3
Hyperparameters and Regularization

Amal Aboulhassan

Logistics

- Homework 1

Logistics

- Homework 1: Extension to Sunday 23rd
- Year Work (50 degrees):
  - Homeworks: 10 marks
  - Quizzes: 10 marks
  - Midterm: 20 marks
  - Final Project: 10 marks
  - Attendance:
- Final exam: 50 marks

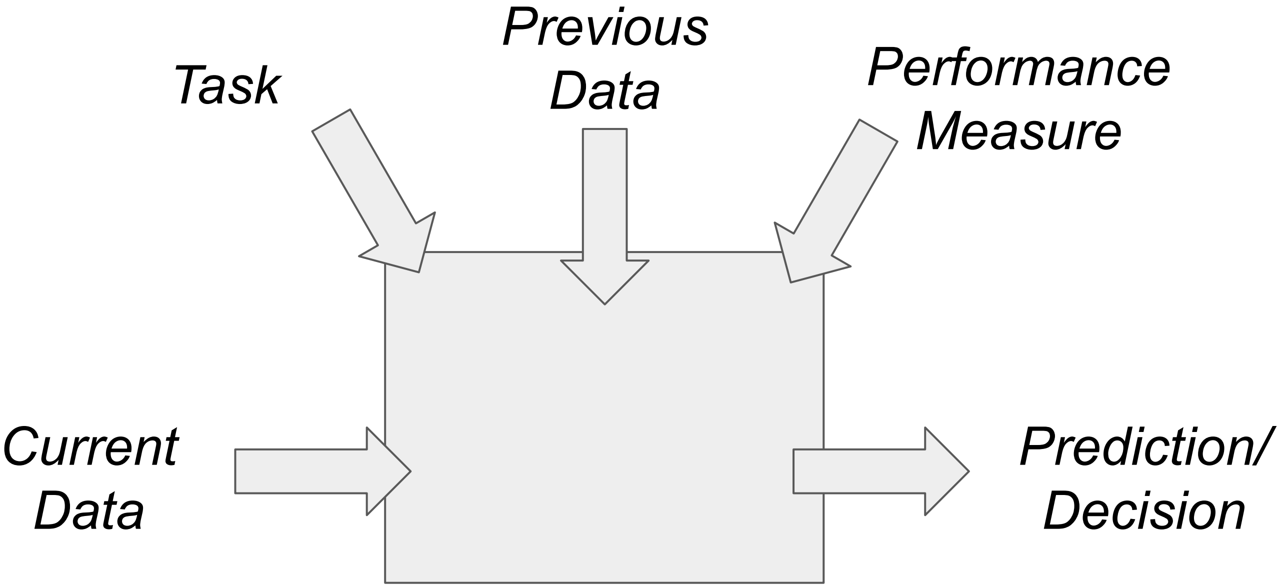
Agenda

- Overfitting/underfitting reasons
- Overfitting/underfitting solutions
- Feature transform definition
- Hyperparameter definition
- Hyperparameter selection
- Generalization error definition
- Generalization error measurement
    - Fixed Validation set
    - Cross Validation

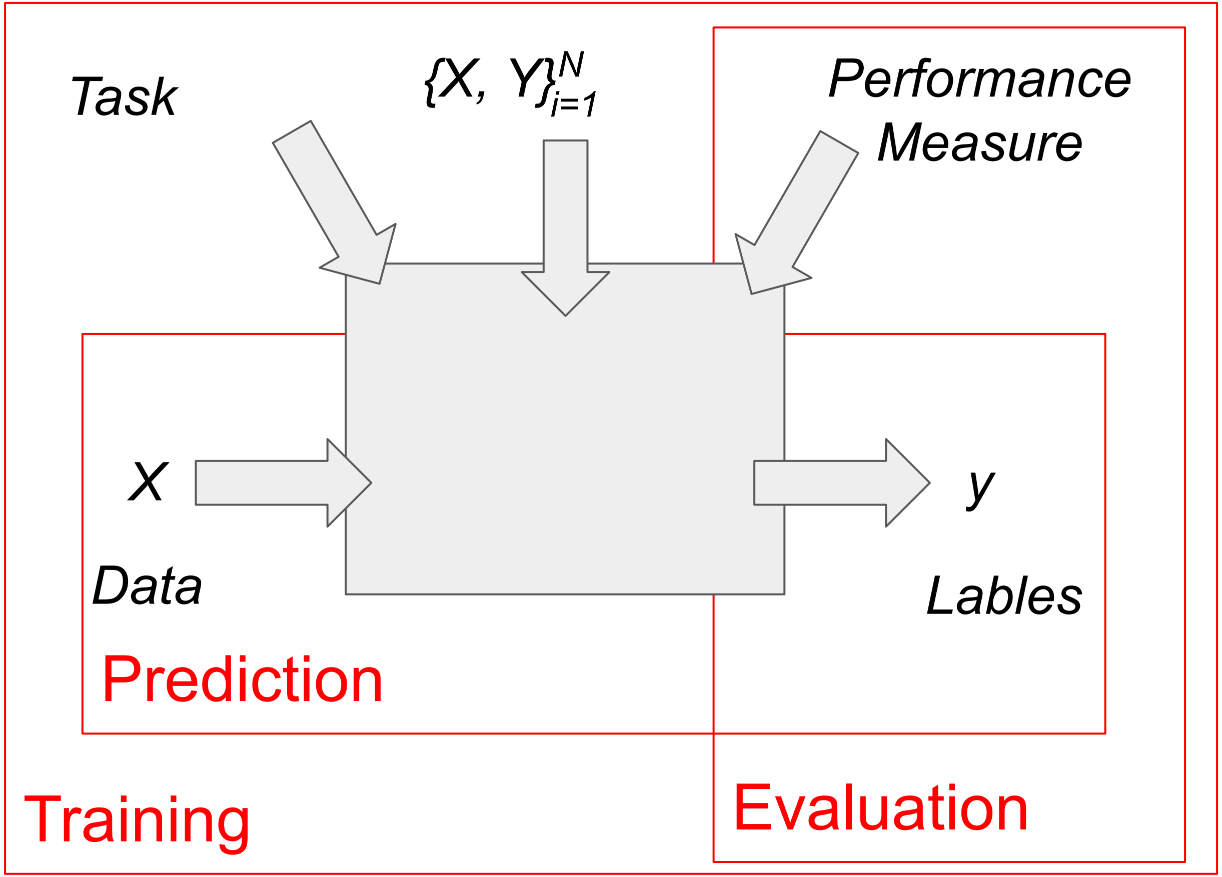
Machine Learning Taxonomy

- Supervised Learning
  - Regression
  - Classification
- Unsupervised Learning
  - Clustering
  - Others
- Reinforcement Learning

Machine Learning Process



Supervised Learning Process



Last Lecture

- Linear regression has been around since more than 200 years.
- Linear regression is a linear model y can be calculated from a linear combination of the input variables (x).
- When there is:
  - single input variable (x), the method is referred to as simple linear regression.
  - multiple input variables, the method is referred to as multiple linear regression.
- Different techniques can be used to prepare or train the linear regression equation from data:
  - Ordinary Least Squares (or Linear Regression or just Least Squares Regression).
  - Gradient Descent
  - Regularization

Last Lecture

- Training “least squares” linear regression
  - 1-dim. features without intercept
  - 1-dim. features with intercept
  - General case: Many features with intercept
  - Note: bias is another name for intercept

Last Lecture

- Linear Regression: (1) Least Squares
  - Task: Training
- Training Data:
  - X: Features
  - Y: Prediction/Labels/Response
- Model Function: Straight line
- Cost Function: Sum of Squared Errors
- Error:
  - Distance between two points observation y and prediction /or multidimensional
- Learning Algorithm: Linear Least Square
  - Output Model: values of w and b which minimize the cost function on the training set
  - Multidimension: values of θ compact form

Last Lecture

- Least Squares Training
  - Closed form
  - Gradient Descent

Linear Regression: (1) Least Squares

- Closed form (single dimension)

$$w = \frac{\sum_{n=1}^{N}(x_n - \bar{x})(y_n - \bar{y})}{\sum_{n=1}^{N}(x_n - \bar{x})^2}$$

$$b = \bar{y} - w\bar{x}$$

$$\bar{x} = \text{mean}(x_1, \dots x_N)$$

$$\bar{y} = \text{mean}(y_1, \dots y_N)$$

(1) Least Squares: F-dim Features

- Closed Form (Multidimension)

$$\theta = [b \ w_1 \ w_2 \ldots w_F]$$
$$\tilde{x}_n = [1 \ x_{n1} \ x_{n2} \ldots x_{nF}]$$
$$\hat{y}(x_n, \theta) = \theta^T \tilde{x}_n$$

$$J(\theta) \triangleq \sum_{n=1}^{N} (y_n - \hat{y}(x_n, \theta))^2$$

Last Lecture

- Gradient Descent
    - Cost function is always convex
    - Iterations over cost function
    - Decay value
    - Stopping criteria

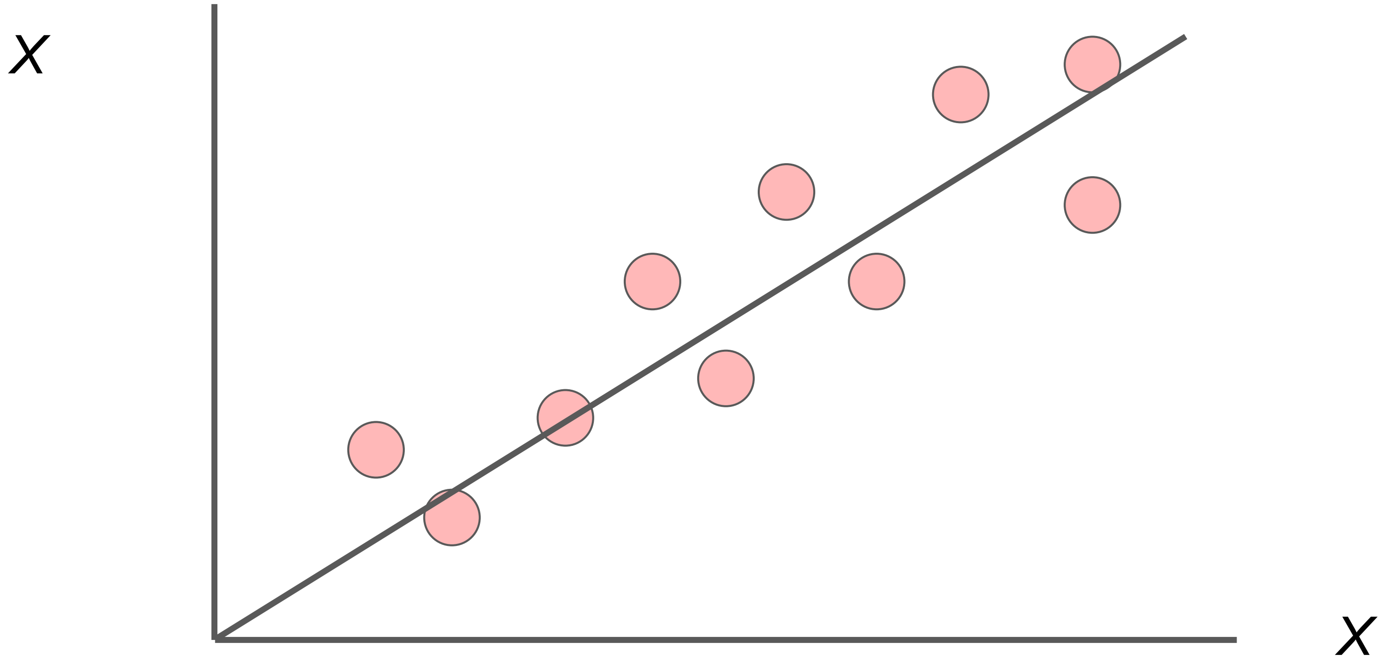
Today

- Overfitting/underfitting reasons
- Overfitting/underfitting solutions
- Hyperparameter definition
- Hyperparameter selection
- Generalization error definition
- Generalization error measurement
  - Fixed Validation set
  - Cross Validation

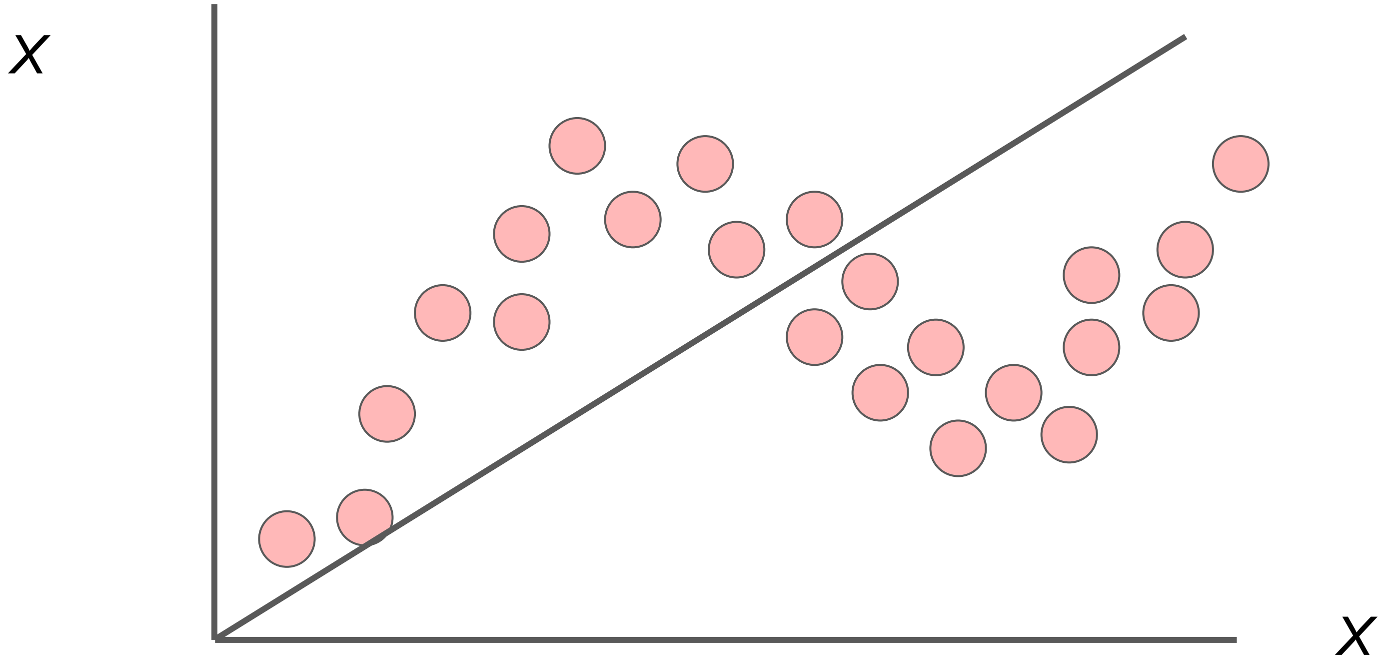
Polynomial Degree

In [11]:
```
from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
```

Example

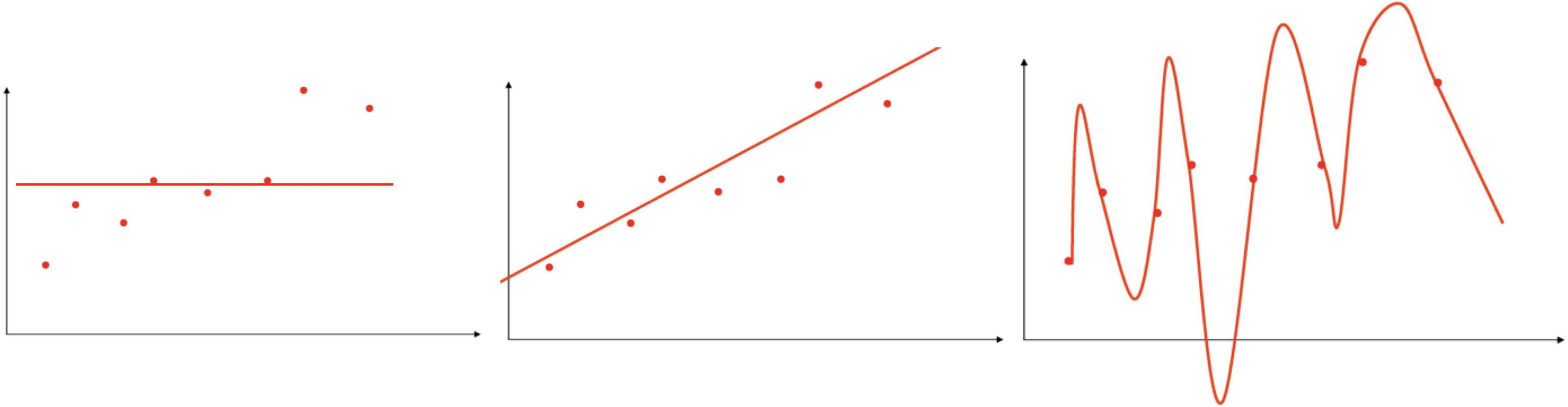


Example



Overfitting/Underfitting

y



X

Polynomial Degree

- What is the difference between linear and nonlinear functions?

Polynomial Degree

- Linear Functions:
  - One degree
  - Plotted as a straight line
- Non-linear Functions
  - 2 or more degrees
  - Plotted as a curve

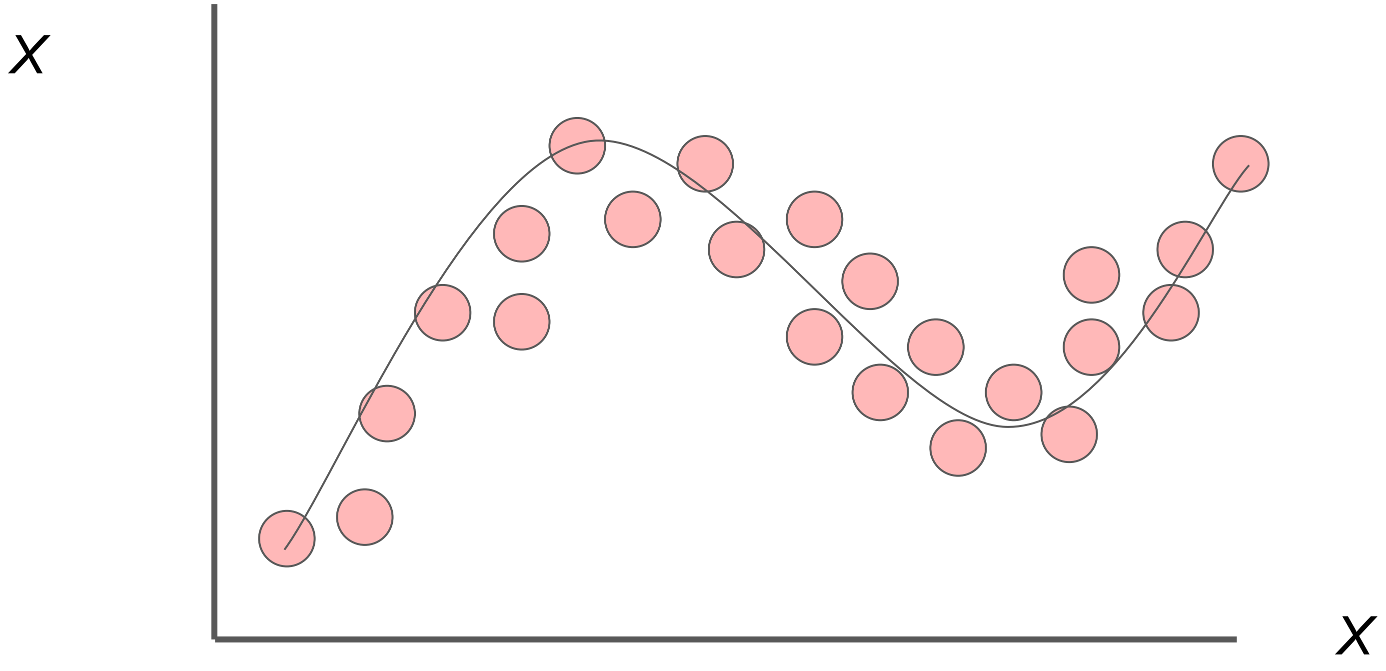
Polynomial Degree

- Polynomial

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,$$

where leading coefficient $a_n \neq 0$, , the a_i are real numbers and n is a nonnegative integer. Linear and quadratic functions are polynomials of degree 1 and 2, respectively; cubic and quartic polynomials are of degrees 3 and 4, respectively

Example



Feature Transform To Non-Linear Functions

- sin / cos for periodic data
- polynomials for high-order dependencies

$$\phi(x_i) = [1 \; x_i \; x_i^2 \ldots]$$

- interactions between feature dimensions

$$\phi(x_i) = [1 \quad x_{i1}x_{i2} \quad x_{i3}x_{i4} \ldots]$$

- Many other choices possible

Linear Function

Parameters:

weight vector $w = [w_1, w_2, \dots w_F]$

bias scalar b Or w_0

Prediction:

$$\hat{y}(x_i) \triangleq \sum_{f=1}^{F} w_f x_{if} + b$$

$$Y = w_0 + w_1X_1 + w_2X_2 + \dots w_fx_f$$

Non-linear Function

A nonlinear function of x:

$$\hat{y}(x_i) = \theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \theta_3 x_i^3$$

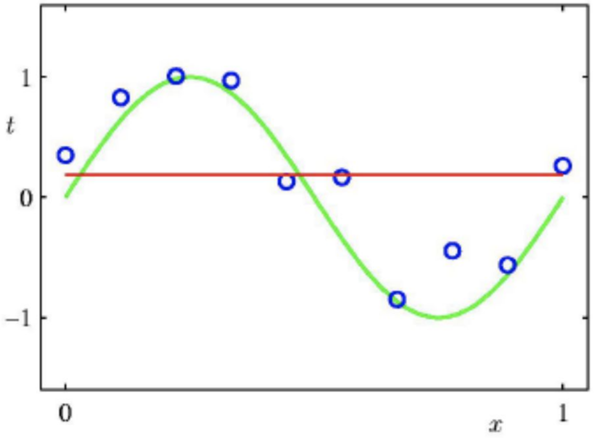
Can be written as a linear function of $\phi(x_i) = [1 \; x_i \; x_i^2 \; x_i^3]$

$$\hat{y}(x_i) = \sum_{g=1}^{4} \theta_g \phi_g(x_i) = \theta^T \phi(x_i)$$

“Linear regression” means linear in the parameters (weights, biases)

Features can be arbitrary transforms of raw data

Suppose we know the function



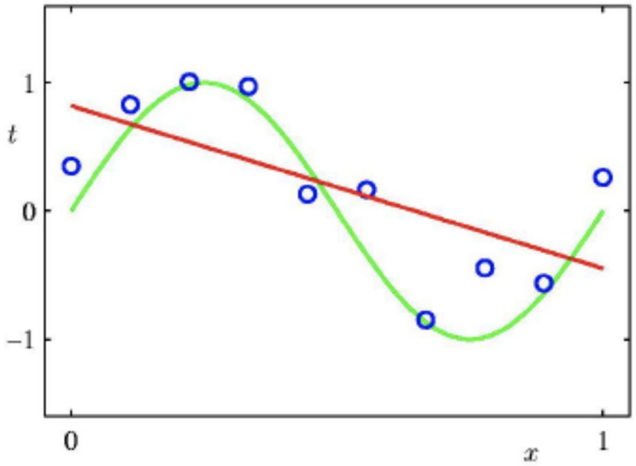
--- true function

o training data

-- predictions from LR using polynomial features

0-degree

Suppose we know the function



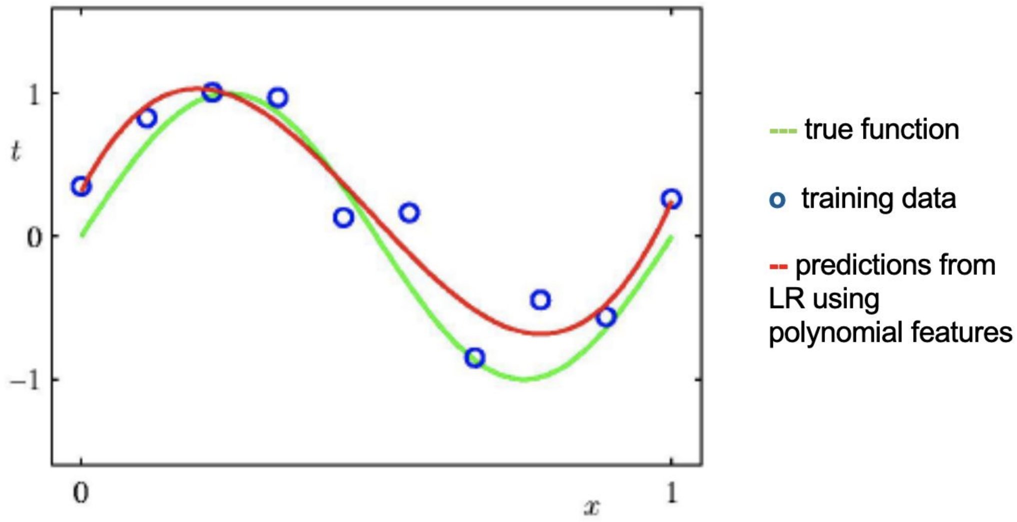
--- true function

o training data

-- predictions from LR using polynomial features

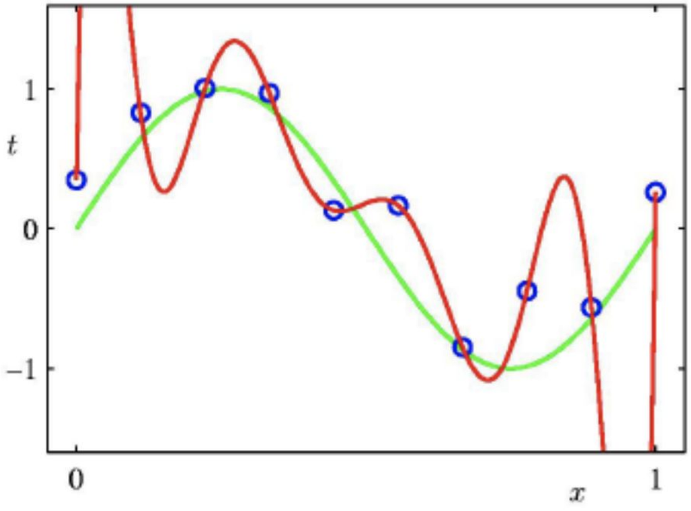
1st-degree

Suppose we know the function



3rd-degree

Suppose we know the function



--- true function

o training data

-- predictions from LR using polynomial features

9th-degree

Choosing the complexity

- At that moment we need to make one of two choices:
    - Which model to use (Linear regression, regularized regression, etc) → Avoid Bias (wrong assumption about the model)
    - Hyperparameters, parameter tuning → Avoid Variance

Overfitting/Underfitting

- Overfitting Problem
  - Trains well, but fails to learn later
- Underfitting Solution
  - Fails to find the right model

Overfitting/Underfitting

- Overfitting Solution
  - Choose best generalizing complexity
  - Add penalty in training objective
- Underfitting Solution
  - Increase model complexity (add more features!)

Overfitting/Underfitting

- In reality, we don't know the function
- Two criteria

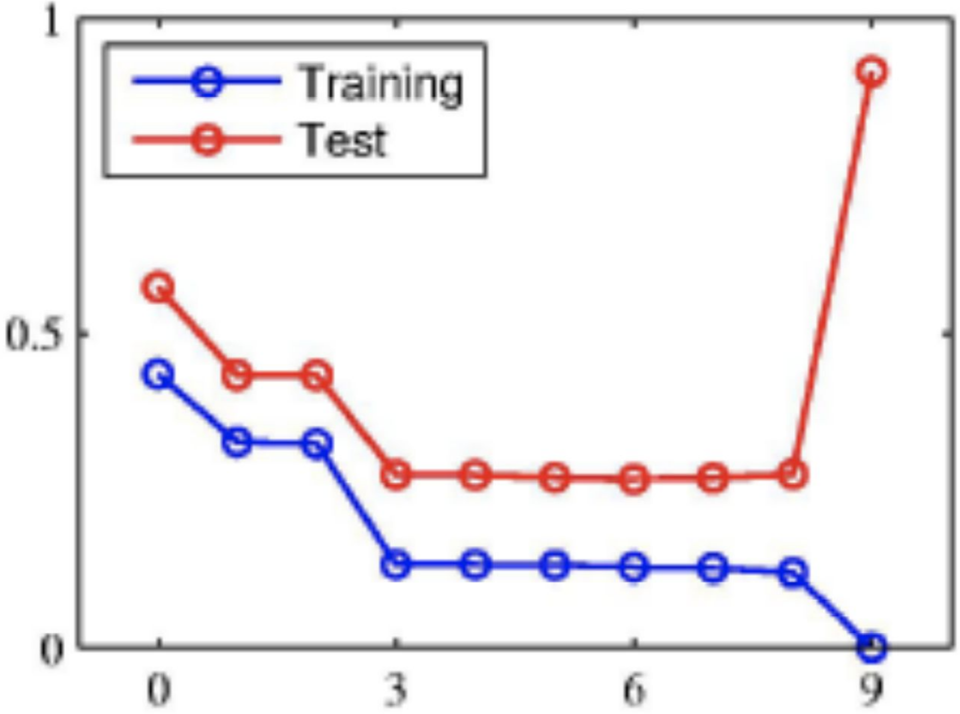
1- Determine whether the system is overfitting or underfitting

- Error curves
- Fixed validation set
- Cross validation

2- Choose the complexity accordingly

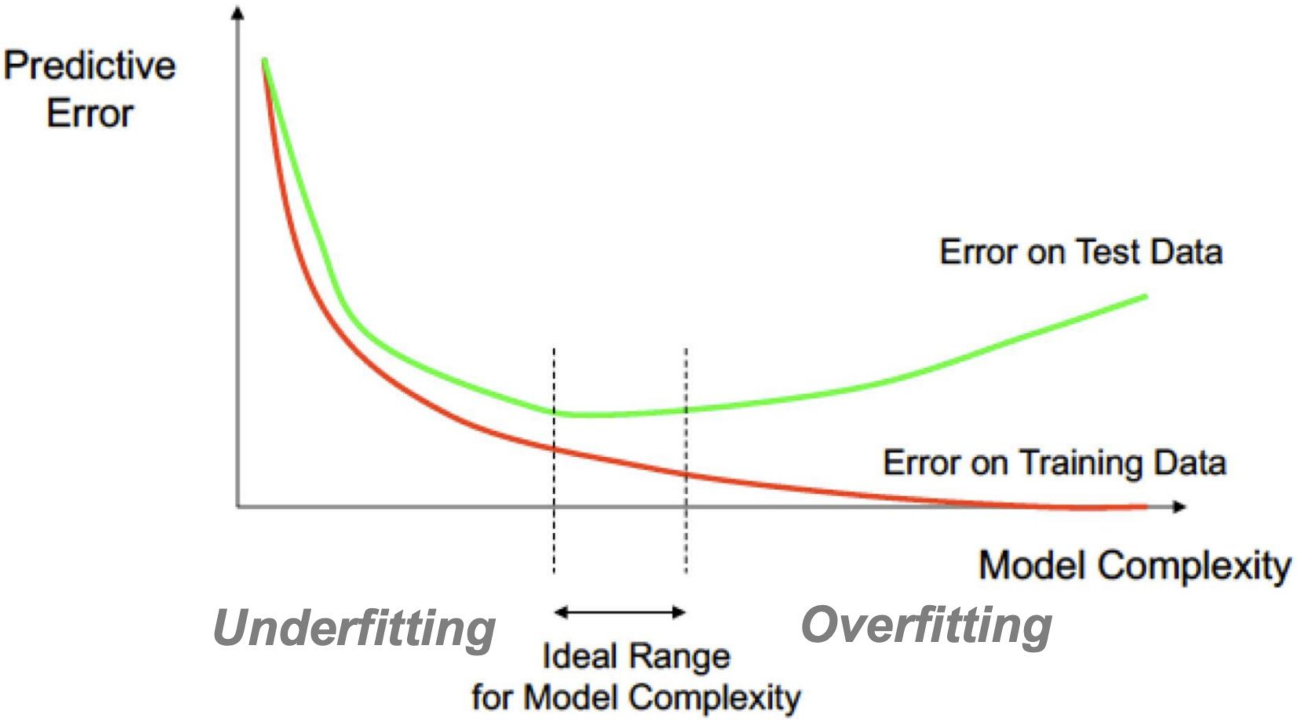
1- Error Curves

mean
squared
error



polynomial degree

1- Error Curves



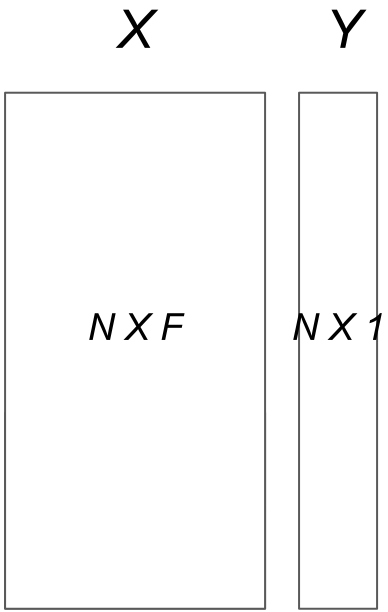
So far we learned three types of plots

- Data Plot
- Cost function Plot
- Error Plot

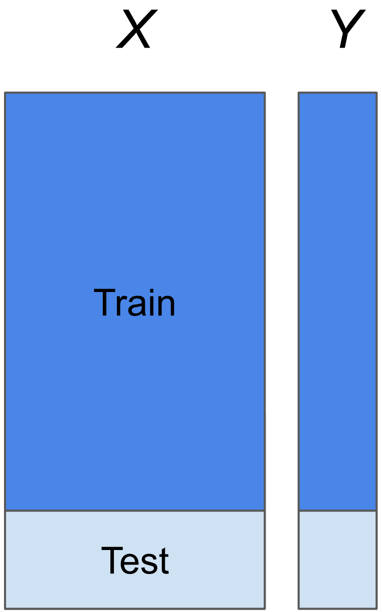
Overfitting/Underfitting

- Error curves are not always easy to plot

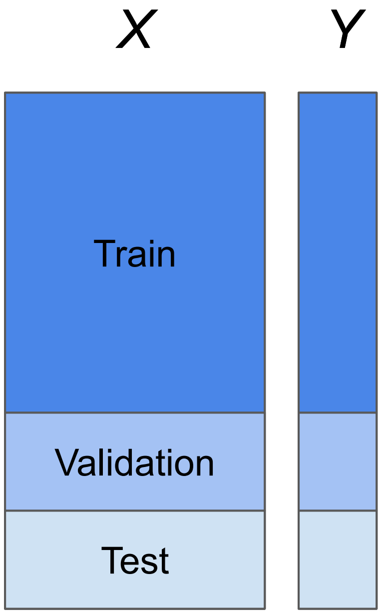
Data Splitting



Data Splitting

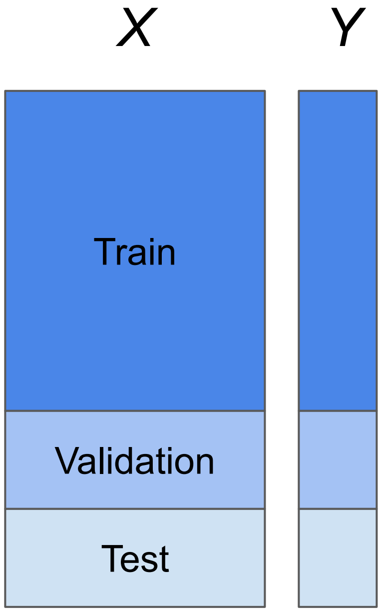


Data Splitting



2- Fixed Validation Set

- The “validation dataset” is usually used to describe the evaluation of models when tuning hyperparameters and data preparation
- The “test dataset” is usually used to describe the evaluation of a final tuned model when comparing it to other final models.



2- Fixed Validation Set

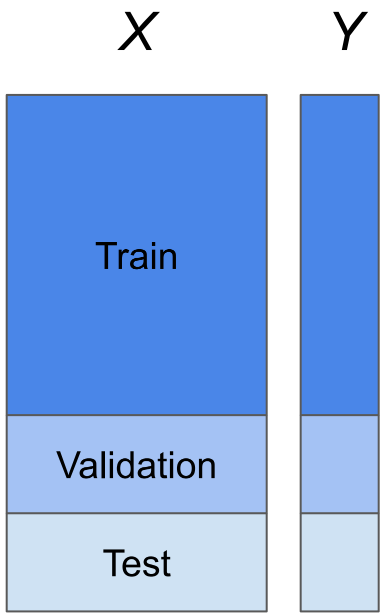
Option: Fit on train, select on validation

1) Fit each model to training data

2) Evaluate each model on validation data

3) Select model with lowest validation error

4) Report error on test set

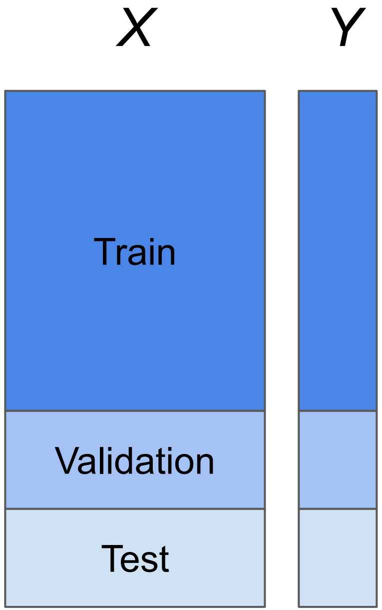


2- Fixed Validation Set

What sizes to pick?

- Will train be too small?
- Is validation set used

effectively? (only to evaluate predictions?)



2- Fixed Validation Set

For small datasets, randomness in validation split **will impact selection**

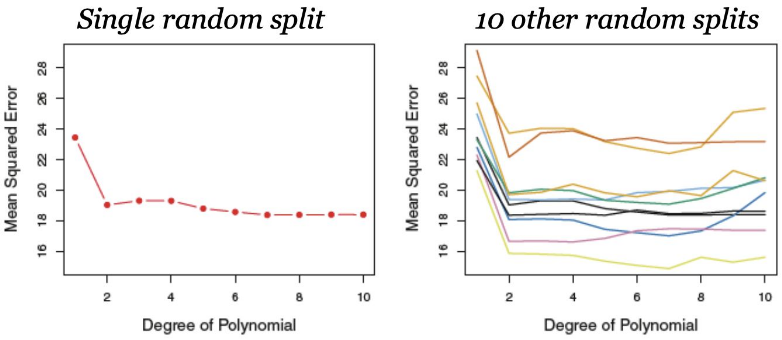
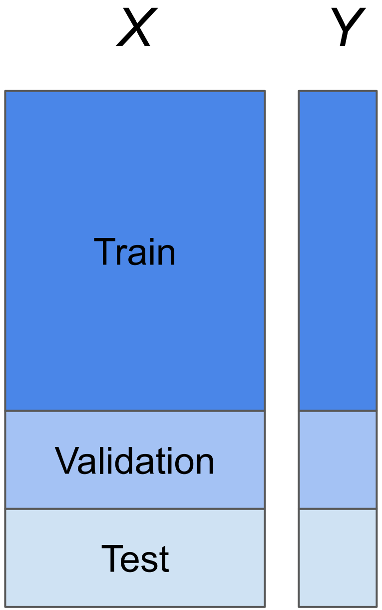


FIGURE 5.2. *The validation set approach was used on the* Auto *data set in order to estimate the test error that results from predicting* mpg *using polynomial functions of* horsepower. Left: *Validation error estimates for a single split into training and validation data sets.* Right: *The validation method was repeated ten times, each time using a different random split of the observations into a training set and a validation set. This illustrates the variability in the estimated test MSE that results from this approach.*

Credit: ISL Textbook, Chapter 5



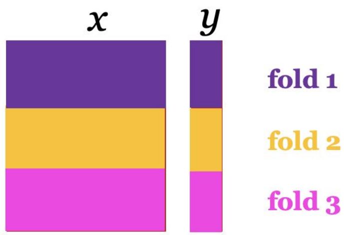
3- Cross validation

- Model performs well in Training but poorly in validation - > OverFitting
- Model performs poorly on both → UnderFitting
- What does poor performance mean?
  - Big error value

3- Cross Validation

3-fold Cross Validation

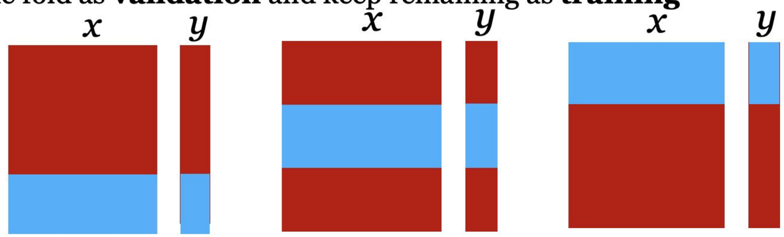
Divide labeled dataset into 3 even-sized parts



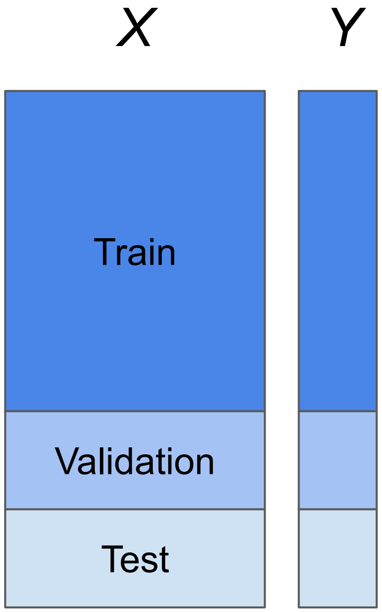
Fit model 3 independent times.
Each time leave one fold as **validation** and keep remaining as **training**

train

validation

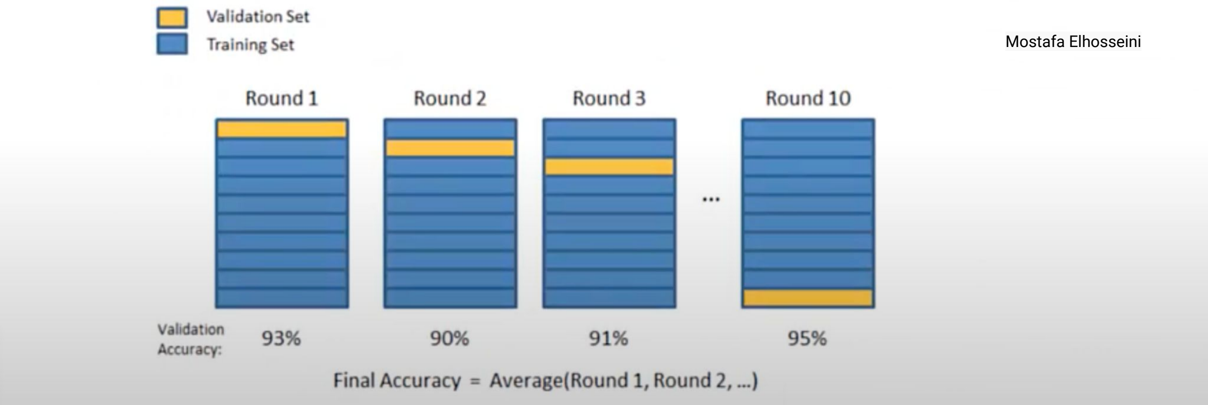


Heldout error estimate: average of the validation error across all 3 fits

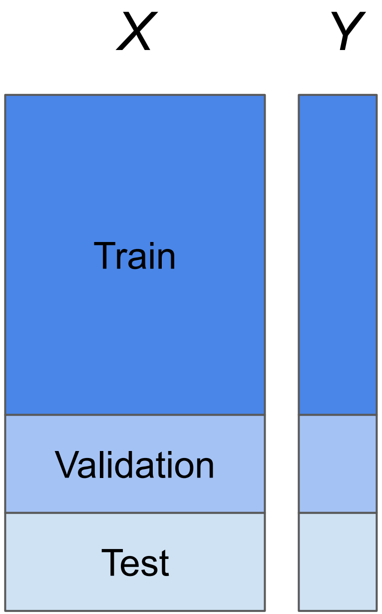


3- Cross validation

- K-fold Cross Validation
  - Split data into k different subsets
  - Use k-1 subsets in training
  - Leave the last fold for validation
  - Take the average against each fold
  - Finalize the model



Mostafa Elhosseini



Cross Validation

K-fold CV: How many folds K?

Can do as low as 2 fold

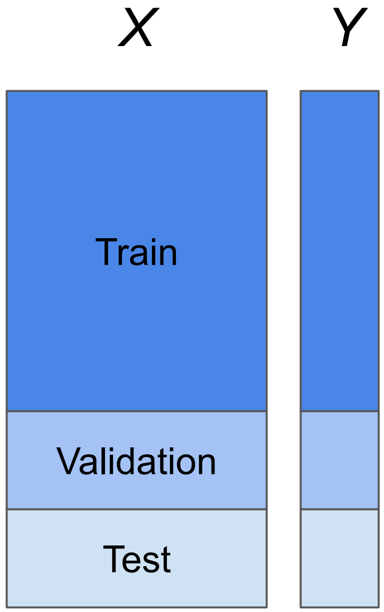
Can do as high as N-1 folds ("Leave one out")

Usual rule of thumb: 5-fold or 10-fold CV

Computation runtime scales linearly with K

Larger K also means each fit uses more train data, so each fit might take longer too

Each fit is independent and parallelizable



Cross Validation

Leave-one-out (N-1 folds)

9 separate splits
Each one with 10 folds

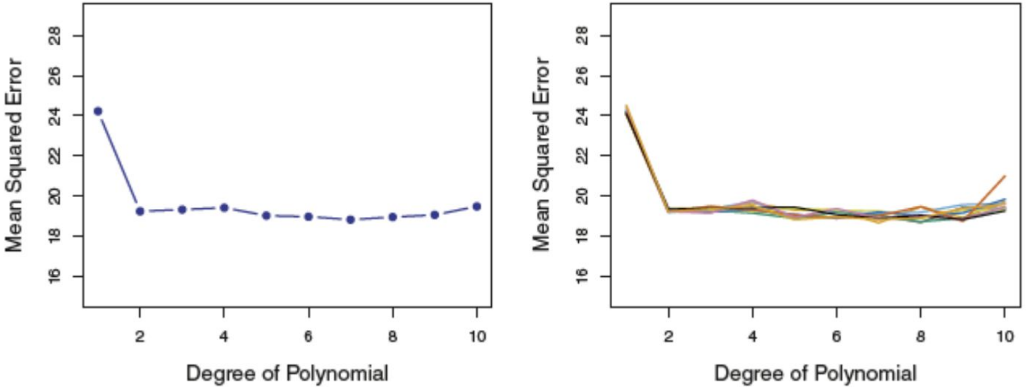


FIGURE 5.4. *Cross-validation was used on the* `Auto` *data set in order to estimate the test error that results from predicting* `mpg` *using polynomial functions of* `horsepower`. Left: *The LOOCV error curve.* Right: 10*-fold CV was run nine separate times, each with a different random split of the data into ten parts. The figure shows the nine slightly different CV error curves.*

Credit: ISL Textbook, Chapter 5

Regularization

- By now, we learned methods to detect overfitting and underfitting
- One way of solving underfitting is to increase the model complexity
- How to solve overfitting?
  - Reduce number of features/Polynomial degree - manually or automatically (will discuss in next lectures)
  - Regularization (reduce theta)

Regularization

- Ridge (L2 Penalty) → sum of squares of the weights
- Lasso (L1 Penalty) → sum of absolute values of the weights (Manhattan distance)

Regularization: Ridge

Idea: Add Penalty Term to Loss

Goal: Avoid finding weights with large magnitude
Result: **Ridge regression**, a method with objective:

$$J(\theta) = \sum_{n=1}^{N} (y_n - \theta^T \phi(x_n))^2 + \alpha \sum_{g=1}^{G} \theta_g^2$$

Penalty term: Sum of squares of entries of theta
= Square of the "L2 norm" of theta vector
Thus, also called "L2-penalized" linear regression

Hyperparameter: Penalty strength "alpha" $\alpha \geq 0$

Alpha = 0 recovers original unpenalized Linear Regression
Larger alpha means we prefer smaller magnitude weights

Regularization: Ridge

Rewrite in matrix notation?

N : num. examples
G : num transformed features

$$J(\theta) = \sum_{n=1}^{N} (y_n - \theta^T \phi(x_n))^2 + \alpha \sum_{g=1}^{G} \theta_g^2$$

Rewriting, this is equivalent to

Can rewrite sum of squares as an inner product of theta vector with itself

$$J(\theta) = (y - \Phi\theta)^T (y - \Phi\theta) + \alpha\theta^T\theta$$

$$\underset{N\,x\,G}{\Phi =} \begin{bmatrix} 1 & \phi_1(x_1) & \dots & \phi_{G-1}(x_1) \\ 1 & \phi_1(x_2) & \dots & \phi_{G-1}(x_2) \\ \vdots & & \ddots & \\ 1 & \phi_1(x_N) & \dots & \phi_{G-1}(x_N) \end{bmatrix} \quad \underset{N\,x\,1}{y =} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad \underset{G\,x\,1}{\theta =} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_G \end{bmatrix}$$

Regularization: Ridge

Estimating weights for L2 penalized linear regression

Optimization problem: "Penalized Least Squares"

$$\min_{\theta} \; (y - \Phi\theta)^T (y - \Phi\theta) + \alpha\theta^T\theta$$

Solution:

$$\theta^* = (\Phi^T\Phi + \alpha \underbrace{I_G}_{\substack{G \, x \, G \\ \text{identity matrix}}})^{-1}\Phi^T y$$

If alpha > 0 , the matrix is **always** invertible!
Always one unique optimal theta vector, provided by this formula

Regularization: Ridge

- Make sure to choose Lambda carefully not to over minimize thetas

Regularization: Ridge Limitations

- Ridge Regression is more sensitive to the scale of your features.
- Before feeding data into a Ridge regression model, should standardize the scale of all features, so the penalty acts on each feature in more uniform way.
  - Rescale each column between 0 and 1: sklearn's MinMaxScaler https://scikit-learn.org/stable/modules/preprocessing.html#scalingfeatures-to-a-range
  - Transform each column to have mean 0 and variance 1: sklearn's StandardScaler https://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler
  -
- OR, you can impose your own feature-specific penalties if you want.

Regularization: Lasso

N : num. examples
G : num transformed features

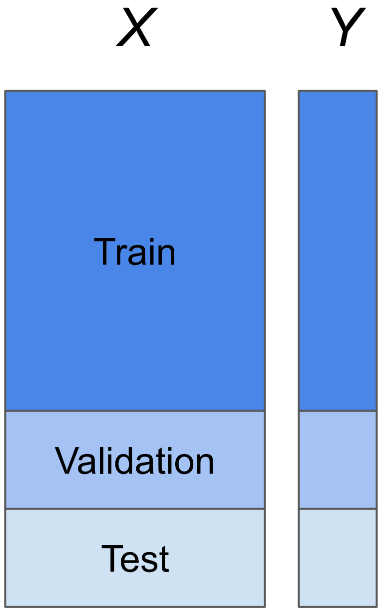
$$\min_{\theta} \quad (y - \Phi\theta)^T (y - \Phi\theta) + \alpha \sum_{g=1}^{G} |\theta_g|$$

Sum of absolute values of entries (aka the L1 norm of the vector theta)

Like L2 penalty (Ridge), the Lasso objective above encourages small magnitude weights.

Regularization

- Regularization is added in the training phase
- During the validation, the cost function is used without regulization



Recap

- New model: Nonlinear, by increasing hyperparameters
- Methods to detect overfitting and underfitting
- We can solve underfitting by increasing the complexity
- We can solve the over fitting by regularization

Questions!