf

Regularization summary



HTF, p 225

Daily Summary

- Through feature expansion, we can explore large model spaces for linear models
- We can use model selection methods to find the best combination of features
- Regularization is a “soft” version of making models “simpler”: reduces variance at the cost of bias
- L2: Distributed solution L1: sparser solution
- Regularization parameters need to be “tuned” through validation set.
- Regularization can be viewed as a prior on parameters
- Combination of feature spaces and regularization leads to powerful machine learning methods (Gaussian processes, SVMs, Kernel PCA).