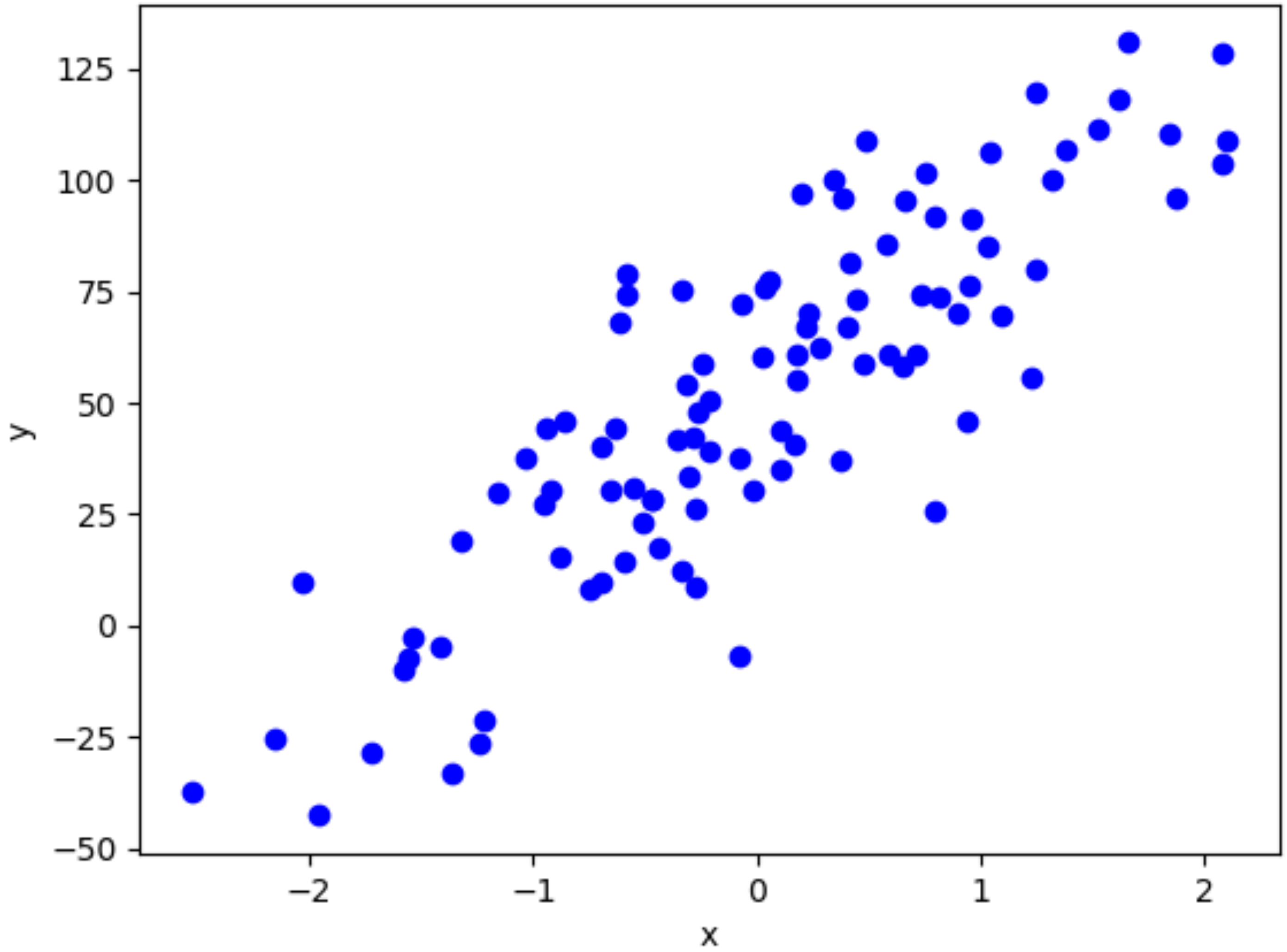
Linear Regression

Data Mining and Text Mining

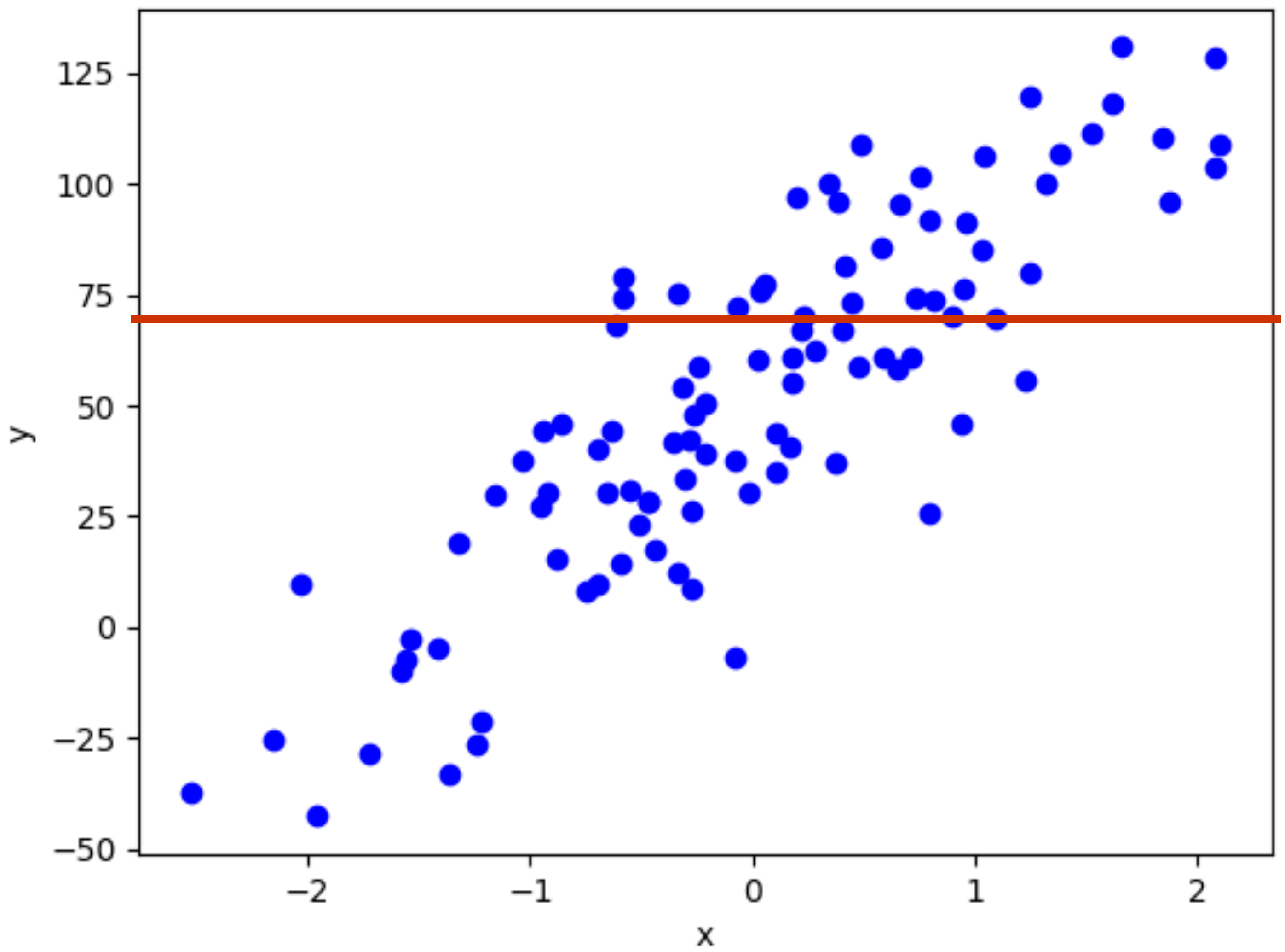
Simple Linear Regression



The training data points

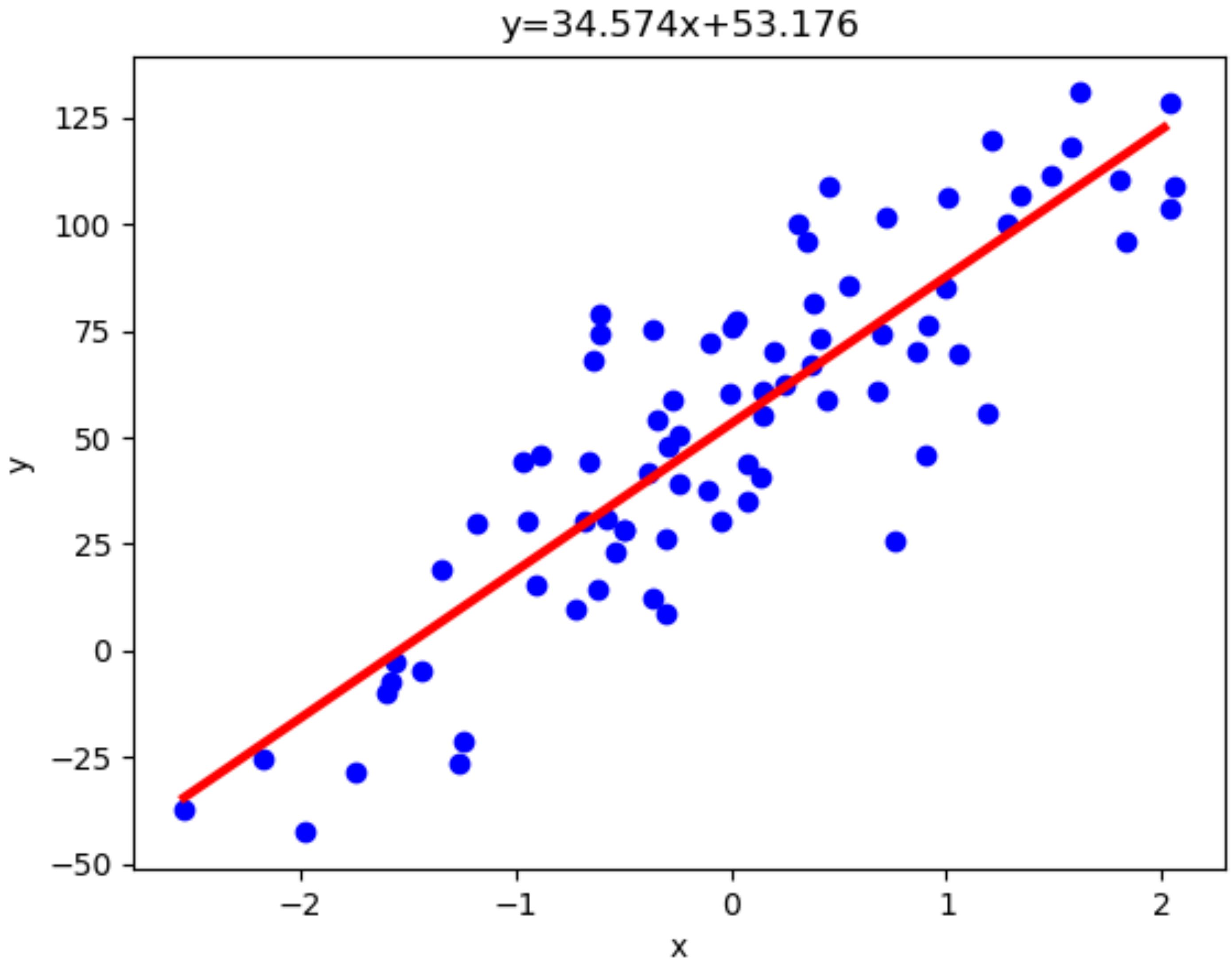
Can we predict the value of y from x?

What would be the simplest predictor?
Providing a baseline performance?



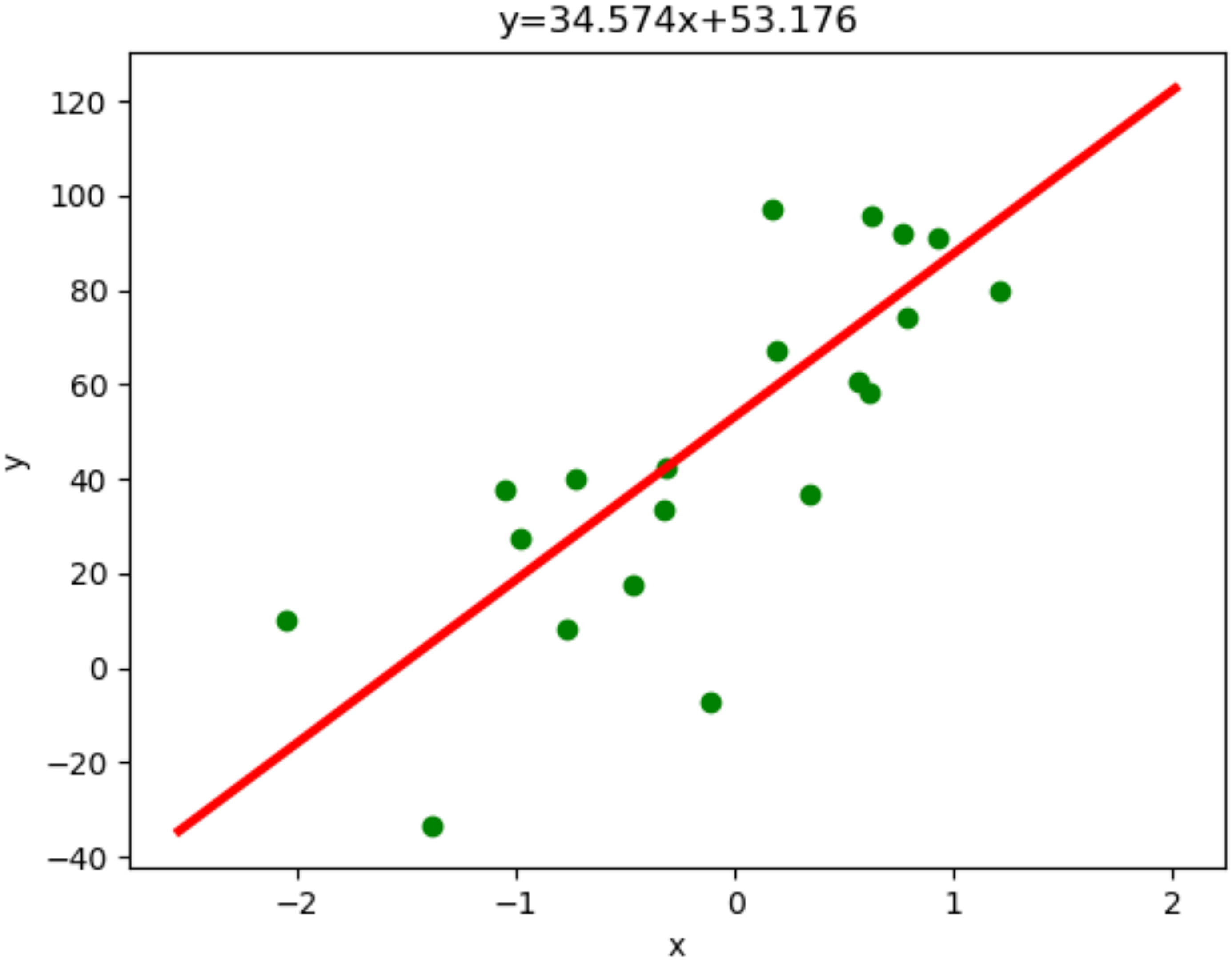
We might use basic average as a predictor. It is a very simple baseline, but it represents the simplest possible model and a value to evaluate other models.

Otherwise we can use
a simple linear model



Training data points with a simple linear regression model

But the model we built will not be used not on the same data …



The previous model applied to the model learned from the training data regression = model building + model usage

How Do We Evaluate a Regression Model?

- Given N examples, pairs x_i y_i, linear regression computes a model

$$y = w_0 + w_1 x$$

- So that for each point,

$$y_i = w_0 + w_1 x_i + \varepsilon_i$$

- We evaluate the model by computing the Residual Sum of Squares (RSS) computed as,

$$RSS(w_0, w_1) = \sum_{i=1}^{N} \varepsilon_i^2 = \sum_{i=1}^{N} \left(y_i - (w_0 + w_1 x_i)\right)^2$$

The goal of linear regression is thus to find the weights that minimize RSS

$$
\begin{aligned}
w_0, w_1 &= \arg\min_{w_0, w_1} RSS(w_0, w_1) \\
&= \arg\min_{w_0, w_1} \sum_{i=1}^{N} (y_i - (w_0 + w_1 x_i))^2
\end{aligned}
$$

How do we compute the best weights?

- Use gradient of RSS
- Gradient is vector of partial derivatives
- For simple linear regression, gradient has 2 components one for w_0 and one for w_1

$$\nabla RSS(w_0, w_1) = \begin{bmatrix} -2\sum_{i=1}^{N}(y_i - (w_0 + w_1 x_i)) \\ -2\sum_{i=1}^{N}(y_i - (w_0 + w_1 x_i))\,x_i \end{bmatrix}$$

How Do We Compute the Best Weights?

- Approach 1
  - Set the gradient of $RSS(w_0,w_1)$ to zero; but the approach is infeasible in practice

- Approach 2
  - Apply gradient descent

$$\text{while not converged}$$
$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta \nabla RSS(\vec{w}^{(t)})$$

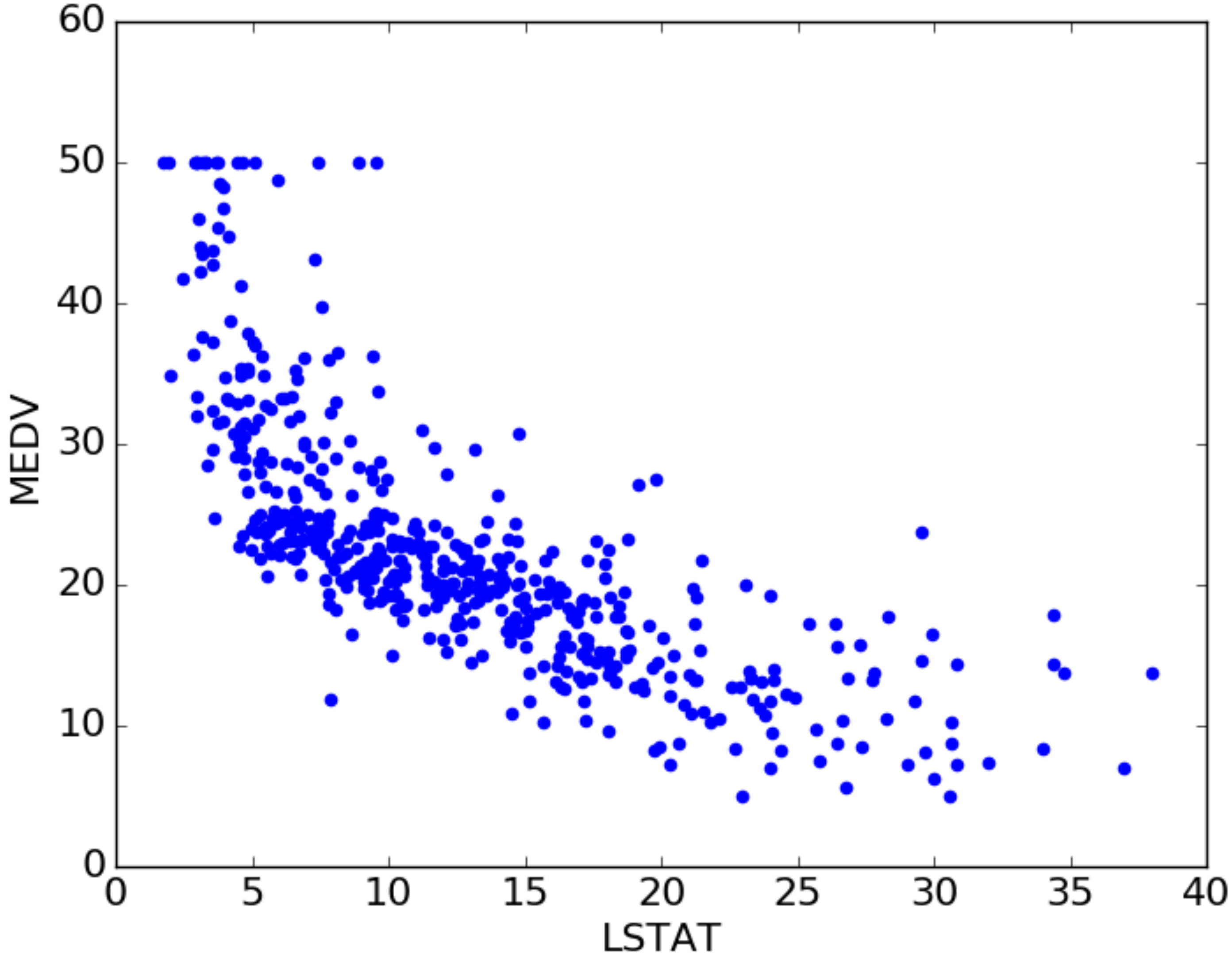
  - If η is large, we are making large steps but might not converge, if η is small, we might be very slow. Typically, η adapts over time, e.g., $\eta(t) = \alpha/t$ or $\alpha/\mathrm{sqrt}(t)$

The RSS Gradient

- For simple linear regression the gradient of RSS has only two components one for w_0 and one for w_1

$$\nabla RSS(w_0, w_1) = \begin{bmatrix} -2\sum_{i=1}^{N}(y_i - (w_0 + w_1 x_i)) \\ \\ -2\sum_{i=1}^{N}(y_i - (w_0 + w_1 x_i))\, x_i \end{bmatrix}$$

Multiple Linear Regression



Input variable: LSTAT - % lower status of the population

Output variable: MEDV - Median value of owner-occupied homes in $1000's

Can we predict the property value using other variables?

- Given a set of examples associating $LSTAT_i$ values to $MEDV_i$ values, simple linear regression find a function f(.) such that

$$MEDV_i = f(LSTAT_i) + \varepsilon_i$$

- Where $\boldsymbol{\varepsilon}_i$ is the error to be minimized
- Typically, we assume a model and fit the model into the data
- With linear model we assume that f(.) is computed as

$$f(LSTAT_i) = w_0 + w_1 \times LSTAT_i$$

- A polynomial model would fit the data points with a function,

$$f(LSTAT_i) = w_0 + \sum_{j=1}^{D} w_j \times {LSTAT_i}^j$$

Multiple Linear Regression

- Given, D input variables, assumes that the output y can be computed as,

$$y = w_0 + \sum_{j=1}^{D} w_j x_j + \varepsilon$$

- The model cost is computed using the residual sum of square, RSS(w) as,

$$RSS(\vec{w}) = \sum_{i=1}^{N} \left(y_i - w_0 - \sum_{j=1}^{D} w_j x_{i,j} \right)^2$$

Coefficient of Determination R^2

- Total sum of squares

$$TSS = \sum_{i=1}^{N} (y_i - \overline{y})^2$$

- Coefficient of determination

$$R^2 = 1 - \frac{RSS}{TSS}$$

- R^2 measures of how well the regression line approximates the real data points. When R^2 is 1, the regression line perfectly fits the data.

Multiple Linear Regression: General Formulation

- In general, given a set of input variables x, a set of N examples x_i, y_i and a set of D features h_j computed from the input variables x_i, multiple linear regression assumes a model,

$$y_i = \sum_{j=0}^{D} w_j h_j(\vec{x_i}) + \varepsilon_i$$

- $h_j(.)$ identify variables derived from the original inputs
- $h_j(.)$ could be the squared value of an existing variable, a trigonometric function, the age given the date of birth, etc.

Multiple Linear Regression: General Formulation

- Multiple linear regression aims at minimizing,

$$RSS(\vec{w}) = \sum_{I=1}^{N} \left(y_i - \sum_{j=0}^{D} w_j h_j(\vec{x}_i) \right)^2$$

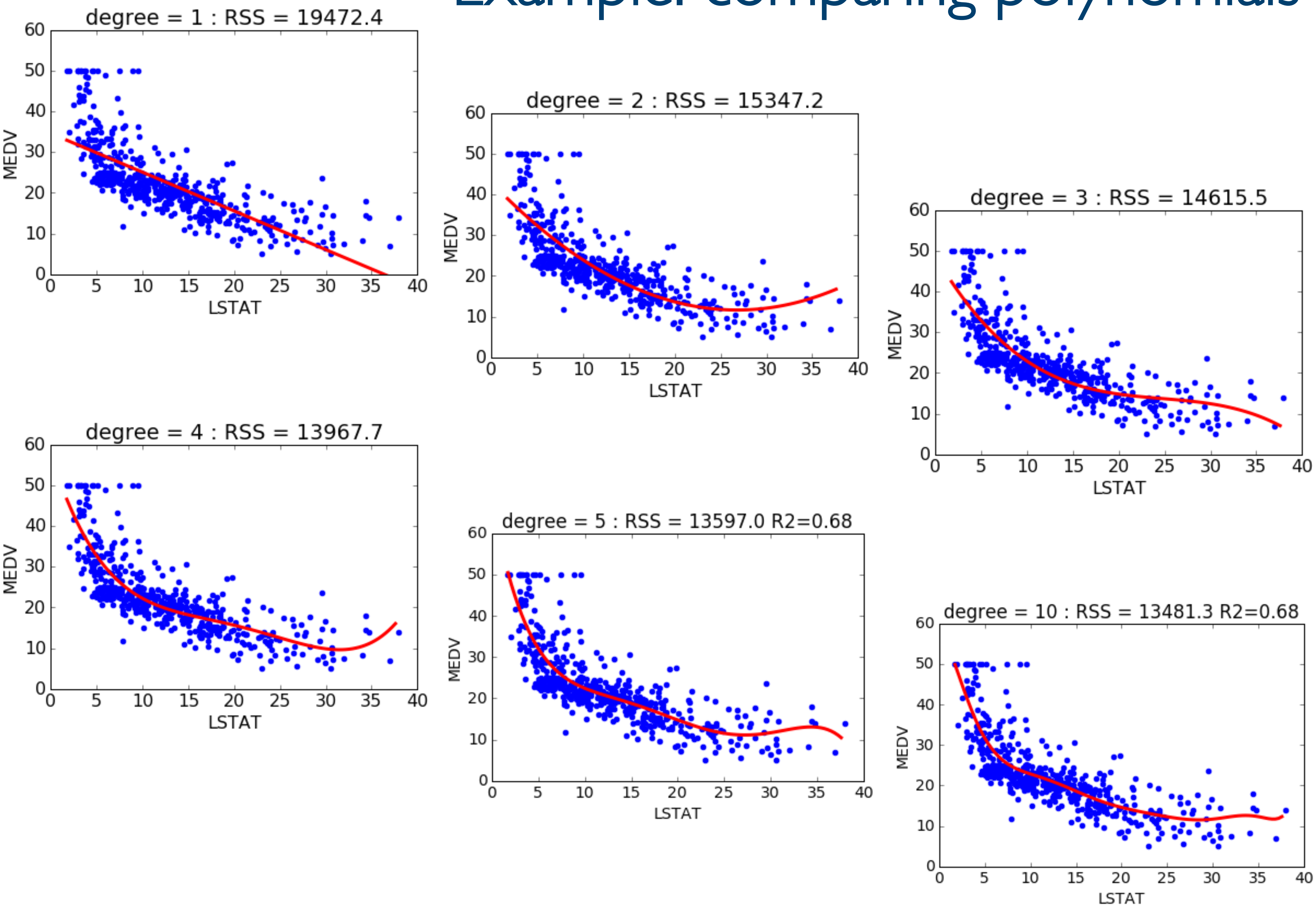
- For this purpose, it can apply gradient descent to update the weights as,

$$w_j^{(t+1)} = w_j^{(t)} + 2\eta \sum_{i=1}^{N} \underbrace{h_j(\vec{x}_i)}_{\text{feature value}} (\underbrace{y_i - \hat{y}_i(\vec{w}^{(t)})}_{\text{residual error}})$$

Multiple Linear Regression with Gradient Descent

$$
\begin{aligned}
& t = 0 \\
& \text{init } w^{(0)} \\
& \text{while not converged} \\
& \qquad \text{for } (j = 0;\ j \leq D;\ j = j + 1) \\
& \qquad\qquad \Delta w_j = -2 \sum_{i=1}^{N} h_j(\vec{x}_i)(y_i - \hat{y}_i(\vec{w}^{(t)})) \\
& \qquad\qquad w_j^{(t+1)} = w_j^{(t)} - \eta \Delta w_j \\
& \qquad t = t + 1
\end{aligned}
$$

Example: comparing polynomials



Model Evaluation

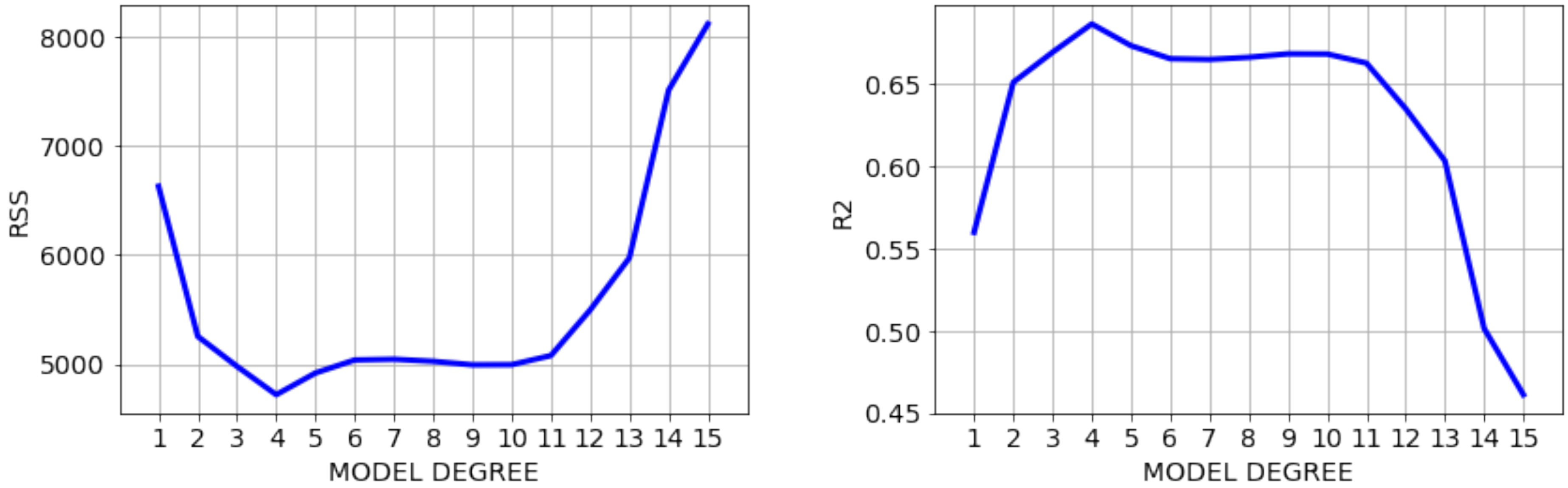
- Models should be evaluated using data that have not been used to build the model itself
- For example, would be feasible to evaluate students using exactly the same problems solved in class?
- The available data must be split between training and test
  - Training data will be used to build the model
  - Test data will be used to evaluate the model performance

Holdout Evaluation

- Reserves a certain amount for testing and uses the remainder for training
  - Too small training sets might result in poor weight estimation
  - Too small test sets might result in a poor estimation of future performance

- Typically,
  - Reserve ½ for training and ½ for testing
  - Reserve 2/3 for training and 1/3 for testing

- For small or "unbalanced" datasets, samples might not be representative



Given the original dataset, we split the data into 2/3 train and 1/3 test and then apply multiple linear regression using polynomials of increasing degree. The plot show how RSS and R^2 vary.

Holdout Evaluation using the Housing Data

- Given the original dataset, we split the data into 2/3 train and 1/3 test and then apply multiple linear regression using polynomials of increasing degree.

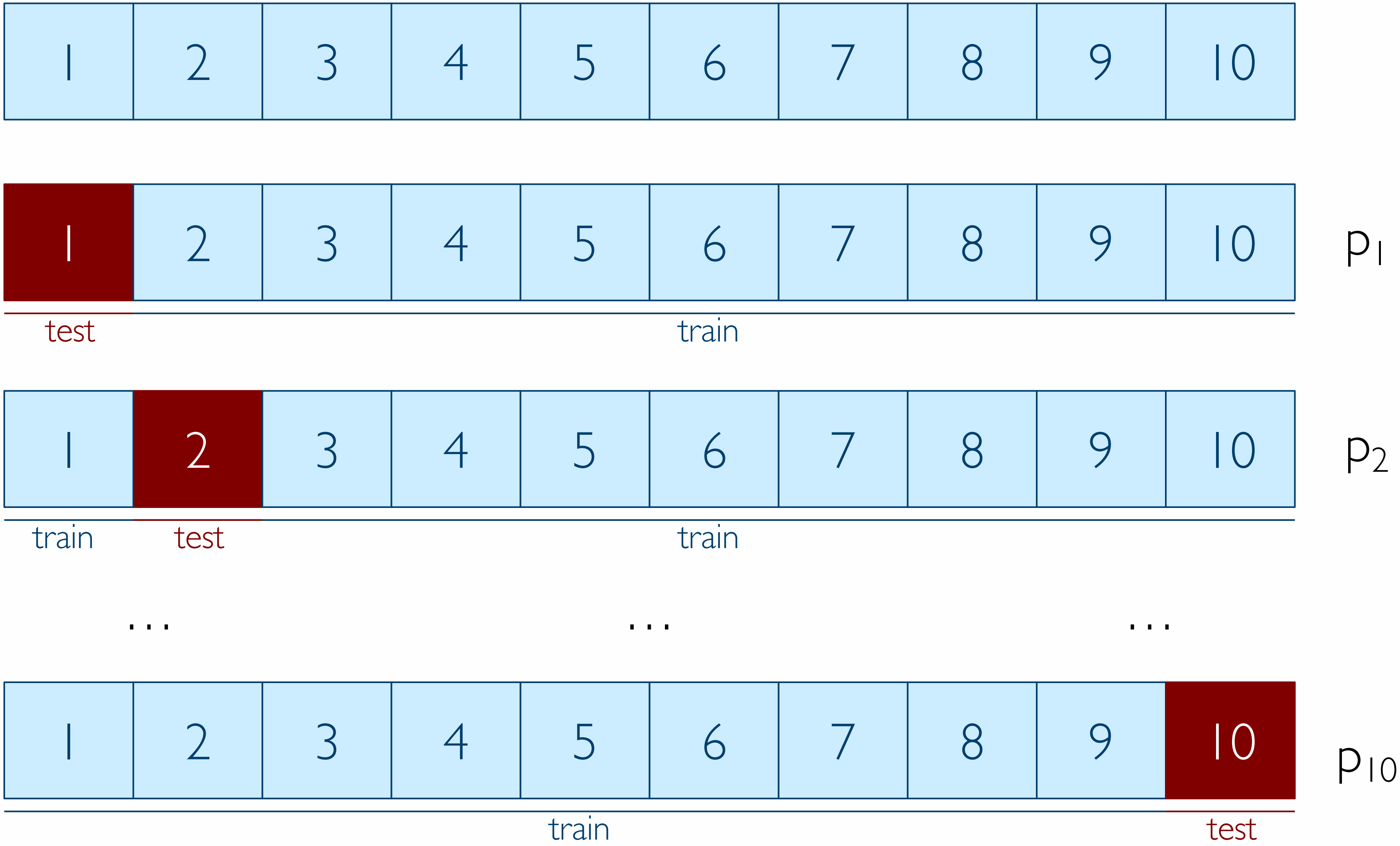
- RSS initially decreases as polynomials better approximate the data but then higher degree polynomials overfit

- The same is shown by the R^2 statistics

Cross-Validation

- First step
  - Data is split into k subsets of equal size
- Second step
  - Each subset in turn is used for testing and the remainder for training
- This is called k-fold cross-validation and avoids overlapping test sets
- Often the subsets are stratified before cross-validation is performed
- The error estimates are averaged to yield an overall error estimate

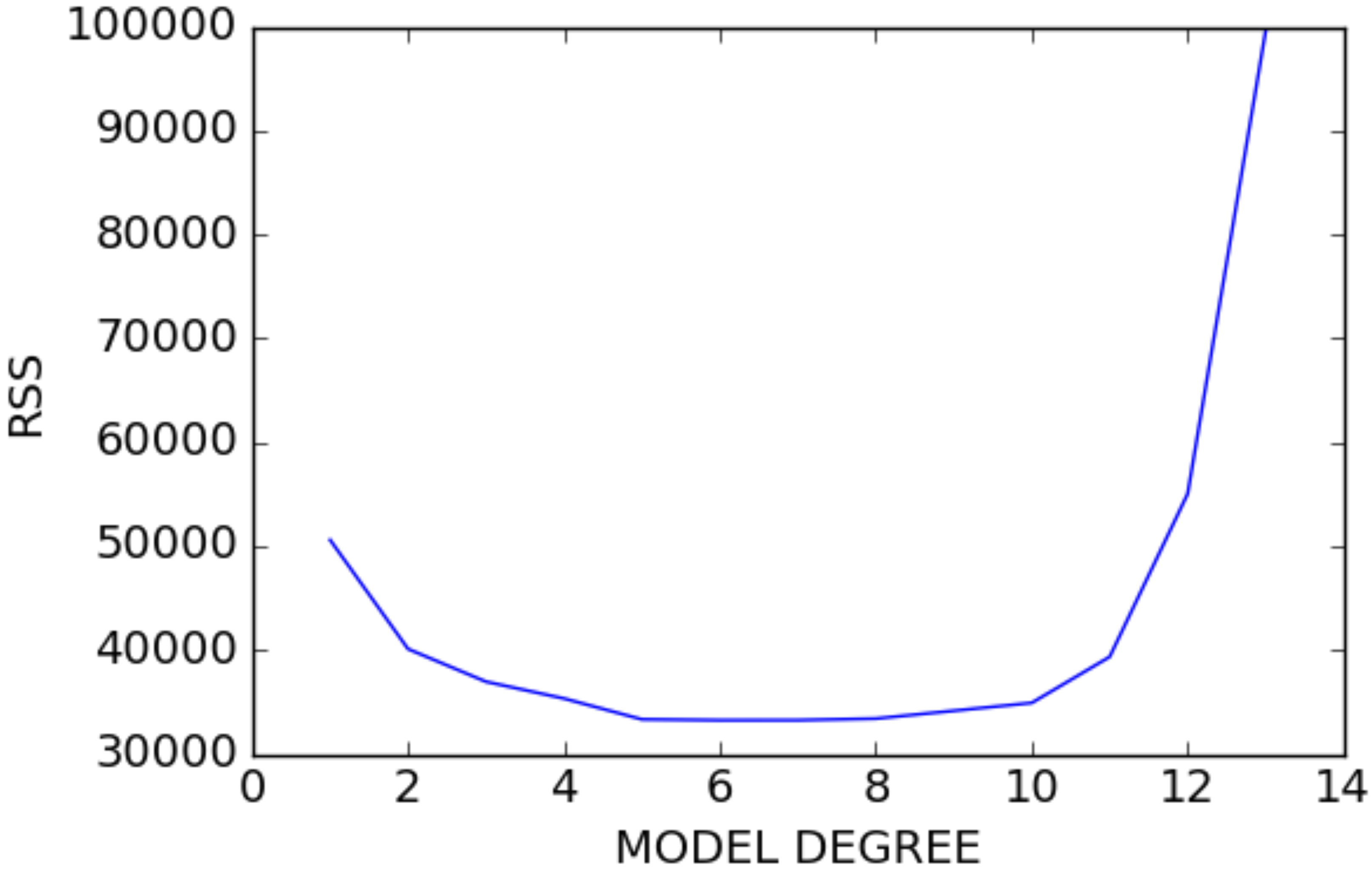
Ten-fold Crossvalidation



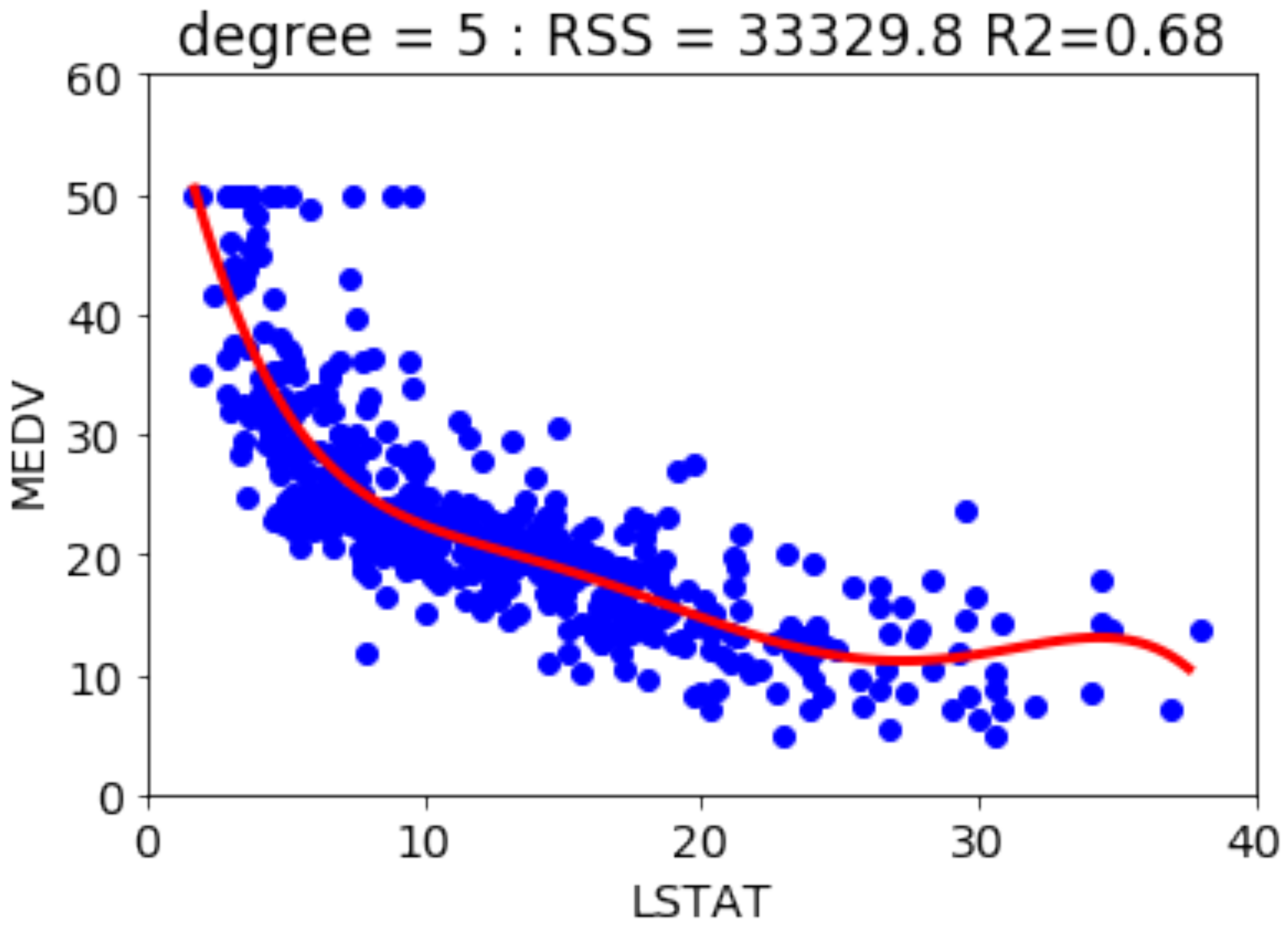
The final performance is computed as the average p_i

Cross-Validation

- Standard method for evaluation stratified ten-fold cross-validation
- Why ten? Extensive experiments have shown that this is the best choice to get an accurate estimate
- Stratification reduces the estimate's variance
- Even better: repeated stratified cross-validation
- E.g. ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)
- Other approaches appear to be robust, e.g., 5x2 crossvalidation



As we increase the degree of the fitting polynomial, the crossvalidation error starts to increase, because the model the model starts to overfit! Best performance for the 5^{th} degree polynomial.



Fitting using the 5th degree polynomial

Overfitting

What is Overfitting?

Very good performance on the training set

(model fits precisely patterns present in training data)

Terrible performance on the test set

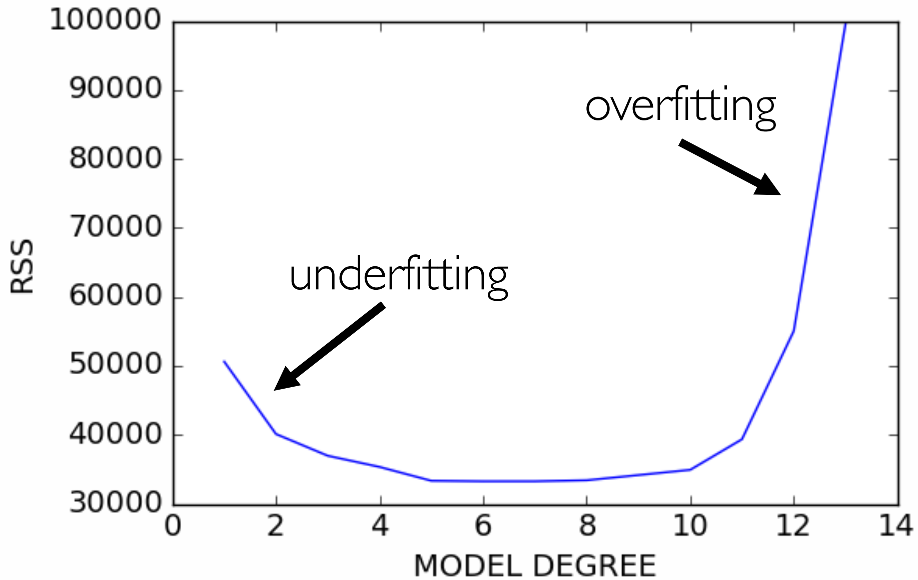
(patterns were just noise and are no longer present)

Underfitting vs Overfitting

- Having too few parameters leads to underfitting
  - Model isn't powerful enough to describe data

- While too many parameters leads to overfitting
  - Model is too powerful

- By increasing degree of polynomial
  - First see improvements in cross-validation error
  - Then error starts to increase again as model starts to overfit



- At the end, best performance for 5th degree polynomial

In regression, overfitting is often associated
to large weights estimates

Add to the usual cost (RSS) a term
to penalize large weights to avoid overfitting

Total cost = Measure of Fit + Magnitude of Coefficients

Ridge Regression (L_2 Regularization)

- Minimizes the cost function,

$$
\begin{aligned}
Cost(\vec{w}) =& RSS(\vec{w}) + \alpha||\vec{w}||_2^2 \\
=& \sum_{i=1}^{N}\left(y_i - \sum_{j=0}^{D} w_j h_j(x_i)\right)^2 + \alpha \sum_{j=0}^{D} w_j^2
\end{aligned}
$$

- If α is zero, the cost is exactly the same as before; if α is infinite, then the only solution corresponds to having all the weights to 0

- In the gradient descent algorithm the update for weight j becomes,

$$
\Delta w_j = -2\sum_{i=1}^{N} h_j(\vec{x}_i)(y_i - \hat{y}_i(\vec{w}^{(t)})) + 2\alpha w_j
$$

Lasso Regression (L_1 Regularization)

- Minimizes the cost function,

$$
\begin{aligned}
Cost(\vec{w}) =& RSS(\vec{w}) + \alpha||\vec{w}||_1 \\
=& \sum_{i=1}^{N}\left(y_i - \sum_{j=0}^{D} w_j h_j(x_i)\right)^2 + \alpha \sum_{j=0}^{D}|w_j|
\end{aligned}
$$

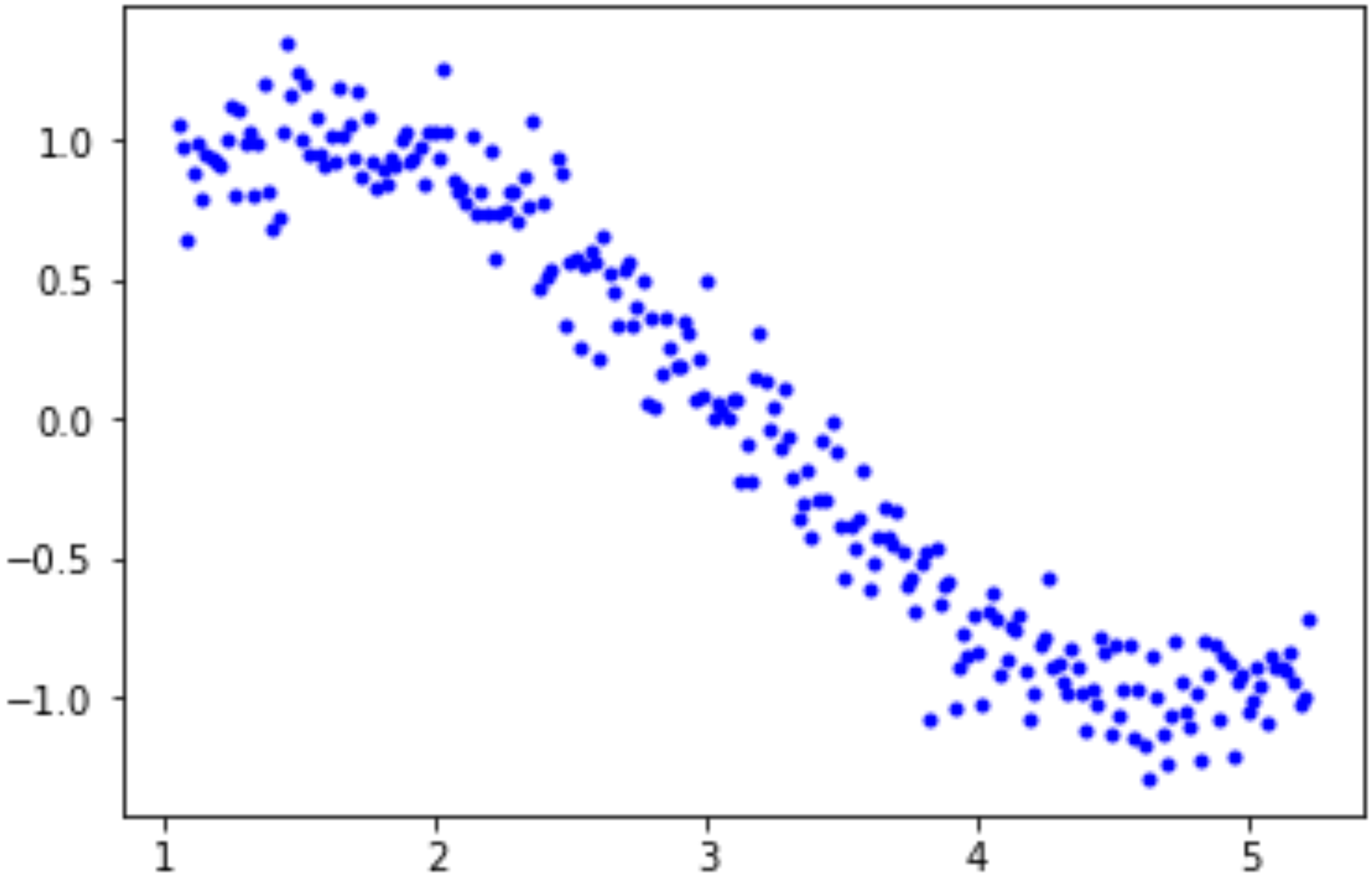
- If α is zero, the cost is exactly the same as before; if α is infinite, then the only solution corresponds to having all the weights to 0
- In gradient descent, weight j is modified as,

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

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

$$
w_j = \begin{cases} (\rho_j + \alpha/2)/z_j & \texttt{if } \rho_j < -\alpha/2 \\ 0 & \texttt{if } \rho_j \in [-\alpha/2, \alpha/2] \\ (\rho_j - \alpha/2)/z_j & \texttt{if } \rho_j > \alpha/2 \end{cases}
$$

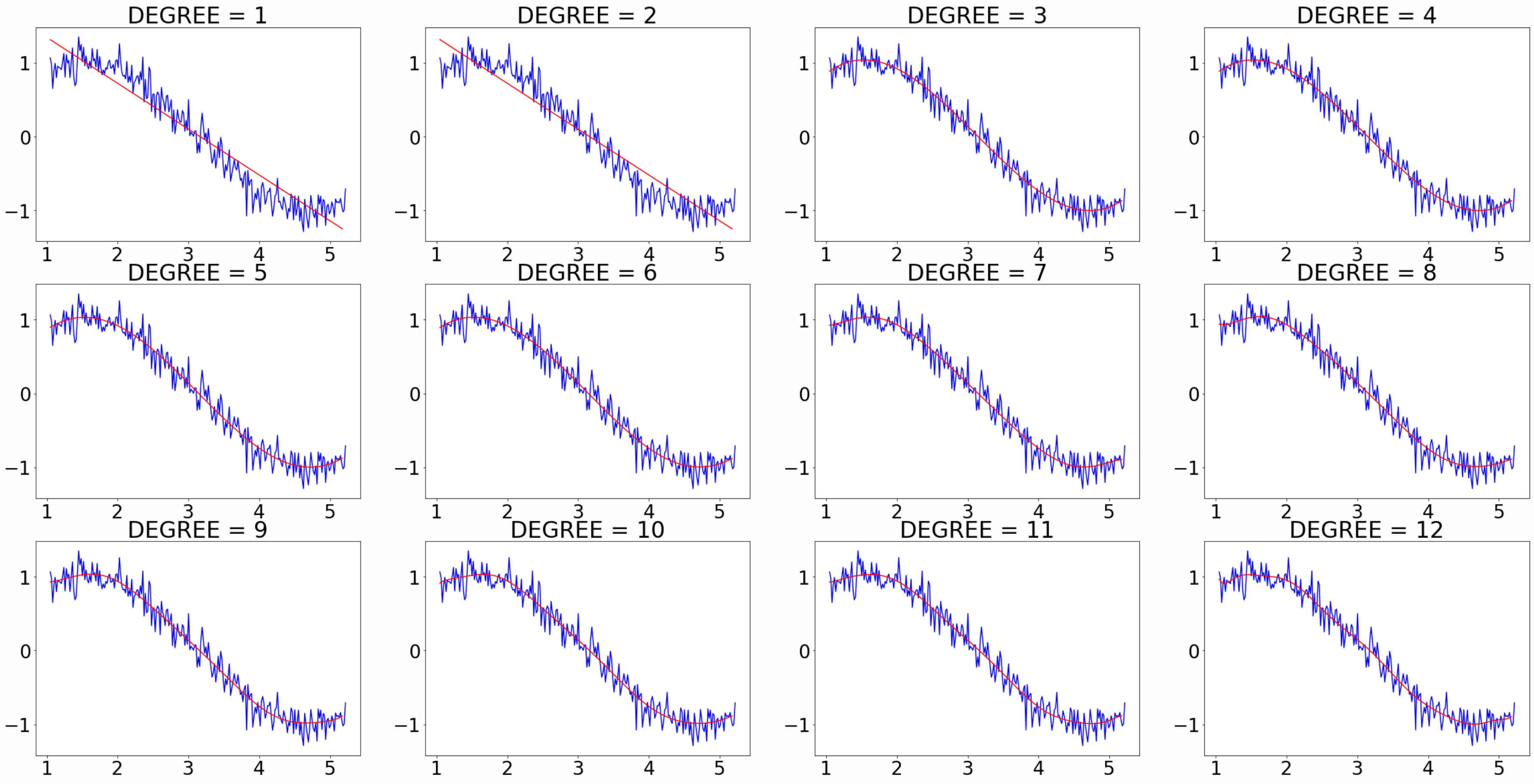
Comparing No Regularization with Ridge Regression and Lasso



Simple example data generated using a trigonometric function.

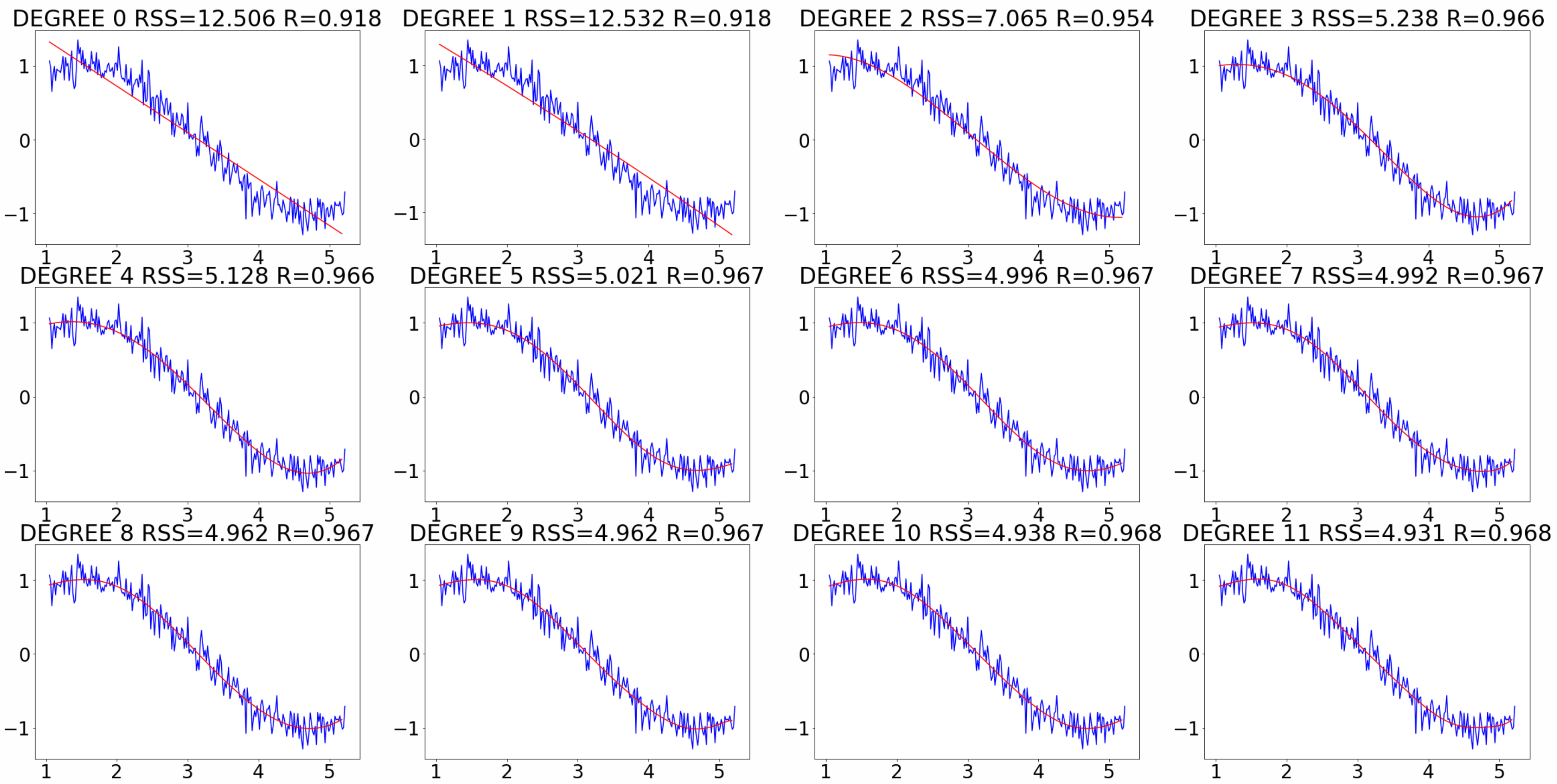
Unregularized Case

- Applying multiple linear regression (to training data)



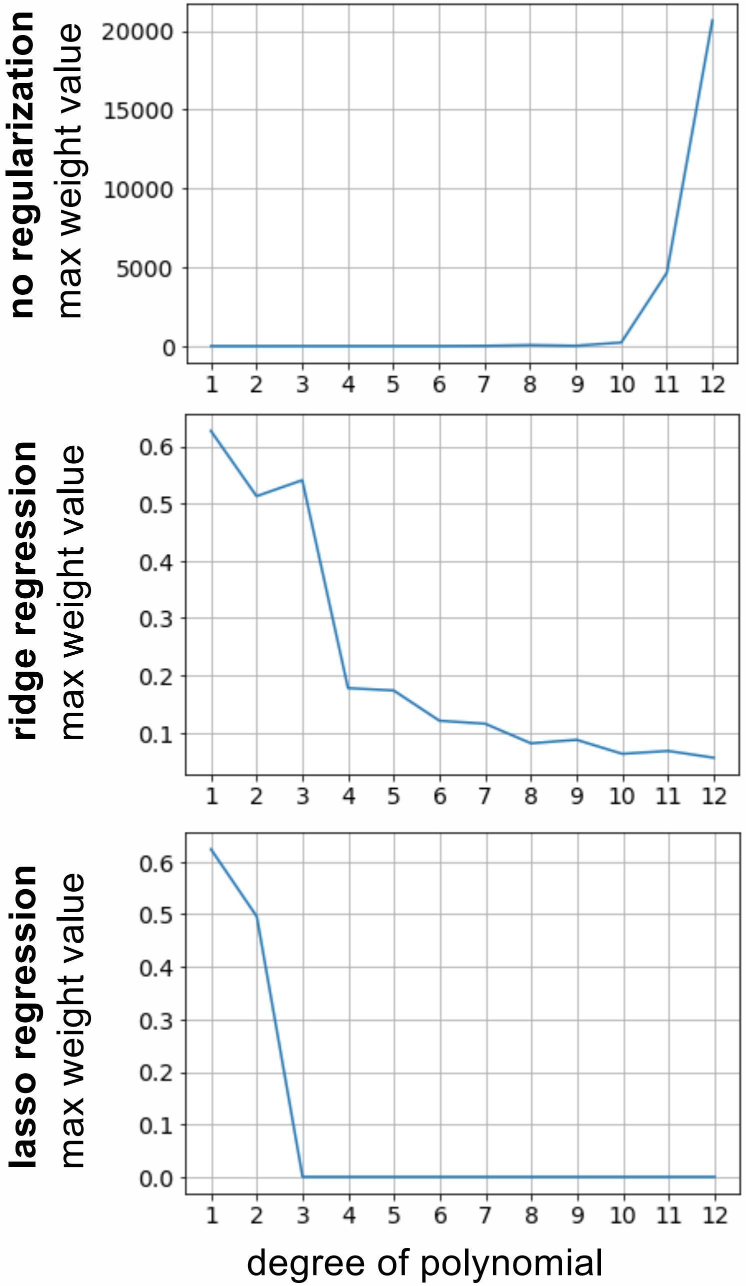
Regularized Case (Ridge Regression)

- Cubic model doesn't quite fit the data quite as well as before



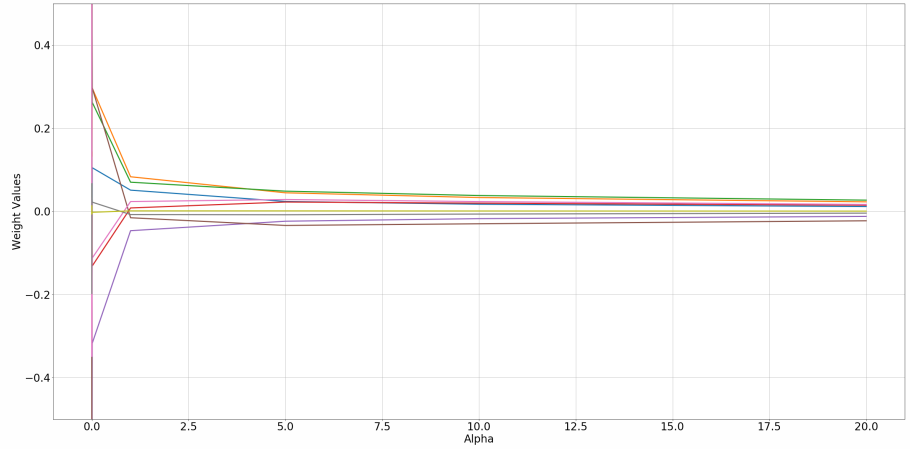
Comparing parameters

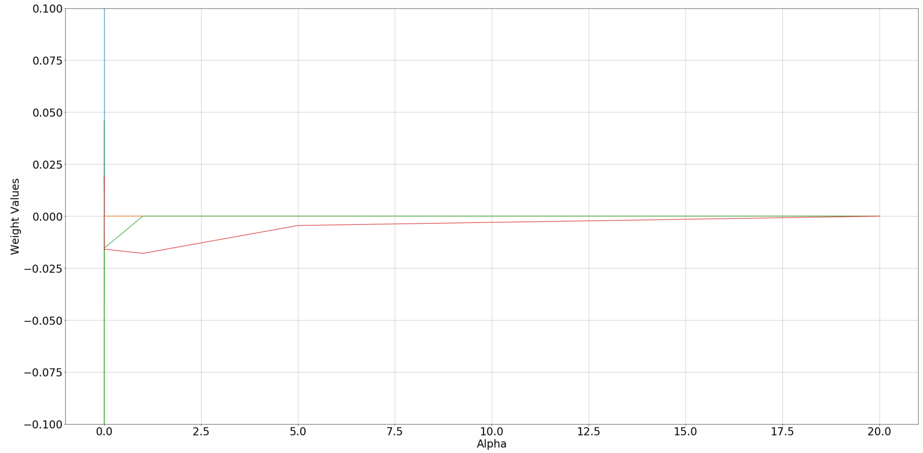
- Plots show largest (absolute) value of across the weight parameters
  - For polynomials of certain degree
  - For each regularization technique: top: none, middle: ridge, bottom: lasso
- Note the massive scale for first graph compared to other graphs (max value of 20,000 vs 0.6)
- In the unregularized case, massive values occur for high order polynomials indicates overfitting



Effect of regularization parameter

- Plots showing effect of modifying the amount of regularization for ridge (top) & lasso (bottom)
- Each coloured line on plot is different weight parameter of 10 degree polynomial
- As **α** increases,
  - For ridge, none of weights ever go to zero
  - For lasso, almost all of parameters are forced to zero





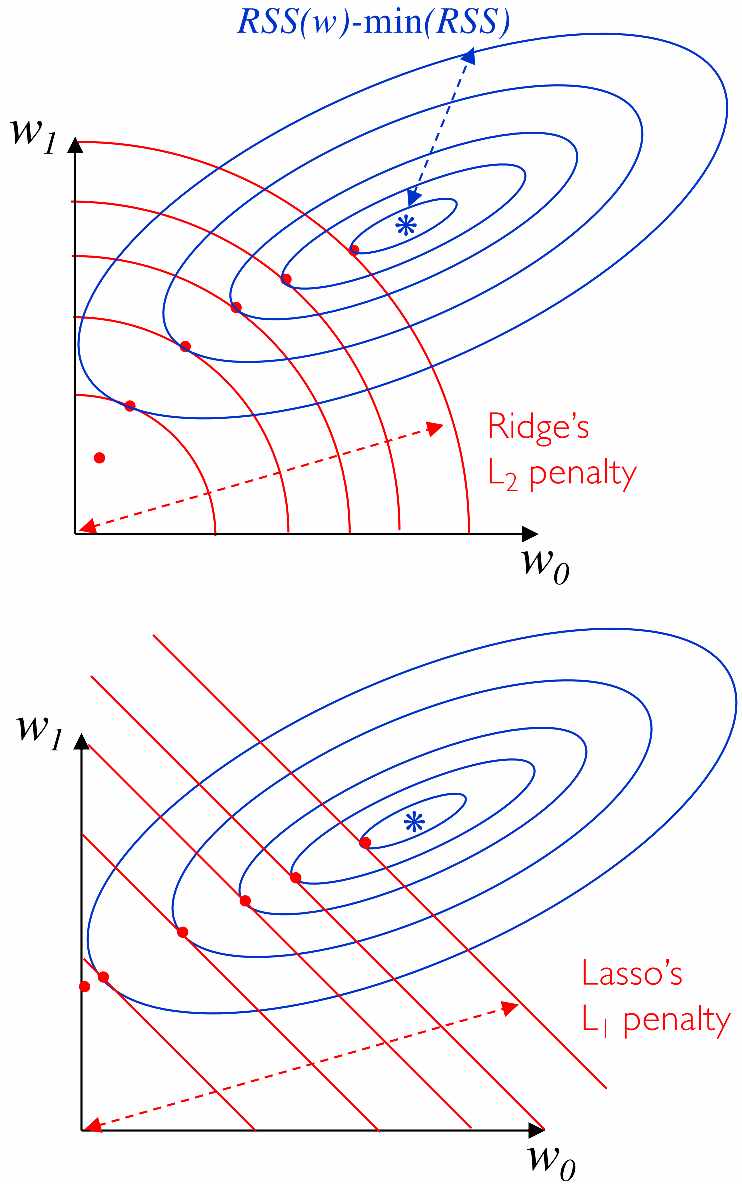
Amount of regularization: value of hyper-parameter, **α**

Lasso tends to zero out less important features and produces sparser solutions

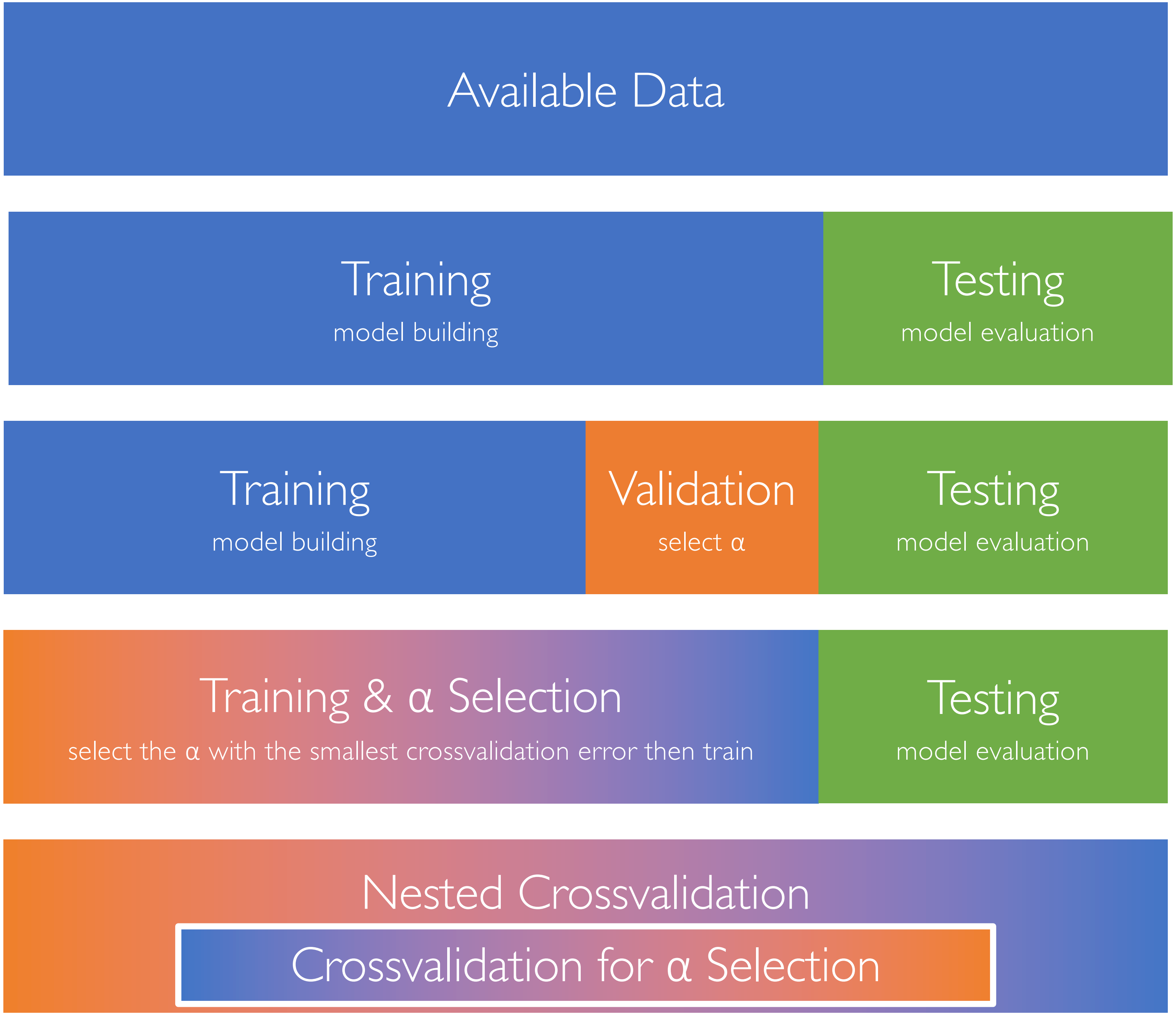
Basically, by penalizing large weights it also performs feature selection

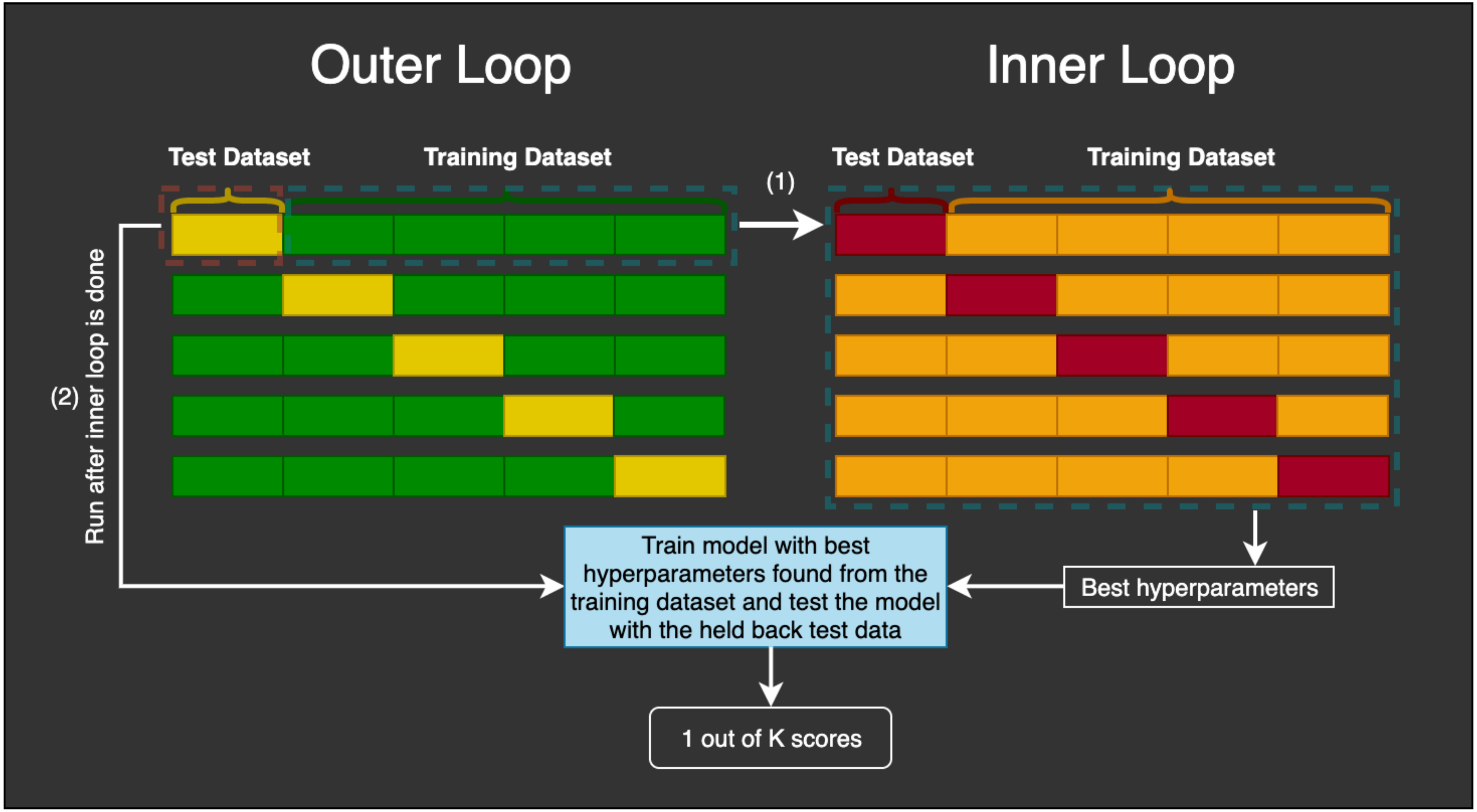
Explaining Lasso's Sparsity Effect

- Graphs show level sets (lines of equal value) for
  - RSS loss function in blue and
  - Ridge and Lasso penalties in red
- As we increase alpha
  - Optimal solution to penalized loss function moves back toward the origin (red dots)
  - For lasso it first heads to an axis, removing the weight altogether
  - For ridge, the optimal point never reaches the axis



Hyperparameter Optimization: Choosing α


Available Data
Training
model building
Testing
model evaluation
Training
model building
Validation
select α
Testing
model evaluation
Training & α Selection
select the α with the smallest crossvalidation error then train
Testing
model evaluation
Nested Crossvalidation
Crossvalidation for α Selection



https://mlfromscratch.com/nested-cross-validation-python-code/

The validation set is also called “development set”

Because the set is used to develop the model before production

Selecting the Best α

- To select the best value of α we cannot use the test set since it is going to be used for evaluating the final model (which uses α)

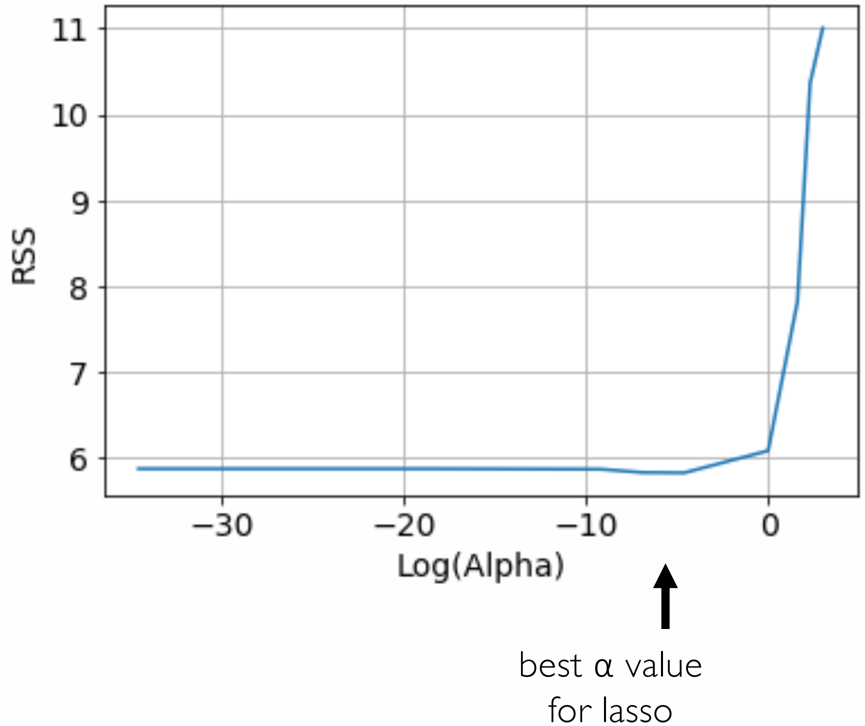
- Need to reserve part of the training data to evaluate possible candidate values of α and to select the best one

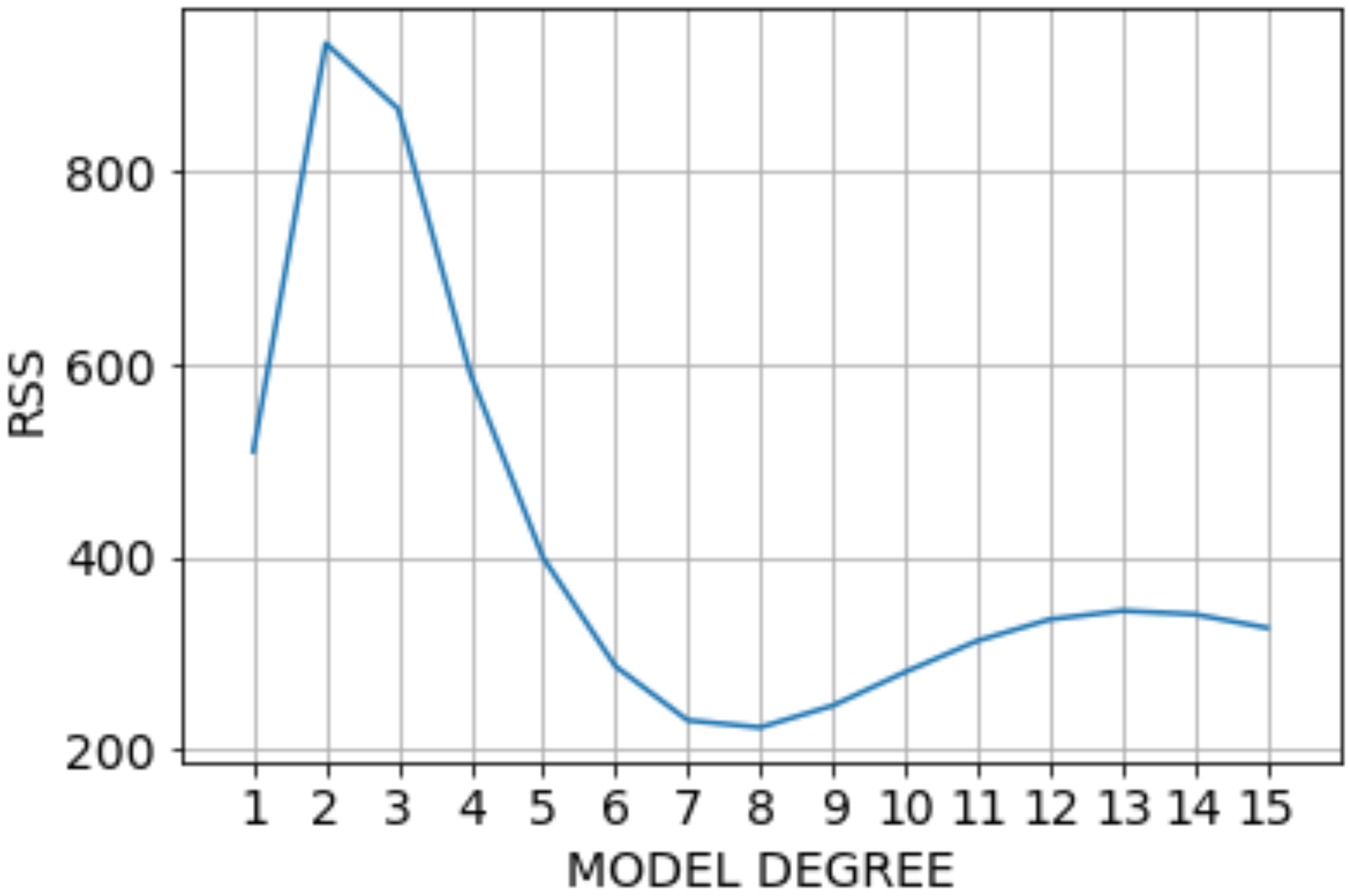
- If we have enough data, we can extract a validation set from the training data which will be used to select α

- If we don't have enough data, we should select α by applying k-fold cross-validation over the training data choosing the α corresponding to the lowest average cost over the k folds

Selecting the Best α

- To select best value of α
  - cannot use test set since it will be used for evaluating final model (which uses α)
  - need to reserve part of training data to evaluate candidates and select best value
- If have enough data, extract validation set from training data to select α
- If don't have enough data
  - Select α by applying k-fold cross-validation over training data
  - Choose α corresponding to lowest average cost over k folds





Applying Lasso with a α of 0.01 with different polynomials

Summary

Linear Regression Algorithms

- The goal is to minimize the residual sum of squares (RSS)
- Exact methods
  - Compute the set of weights that minimizes RSS
- Gradient descent (batch and stochastic)
  - Start with a random set of weights and update them based on the direction that minimizes RSS
- Ridge regression/Lasso
  - Compute the cost also using the magnitude of the coefficients
  - The larger the coefficients the more likely we are overfitting