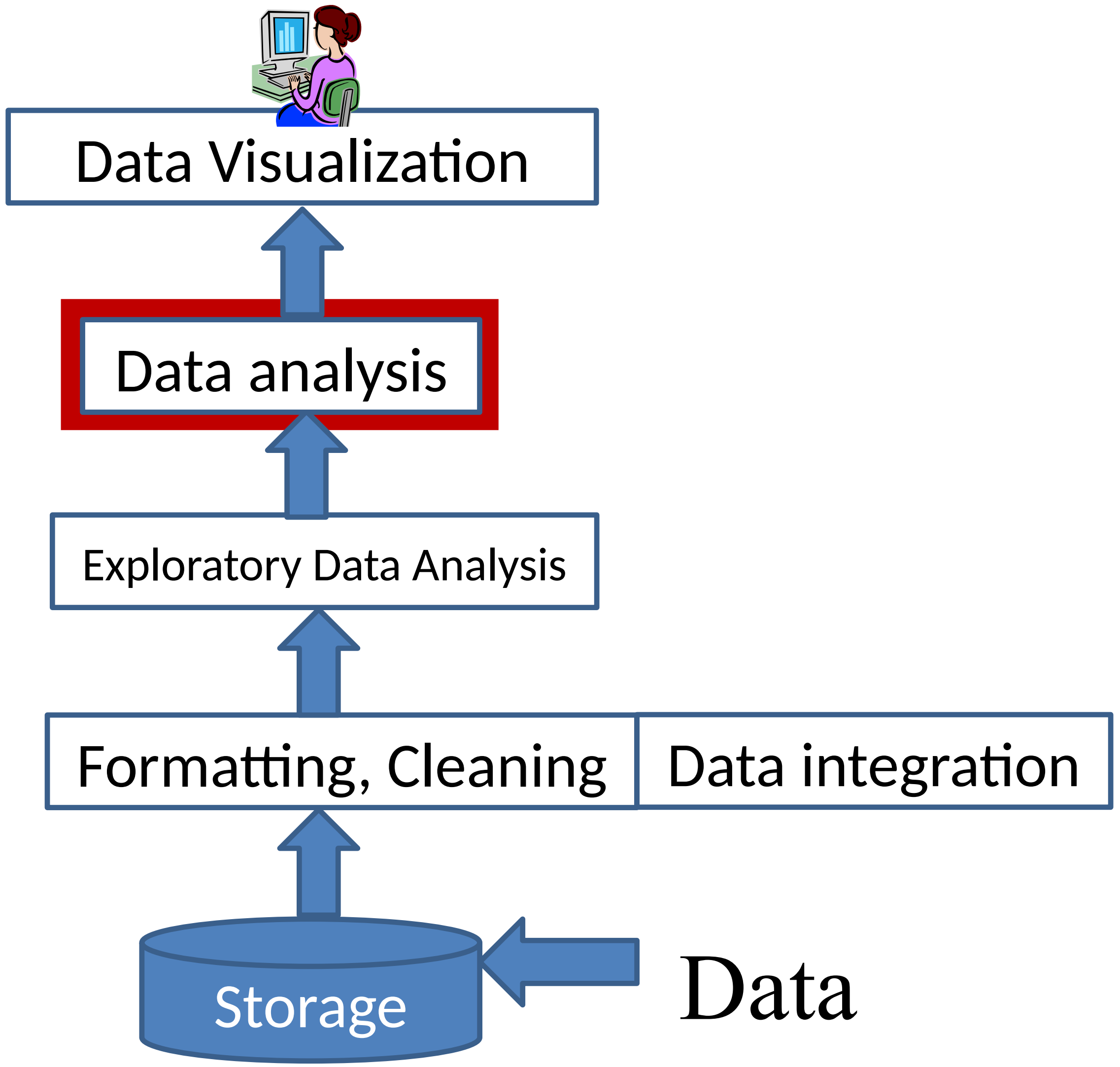
CPSC 483
Evaluation of ML models

Anand Panangadan
apanangadan@fullerton.edu

What we will cover this week

- Linear regression
- Evaluating a regression model
- Analytical and iterative approaches
- Regularization

The Data Science Process



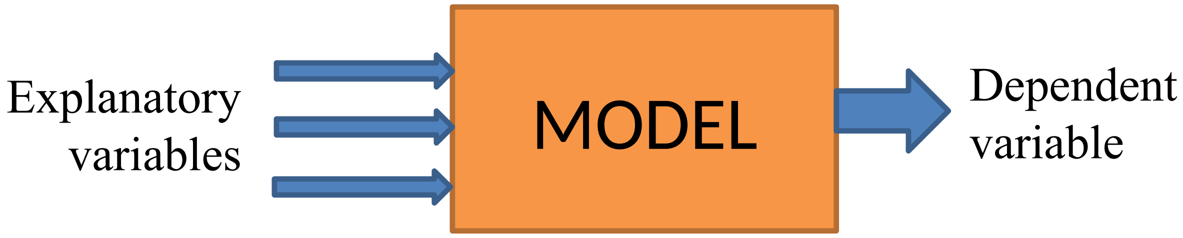
What is a Math Model?

- Model
  - Representation of a phenomenon
  - Describes the relationship between variables
- Mathematical model
  - Numerically describe relationship between variables

"All models are wrong, but some are useful"

Regression Model

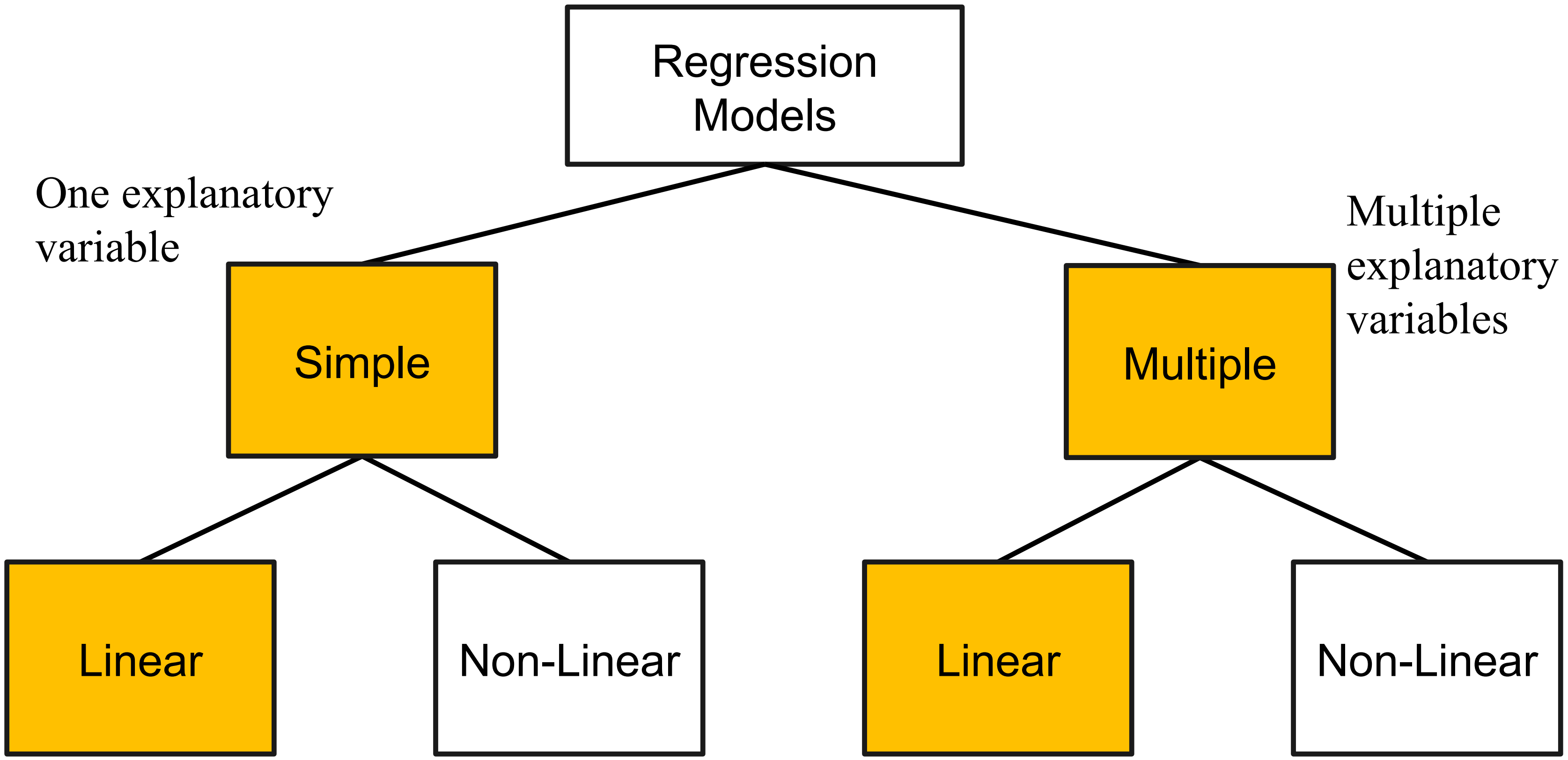
- Relationship between one dependent variable and explanatory variable(s)
- Dependent variable is continuous
- One equation describes the relationship



Steps to modeling

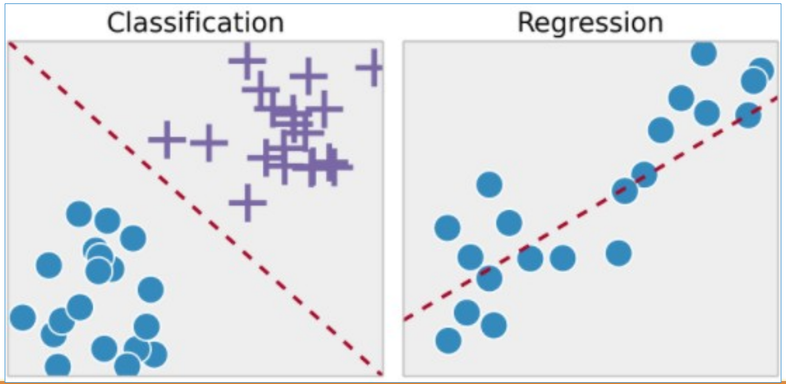
1. Define a family of models
   - Specify the generic pattern of relationship between variables
     - Linear, Quadratic, …
2. Fit the model to the data
   - Identify the “best” model from the family of models
   - Give values to the model parameters

Types of Regression Models



Supervised Learning

- Given a training data set
  - Learn a function
    - The process of learning is called training.
  - Predict outcomes for a given (**new data**):
    - If the learned function f is used to predict a continuous value, , it's called regression.
    - If the learned classifier f is used to determine a discrete value, , it's called classification.
- **Classification vs. regression**
  - Classification is to find the *decision boundary* that **separates** the different groups of data as clearly as possible.
  - Regression is to find the *function* that **fits** the data the best.



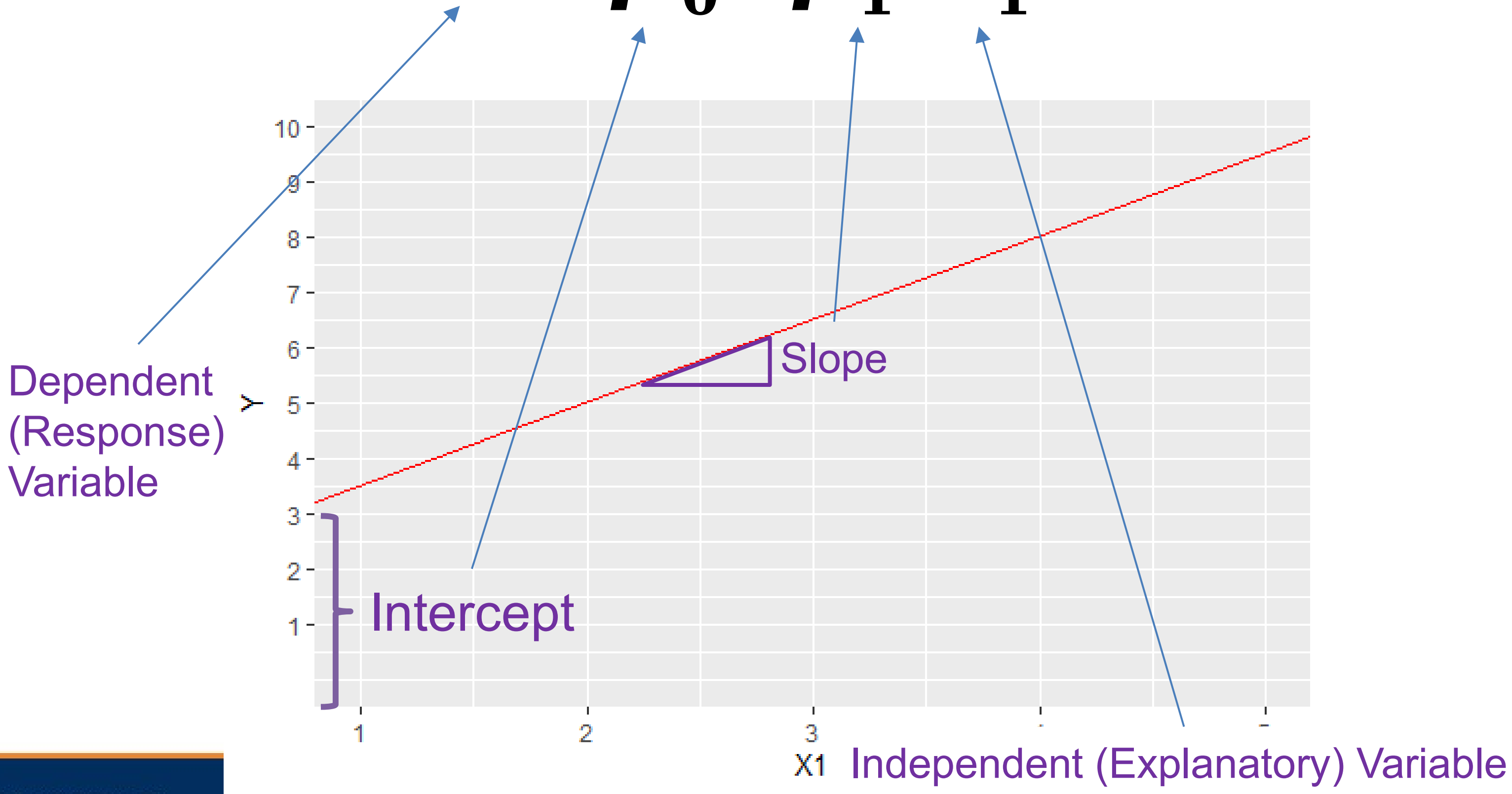
Class work

- Consider the problem of estimating Height given Weight and Age
  - What are some linear models for this?
  - What are some nonlinear models for this?

Linear Regression Model

- Relationship between variables is a linear function

$$\boldsymbol{Y} = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 \boldsymbol{X}_1$$



Interpretation of Coefficients

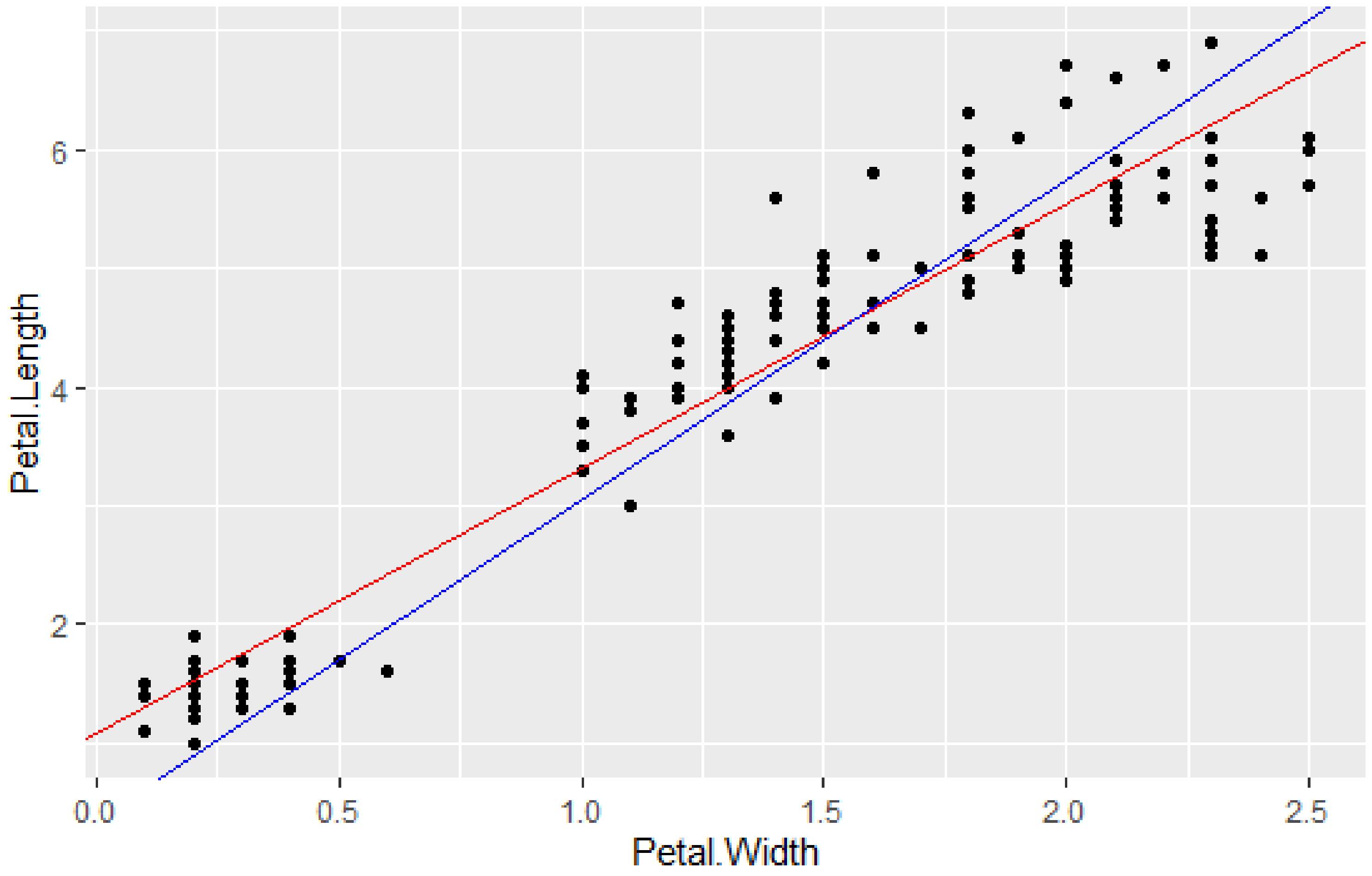
1. Slope
   - per unit change in the dependent variable for each unit change in the independent variable.
2. Y-Intercept
   - Average Value of *Y* When *X* = 0

Class work

- How does Petal.Length change with Petal.Width in the iris dataset?
  - Which is the dependent variable? Which is the explanatory/independent variable?
  - Draw a scatterplot
  - Draw *any* straight line that fits the data well

```
from sklearn.datasets import load_iris
iris = load_iris(as_frame=True)
fig, ax = plt.subplots()
…
plt.plot(x, y, "b.")
plt.grid()
ax.axline(xy1=(0, intercept), slope=slope, color='red')
```

Which model/line is better?



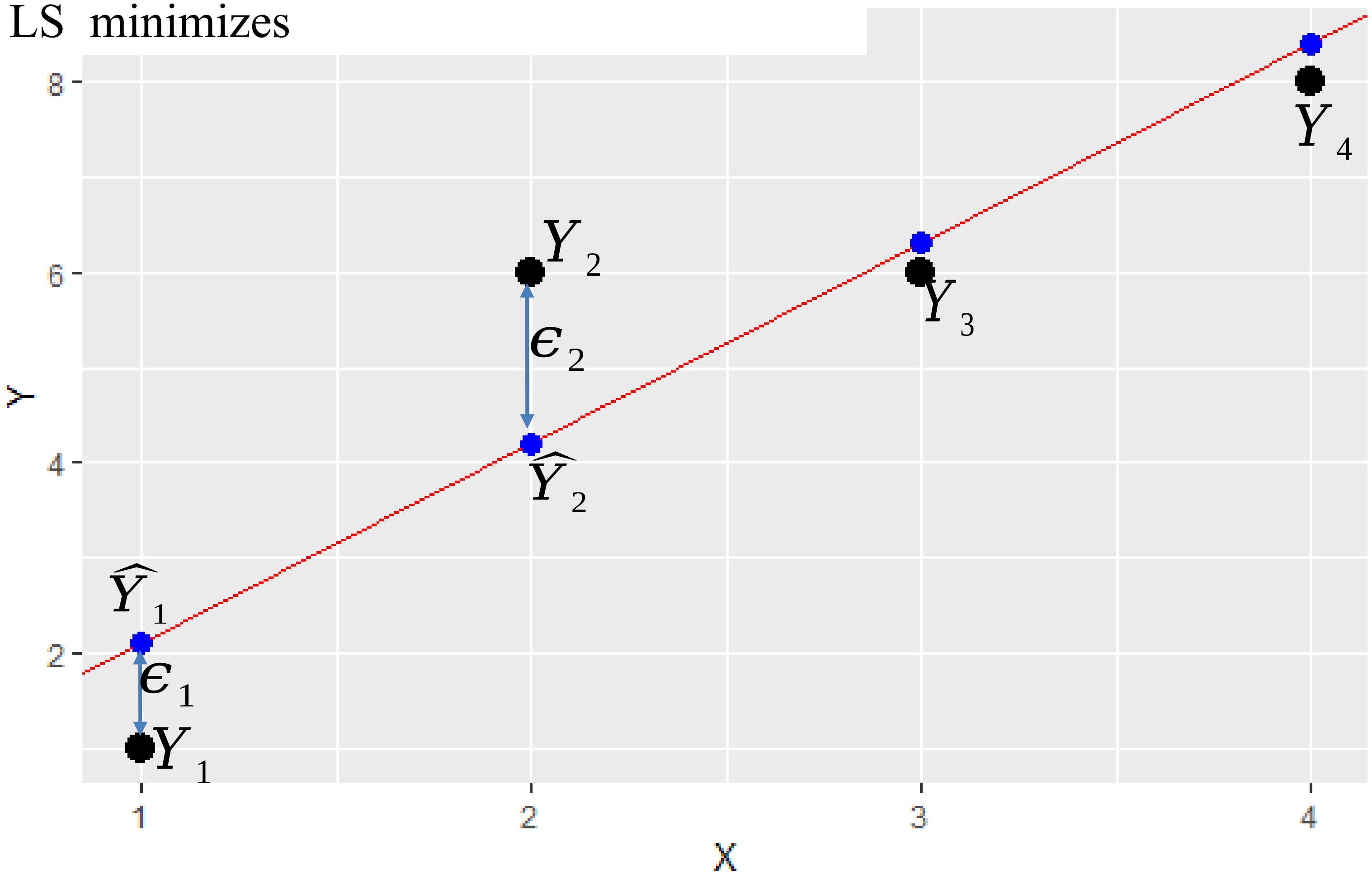
Least Squares (LS)

- Best Fit
  - Select the slope & intercept that gives the smallest difference between Actual Y values () & Predicted Y values ().
  - Square the differences
  - Differences are called *errors* or *residuals*

$$\sum_{i=1}^{n}\left(Y_i-\widehat{Y}_i\right)^2=\sum_{i=1}^{n}\epsilon_i^2$$

Least Squares Graphically

LS minimizes



LS minimization is efficient!

- Sample slope

- Sample Y – intercept

- Prediction equation

When can we use LM?

- One dependent variable
- Dependent variable is continuous

Linear Regression in scikit-learn

- from sklearn.linear_model import LinearRegression

- lin_reg = LinearRegression()
- lin_reg.fit(X, y)
- lin_reg.intercept_, lin_reg.coef_

Plot the model

```
y_predict = lin_reg.predict(X)
plt.plot(X, y_predict, "r-")

OR for simple linear regression:

ax.axline(xy1=(0, lin_reg.intercept_),
slope=lin_reg.coef_[0], color='red')
```

Class work

- Consider the “Auto MPG” dataset which “concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 multivalued discrete and 5 continuous attributes.” The goal is to model mpg given engine displacement.
  - Load the autompg.csv file on Canvas
  - Which is the dependent variable? Which is the independent variable?
  - Plot mpg vs. displacement (code, plot)
  - Overlay best fit line over the dataset (code, plot)

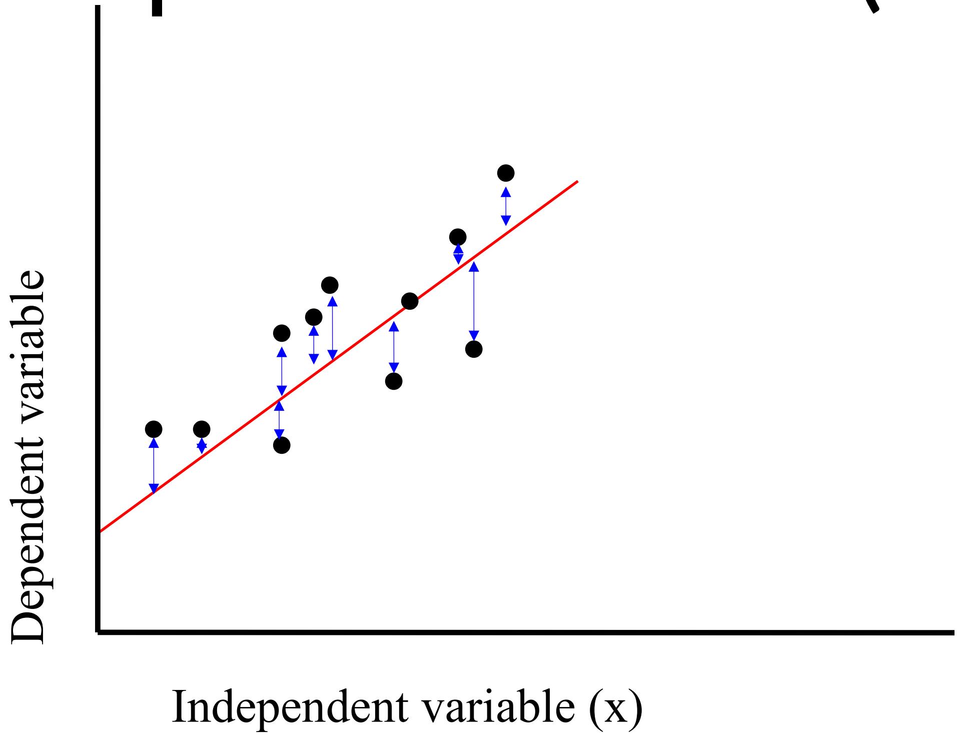
When should we use LM?

- Is the “best” fit model a good model?!
- Evaluating the model
- Two conflicting objectives
  - Goodness-of-Fit
    - We want model to match the data
  - Complexity
    - We want model to be “simple”
- “Principle of parsimony”
  - Find a model that is as simple as possible without sacrificing too much goodness-of-fit

Coefficient of Determination

- The proportion of total variation (SST) that is explained by the regression (SSR) is the Coefficient of Determination (
- ranges between 0 and 1
  - higher its value, the more accurate is the regression model

Sum of Squares of Error (SSE)

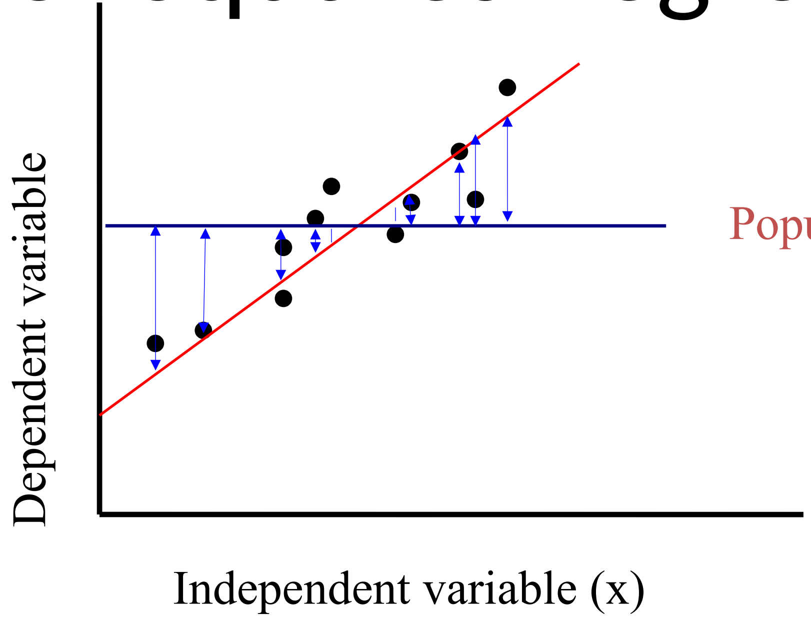


A least squares regression selects the line with the lowest total sum of squared prediction errors.

This value is called the Sum of Squares of Error (SSE)

This is the "unexplained" variation

Sum of Squares Regression (SSR)



Population mean:

The Sum of Squares Regression (SSR) is the sum of the squared differences between the prediction for each observation and the population mean.

This is the "explained" variation

Total Sum of Squares (SST)

SSR = (measure of explained variation)

SSE = (measure of unexplained variation)

SST = SSR + SSE = (measure of total variation in y)

Coefficient of Determination

```
lin_reg.score(X, y)
```

OR

```
from sklearn.metrics import r2_score
r2_score(y, y_predict)
```

Regression Diagnostics

- The three conditions required for the validity of the regression analysis are:
  1. the error variable is normally distributed
  2. the error variance is independent of x
  3. The errors are independent of each other
- How can we diagnose violations of these conditions?

Residuals

- Also called “errors”
- residuals = y – y_predict

Residual Analysis

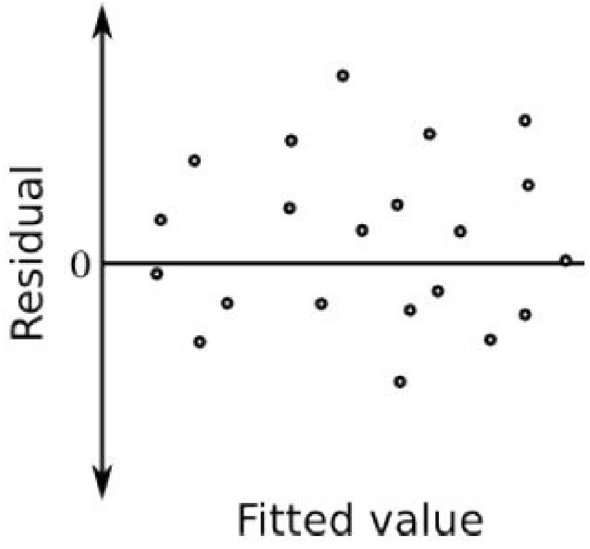
- Examine the residuals
- Non-normality.
  - Examine the residual histogram and look for a bell shaped curve with a mean close to zero

```
residuals = y - y_predict
plt.hist(residuals) # optional bins= parameter
```

Residual Analysis

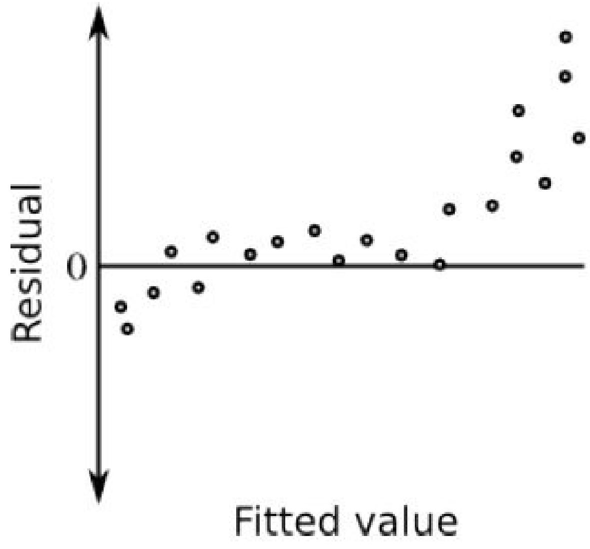
- Plot a scatterplot
  - Residuals vs *x*
  - Points should appear to be randomly scattered around zero

```
plt.plot(X, residuals, 'b.')
```

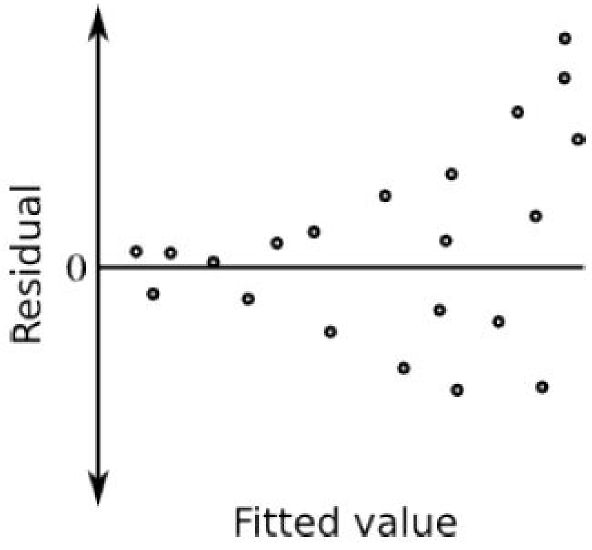


Residuals appear randomly scattered around zero, and their spread appears constant

homoscedasticity



systematic behavior in the residuals



Residuals scattered randomly about zero but variability isn't constant

heteroscedasticity

Outliers

- Outlier: an observation that is unusually small or large
- Several possibilities need to be investigated when an outlier is observed:
  - There was an error in recording the value.
  - The point does not belong in the sample.
  - The observation is valid.
- *Suspect* an observation is an outlier if its

$$|\text{standardized residual}| > 2$$

Making predictions

- The fitted model can be used to make predictions for new *x* values
  - For accuracy, *x* should be within the range seen in the data used for modeling (interpolation)

Procedure for Regression Diagnostics

- Gather data for the variables in the model
- Draw a scatterplot to determine whether a linear model appears to be appropriate
- Determine the regression equation
- Check the required conditions for the errors (residual analysis)
- Check the existence of outliers
- Assess the model fit (R^2)
- If the model fits the data well, use the model to predict new values

Class work

- Load the autompg.csv file on Canvas
- The goal is to model mpg given engine displacement
- Predict the mpg of a car with engine displacement=250
- Predict the mpg of a car with engine displacement=600

Why is LR important?

- Simple
- Efficient
- Assumptions are reasonable
- Model is surprisingly powerful
  - Multiple predictor variables
  - Can use transformed predictors

Multiple Linear Regression

- More than one independent variable can be used to explain variance in the dependent variable
- Multiple regression takes the form
- k+1: number of model parameters to estimate

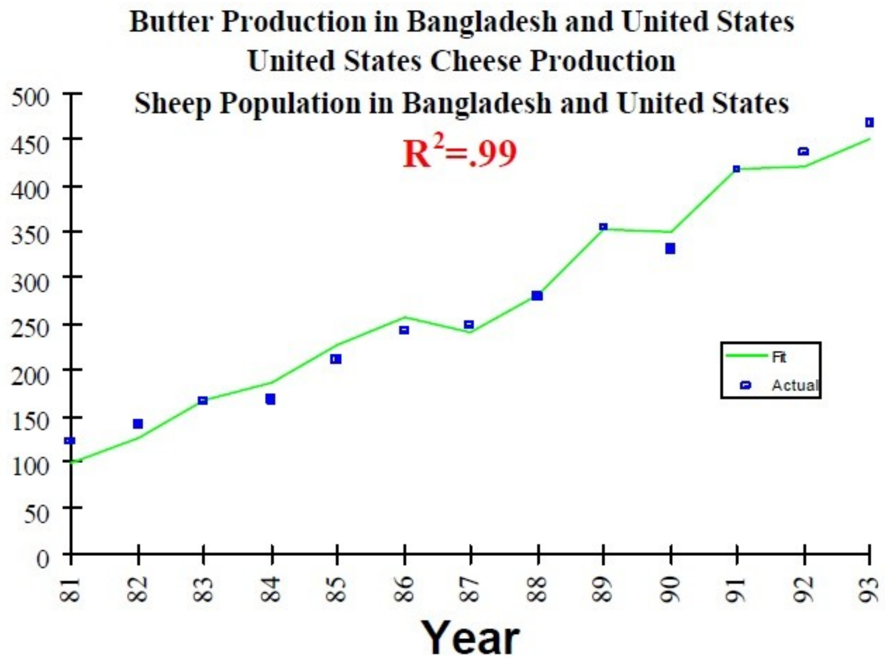
Classwork

- Write the best fit linear model equation relating Petal.Width to Petal.Length and Sepal.Length in the iris dataset
  - Write code to get coefficients

Adjusted R2

- Will adding more independent variables *always* increase R2?
- What if the independent variables are completely random?
- Surprisingly, yes!

- Must focus on number of variables
  - More variables ☾ less trustworthy model
- Adjusted R2
  - Adjusts for the number of variables by reducing R2 for more complex models
- Overfitting: A particular problem in Big Data
- How to get a high R2 model for the S&P 500?!



Adjusted R2

$$\bar{R}^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

- p = total number of explanatory variables (excluding the intercept)
- n = sample size

The Linear Regression Model

- **The general linear model** can be written as an equation:

  - are independent (exploratory) variables/features, representing the data.
  - is a dependent (response) variable.
  - are coefficients/weights; k is number of features.
  - ε is a random variable (noise by measurements or others), assumed to have a $N(0, \sigma^2)$ distribution; is a random variable with mean, μ and variance σ^2.

- Example: Housing price data with four features

Price (K\$) (Y)	*Size (X_1)*	*Bedroom (X_2)*	*Bathroom (X_3)*	*Built year (X_4)*
375 ()	1024 (x_{11})	3 (x_{12})	2 (x_{13})	1978 (x_{14})
425 ()	1329 (x_{21})	3 (x_{22})	5 (x_{23})	1992 (x_{24})
…	…	…	…	…
465 ()	1893 (x_{N1})	4 (x_{N2})	4 (x_{N3})	1980 (x_{N4})

Uppercase is a column data.

Lowercase is a row data.

is a value.

ng dataset.

Regression Model in Vector/Matrix Form

- **Stacking all the equations**
  - k number of variables (features)
  - N number of equations (rows)

- **The equations in vector form**
  - or

is a feature (column) vector.
is a row vector (ith training example).

- **The equations in matrix form**
  - => or

$$\underbrace{\qquad}_{Y}\quad\underbrace{\qquad\qquad\qquad\qquad}_{X}\quad\underbrace{\quad}_{w}$$

The Goal of Regression Modeling

- **Housing price data set**

Price (K$) (Y)	*Size (X_1)*	*Bedroom (X_2)*	*Bathroom (X_3)*	*Built year (X_4)*
375 (y_1)	1024 (x_{11})	3 (x_{12})	2 (x_{13})	1978 (x_{14})
425 (y_2)	1329 (x_{21})	3 (x_{22})	5 (x_{23})	1992 (x_{24})
…	…	…	…	…
465 (y_N)	1893 (x_{N1})	4 (x_{N2})	4 (x_{N3})	1980 (x_{N4})

- **Housing price data set in matrix form**
  - , X and Y are from training data (known); w are unknown.

- **The goal is to estimate** of the function

FINDING REGRESSION MODELS ANALYTICALLY

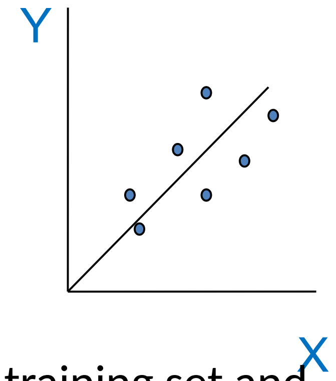
The Least Squares Approach

- The **basic idea** of the (ordinary) least squares approach
  - **Find w that minimizes the squared error**:

    is the actual value.

    is the estimated parameter.

    is the predicted value of y, for a given data x_i.



  - Finding w with minimum squared distances between data in a training set and predicted line.
    - **Note:** The regression problem is **restated** as **an optimization problem** of an **error function.**

- **Why least square?**
  - This problem can be solved **<u>analytically</u>** (convincing probabilistic interpretation). How?

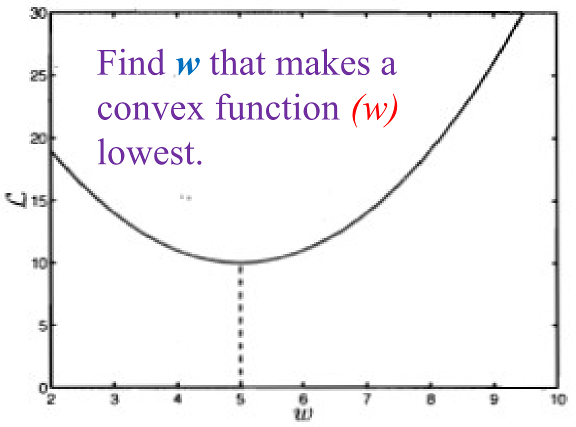
Solving Linear Regression by the <u>Least Square</u> Approach

- **Find parameters** W that **minimizes** the **loss function** for <x, y> value pairs of all the training examples (Gauss and Legendre, 1809).

$$\textbf{Minimize}\; l(w) = \frac{1}{N}\sum_i \left(y_i - f_w(x_i)\right)^2$$

Squaring residuals is important!

N is the number of samples in a training data set.

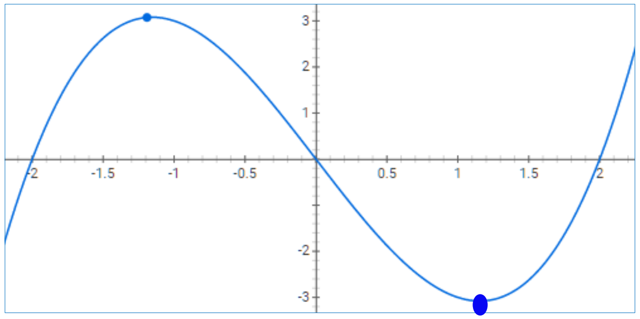


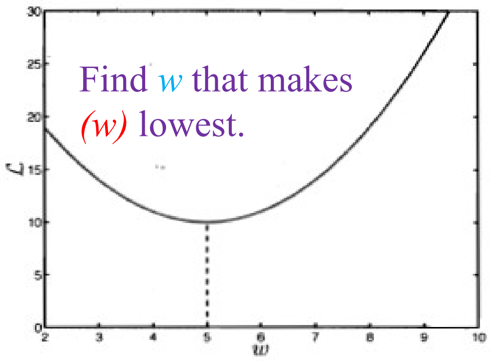
Find *w* that makes a convex function *(w)* lowest.

- Residual Sum of Squares (**RSS**) =
- Loss or Mean Squared Error (**MSE**) =

$$Argmin_w l(w) = \frac{1}{N}\sum_i \left(y_i - w^T x_i\right)^2$$

Derivatives to Test Turning Points





- The **first derivative** test:
  - A local minimum is where $f'(x)$ changes from **–** to **+**.
  - A local maximum is where $f'(x)$ changes from **+** to **–**.
  - **Solving $f'(x) = 0$** returns the x-coordinates of all turning points (solutions).

- The **second derivative** test:
  - **If** $f''(x) > 0$, the turning point at x is concave up (convex); f(x) has a local minimum at x.
  - **Else if** $f''(x) < 0$, the turning point at x is concave down (concave); f(x) has a local maximum at x.
  - **Else**: Saddle point (neither local minima nor maxima)

Deriving a Multiple Regression Model

$$\boxed{Argmin_w l(w) = \frac{1}{N}\sum_i \left(y_i - w^T x_i\right)^2}$$ from

- **To find w that minimizes**
  - Re-arranging =>

    =>
  - Taking a partial derivative of w.r.t w and solving it (= 0)

    (Refer to Rogers at el. book for the detailed derivation)
  - To get the estimated w, multiply both sides:
  - Finally, we derived a linear model:
    - We analytically found a solution that satisfies the least square error.

Predicting a New Value

- Given a **new example** , the **predicted value**, from the **model** obtained by the least square is computed by:

  where

  - A value of is computed by **dot product** of and , that is, $=$

  $= + x_{i1} + x_{i2} + \ldots + x_{ik}$

FINDING REGRESSION MODELS ITERATIVELY

Optimizing Loss Function Iteratively

(to deal with a huge matrix X from a large data set and computing inverse matrix)

- For **simple regression**, **minimize** w.r.t

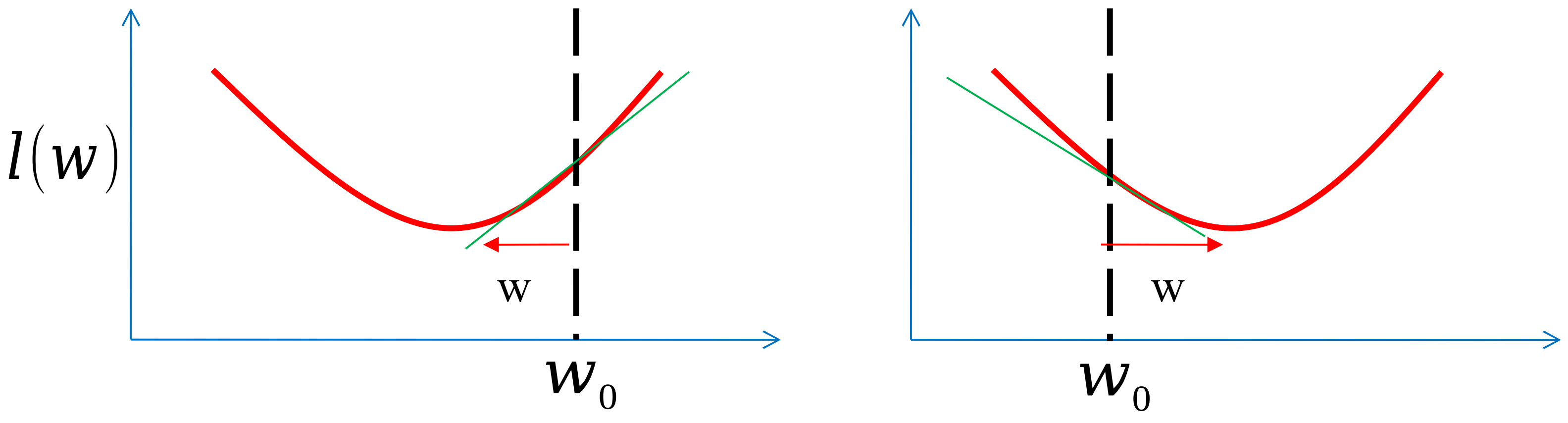
$$l(w_0, w_1) = \frac{1}{N}\sum_i \left(y_i - f_w(x_i)\right)^2 \textbf{ or } \frac{1}{N}\sum_i \left(y_i - (w_0 + x_i * w_1)\right)^2$$

- **Iterative method** for a **computational solution**
  - i. Start with some initial parameters e.g.,
  - ii. Keep changing to reduce
  - iii. Stop when desirable conditions are satisfied.
- **Questions**:
  1) What can be **initial values** of parameters ?
  2) How to **change** the parameter values , by **how much**?
  3) How do we know if the new parameters w will **reduce** ?
  4) What are the conditions to stop the iteration?

(Steepest) Gradient Descent Method

- To **minimize** we can use the following iterative method
  1. Start with an initial
  2. Keep changing w_0 to reduce : $w_{next} = w_0 \pm \Delta w$
  3. Stop when certain conditions are satisfied.

How to tell whether the **gradient** at a **point** w_0 is negative or positive?

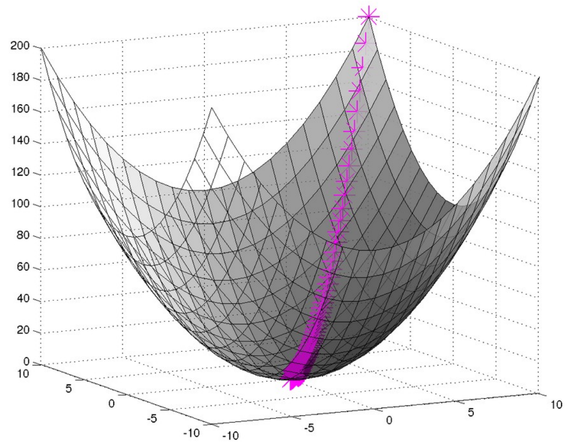


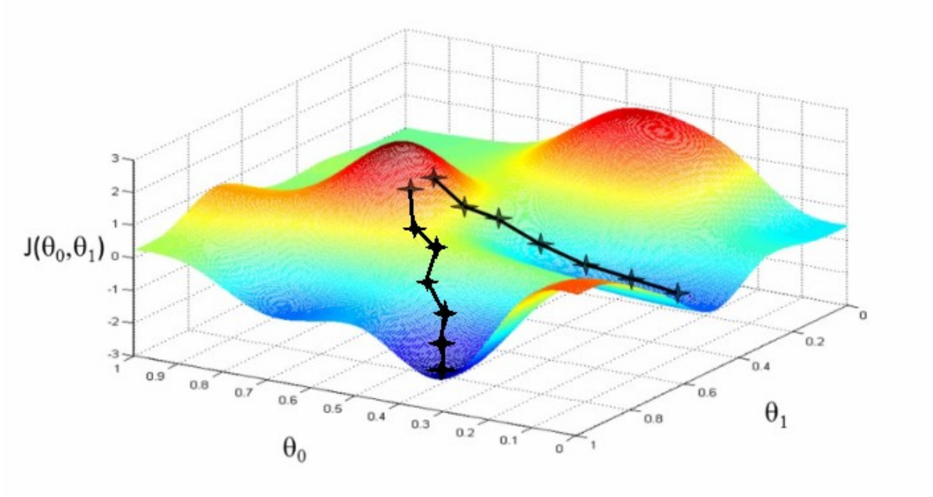
To reduce , left case: Δw should be positive, but right case: Δw negative.

How do we determine the **sign** of Δw (e.g., how do we change , subtract or add)?

Linear Regression as a *Convex* Optimization Problem

Gradient descent (also called *steepest gradient descent*) is an **iterative optimization approach** to find the **minimum** or **maximum** of a function by taking **1st order derivative**.





Gradient descent will reach a ***global* optimum** if a function has only one optimum. Otherwise, it will reach a **local optimum**.

Finding *w* by Gradient Descent Method

To **minimize** for simple regression, **minimize** and **.**

To find

- Initialize with a random value
- **Repeat** updating **until** convergence

where hyper parameter is a tuning parameter (or learning rate) to control the speed and accuracy of convergence.

To find

- Initialize with a random value
- **Repeat** updating **until** convergence

Derivatives of *to determine* Δw

- **From Loss Function**

Or instead of MSE, we can use RSS =

- The **partial derivatives** of w.r.t. and :

$=$

$=$

Simple Regression by Gradient Descent Method

Initialize with random values

Repeat until convergence {

Predicted value

Actual value

}

Each (w_0, w_1) is calculated by the residual sum of all the N examples in a training data set.

Δw

- At each iteration, **update** and **simultaneously** (independently).

- A tuning parameter is set empirically (e.g., [0.0, 1.0] as it can inflate or deflate **Δw**, impacting convergence speed and possibly accuracy.
  - Too small, slow convergence
  - Too large, fail to converge or diverge

Multiple Regression by Gradient Descent Method

$Y = X_0 + X_1 + X_2 + X_3 + \ldots + (= 1)$

```
Initialize  with k+1 random values (k number of features)
Repeat until convergence {
    for each  (j=0, ..., k) in parallel

        Update each  simultaneously (independently)
}
```

Each is calculated by the residual sum of all the N examples in a training data set.

For example, we can compute w_0, w_1, w_2:

– $(= 1)$

A Variant of Gradient Descent Method

- **Batch Gradient Descent** (we just discussed this method.)
  - At each step, use **ALL the training samples**.

$$w_j = w_j - \alpha \frac{1}{N} \boxed{\sum_i} (w^T x_i - y_i) x_{ij}$$

- **Stochastic Gradient Descent**
  - Use a portion of random training samples at each iteration.
  - This is effective with big data or for online/adaptive learning.

When to Update the Weights

- **Batch training**
  - All the training examples are used in every iteration. The weight update for each epoch for the average error. Faster convergence to a local minimum since the weights are moved in the direction that most of the inputs want them to move.
- Stochastic Gradient Descent (SGD)
  - The weights are updated for each one randomly selected input. Repeat the process until convergence. Often used when the training set is very large.
- Minibatches Gradient Descent (MGD)
  - Combination of batch and SGD
  - The training set are split into fixed-size or random batches, then update the weights for each batch.
  - The training set are then randomly shuffled into new batches and the process is repeated.
  - If the batches are small, then the global minimum may be found although at the cost of heading in the wrong direction.

The Least Square vs. Gradient Descent

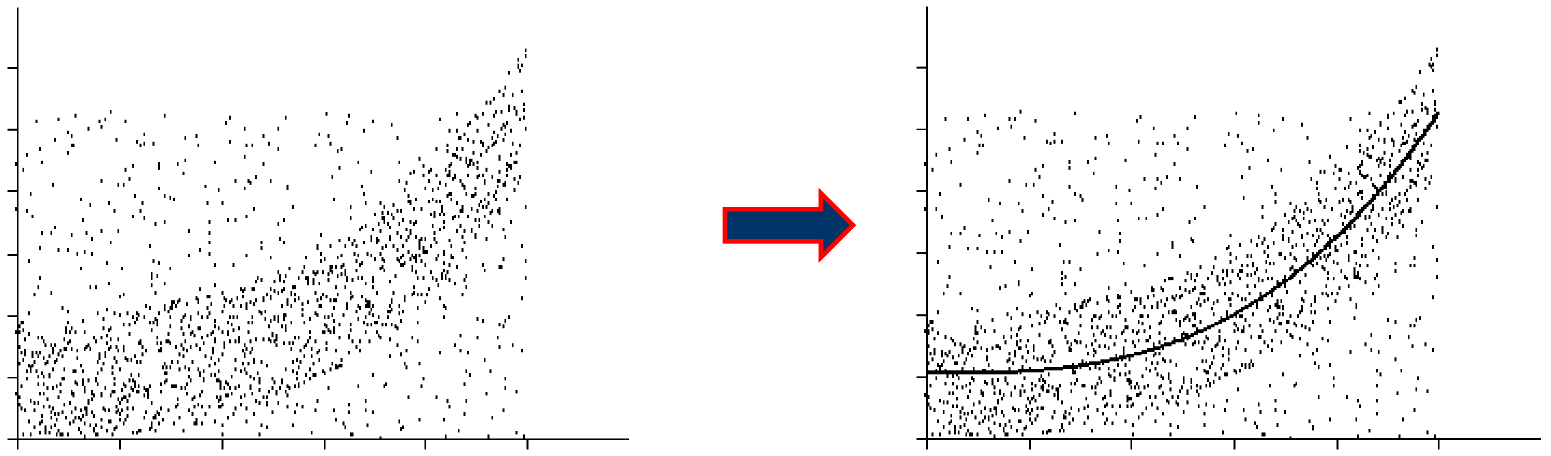
- **The least square method** (analytical solution for minimum error)
  - Very simple formula and simple implementation
  - but for big data, a large matrix requires large memory
  - but computing matrix inverse can be expensive for a large matrix
- **Gradient descent method** (iterative solution by approximation)
  - Fast even with big data (because computing is dot product of vectors)
  - *Stochastic gradient descent* is very *memory efficient*.
  - The method is easily extensible to other problems.
  - but it requires parameter tweaking:
    - How to decide convergence?
    - What is the best learning rate?
- **Most mathematical approaches for machine learning problems**
  - solved based on analytical solution (very few), estimation, approximation, and (min/max) optimization.

Class work

- Load the autompg.csv file on Canvas
- The goal is to model mpg given engine displacement
- Use Stochastic gradient descent to predict
  - the mpg of a car with engine displacement=250
  - the mpg of a car with engine displacement=600
- Experiment with different values of hyperparameters
  - max_inter, eta0
- Are your prediction results sensitive to the value of these hyperparameters?
- How do these predictions compare to those obtained analytically?

```
from sklearn.linear_model import SGDRegressor
sgd_reg = SGDRegressor(max_iter=1000, tol=1e-5, penalty=None,
eta0=0.01)
```

Transforming predictors

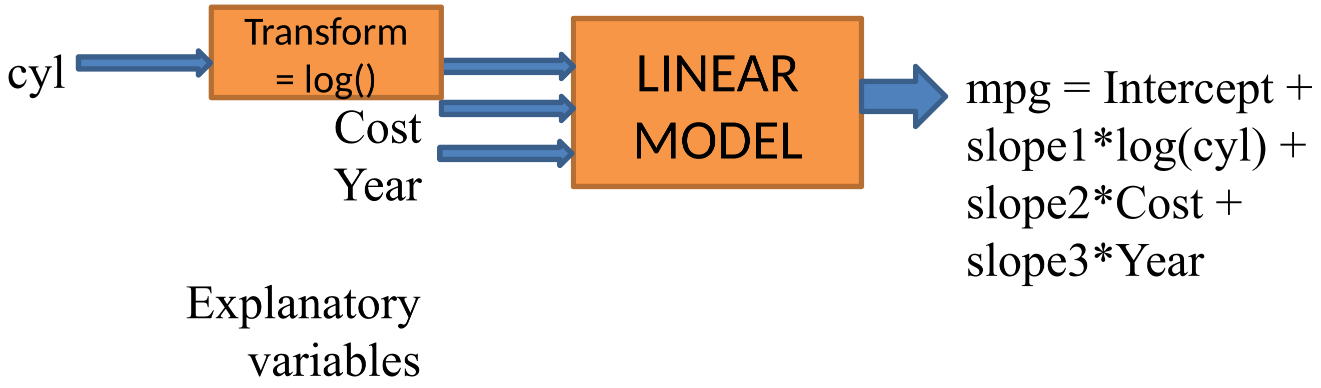


Nonlinear functions can also be fit as regressions. Common choices include Power, Logarithmic, Exponential, and Logistic, but any continuous function can be used.

Transforming predictors

- Create a new independent variable from existing variables
  - Eg.: z = x^2
- Set up new regression equation
  - y =
  - Equivalent to:
- Can even include multiple variables
  - Eg.:

Transforming predictors



mpg = Intercept + slope1*log(cyl) + slope2*Cost + slope3*Year

Class work: nonlinear transformation

For the iris dataset

- Build a linear model to predict Petal.Length given:
  - Petal.Width^2
- What is the regression equation?
- Plot the model predictions

Classwork: nonlinear transformation

For the autompg dataset (on Canvas)

- Build a linear model to predict mpg given:
  - 1/(displacement^2
- What is the regression equation?
- Plot the model predictions

Transforming predictors: categorical variables

- Transform categorical variables using one-hot encoding dummy variables
- If categorical variable takes n values, create
  - n 0/1 variables

Transforming predictors: categorical variables

- Example: Species variable in iris dataset
- Factor with 3 levels: *setosa*, *versicolor*, *virginica*
- First dummy variable for setosa
  - =1 when Species=setosa
  - =0 otherwise
- Second dummy variable for versicolor
  - =1 when Species=versicolor
  - =0 otherwise
- Third dummy variable for virginica
  - =1 when Species= virginica
  - =0 otherwise

Value			
Setosa	1	0	0
Versicolor	0	1	0
Virginica	0	0	1

Multiple parallel lines with a categorical variable

- Build a linear model to predict Petal.Length given Petal.Width and Species
- What is the linear regression equation?

2.909188

Multiple parallel lines with a categorical variable

- Build a linear model to predict Petal.Length given Petal.Width and Species
- What is the linear regression equation?
- PL = 1.211397 + 1.018712*Petal.Width + 1.697791 *Speciesversicolor + 2.276693 *Speciesvirginica

IF Species == setosa (both dummy vars = 0,0)

- PL = 1.211397 + 1.018712*Petal.Width

IF Species == versicolor (dummy vars = 1,0)

- PL = (1.211397 + 1.697791)+ 1.018712*Petal.Width
- PL = 2.909188 + 1.018712*Petal.Width

IF Species == virginica (dummy vars = 0,1)

- PL = (1.211397 + 2.276693) + 1.018712 * Petal Width

Multiple parallel lines with a categorical variable

- Build a linear model to predict Petal.Length given Petal.Width and Species
- What is the linear regression equation?
- Petal.Length = 1.21140 + (1.01871 * Petal.Width) + (1.69779 * Speciesversicolor) + (2.27669 * Speciesvirginica)
- If (Species == setosa)
  - Petal.Length = 1.21140 + (1.01871 * Petal.Width) + ~~(1.69779 * Speciesversicolor) + (2.27669 * Speciesvirginica)~~
  - Petal.Length = 1.21140 + (1.01871 * Petal.Width)
- If (Species == versicolor)
  - Petal.Length = 1.21140 + (1.01871 * Petal.Width) + (1.69779 * 1) + ~~(2.27669 * Speciesvirginica)~~
  - Petal.Length = (1.21140 +1.69779)+ (1.01871 * Petal.Width)
- If (Species == virginica)
  - Petal.Length = 1.21140 + (1.01871 * Petal.Width) + ~~(1.69779 * 0)~~ + (2.27669 * 1)
  - Petal.Length = (1.21140 + 2.27669) + (1.01871 * Petal.Width)

Visualizing the linear model with categorical variables

- Multiple lines
  - Each dummy variable creates its own line
- Example: Species variable in iris dataset
- Plot the predictions

Classwork

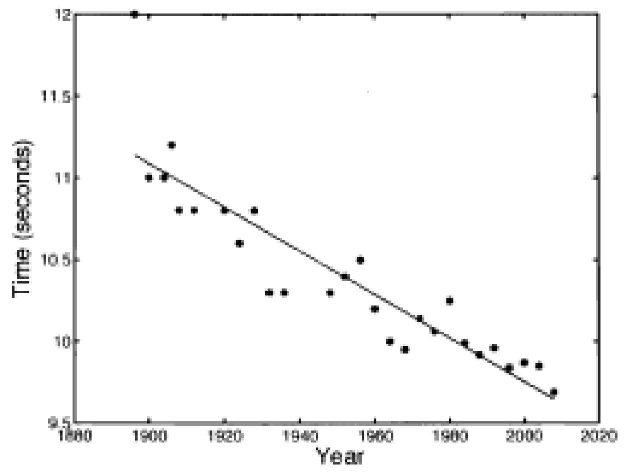
Consider the autompg.csv dataset on Canvas. The goal is to model mpg given engine displacement (a continuous variable) and number of cylinders (pretend this is a categorical variable).

- Which is the dependent variable? Which are the independent variables?
- Load the autompg.csv file on Canvas and convert cylinders variable to a factor
- Create a linear model called mod_displ_cyl of mpg vs. displacement and cylinders.
- Give the model equations relating mpg with displacement and cylinders.
- Plot mpg vs. displacement and overlay the best fit model.

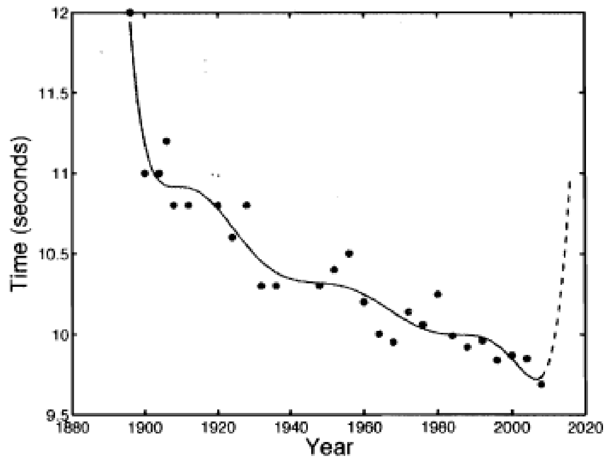
REGULARIZING COMPLEX MODELS TO AVOID OVERFITTING

Choosing the Best Model

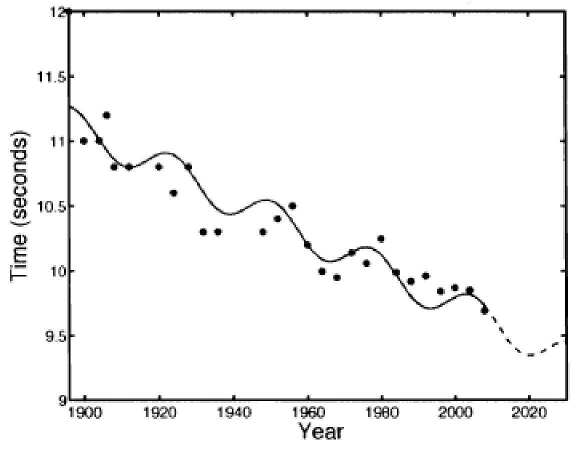
(a) A linear model with $L = 1.358$



(b) An 8th order polynomial model with $L = 0.459$

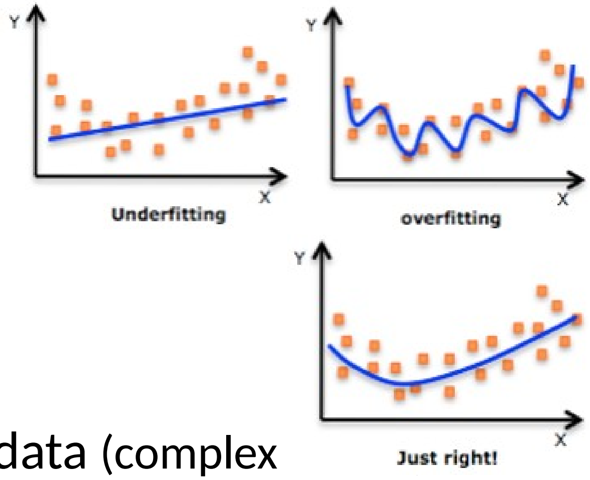


(c) A customized model with $L = 1.1037$



- **Which model <u>looks</u> the best?**
  - The models with increasing complexity (higher order model) fit better, resulting in lower RSS values as it gets closer to the training data, but some predictions can be wildly inaccurate for even small deviations from the training data instances.

Over-fitting and Under-fitting



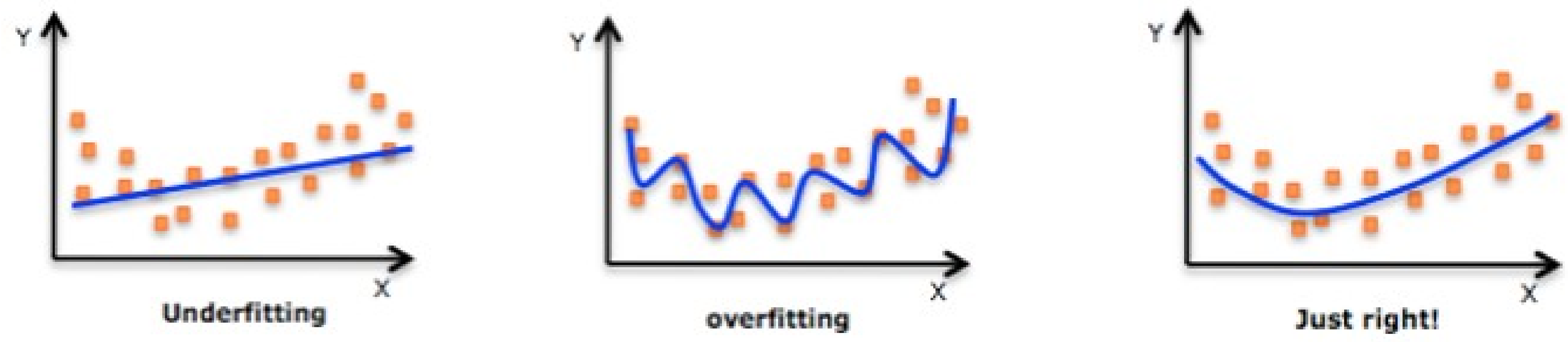
- Each model is a generalization from the **training data**
- Over-fitting: A model fits too closely to the training data (complex but the quality of the predictions can be poor.
  - High variance: **Low** training error and **high** generalization error, causing an algorithm to model the random noise or memorize the training data.
  - Occam's Razor: Given two models with the same generalization errors, the simpler model is preferred over the more complex model.
- Under-fitting: A model cannot adequately capture the underlying structure of the data.
  - High bias: **High** training error and **high** systemic error, causing an algorithm to miss the relevant relations between the independent and dependent variables.

Bias-Variance trade-off

- A model’s generalization error can be expressed as the sum of three types of errors:
- Bias
  - Error due to wrong assumptions, such as assuming that the data is linear when it is actually quadratic.
  - A high-bias model is most likely to **underfit** the training data.
- Variance
  - Model’s sensitivity to variations in the training data.
  - A model with many degrees of freedom (e.g., high-degree polynomial) is likely to have high variance and thus **overfit** the training data.
- Irreducible error
  - Noise in the data. Nothing that the model can do anything about.

Bias-Variance trade-off

- Increasing a model's complexity will typically increase its variance and reduce its bias.
- Conversely, reducing a model's complexity increases its bias and reduces its variance.
- Tradeoff is to determine the optimal model complexity such that it is able to generalize well without over-fitting.
- This is a challenging task!



Underfitting and Overfitting Problems

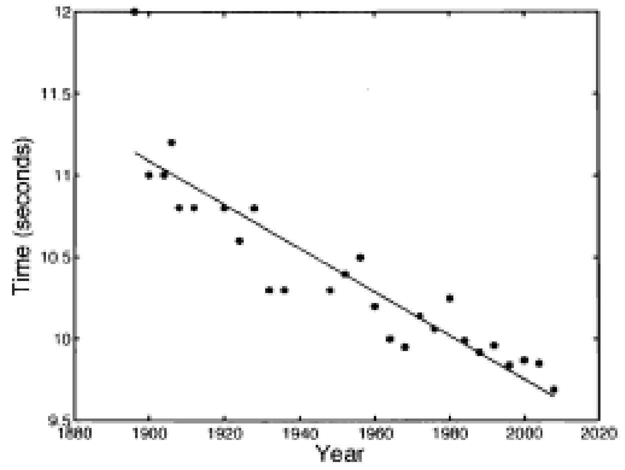
- **Common causes: recommended solutions**
  - Data quality and random noise: preprocess the data
  - Lack of representative examples: use more training data
    - How much data are appropriate? A rule of thumb for statistically reliable result: At least 5 training examples for each dimension and >30 examples
    - Generate more training data using semi-supervised learning or generative method
  - Too many variables: reduce the dimension and select the core features
  - Algorithms with poor generalization capability: use a better algorithm

- **Common causes of underfitting: recommended solutions**
  - Too simple models: make the models complex

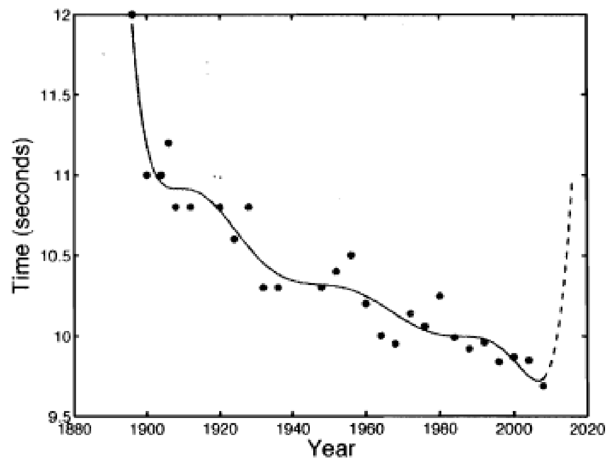
- **Common causes of overfitting: recommended solutions**
  - Too complex models: simplify the models (**regularization**), early stopping, dropout, pruning, Bayesian priors, etc. and evaluate its performance on unseen data (testing/validation data).

Complex Models

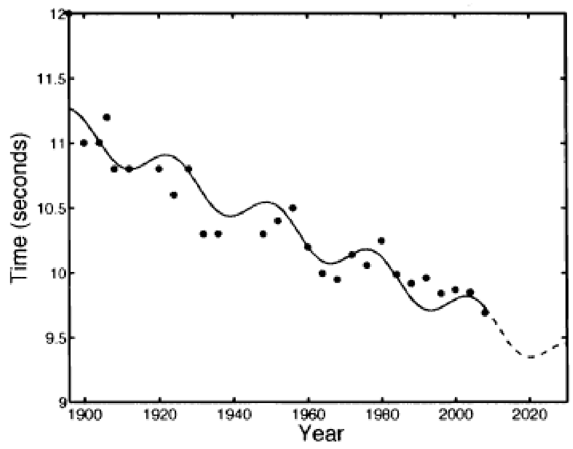
(a) A linear model with *l* = 1.358



(b) An 8th order polynomial model with *l* = 0.459



(c) A customized model with *l* = 1.1037



- **What's going on with higher order models?**
  - The models with increasing complexity (higher order model) fit better the data and result in lower RSS values (*low error in training*) and but possibly matching *even the noise in the data.* Therefore, *some predictions may be very inaccurate* with small deviations in the new examples (not properly generalized).
  - **To correct this problem**, **add** a penalty term in the loss function (regularization).

Regularizing Models

- ***Regularizing*** a model:

  M(w) + λR(w) where λR(w) is a regularized term (penalty term or shrinkage factor);

  λ can be any real value >0; w are shrinking coefficients.

  – Why penalize the magnitude of coefficients w?

- **Regularizing** (**note**: this is a loss function)
  – LASSO (Least Absolute Shrinkage and Selection Operator):

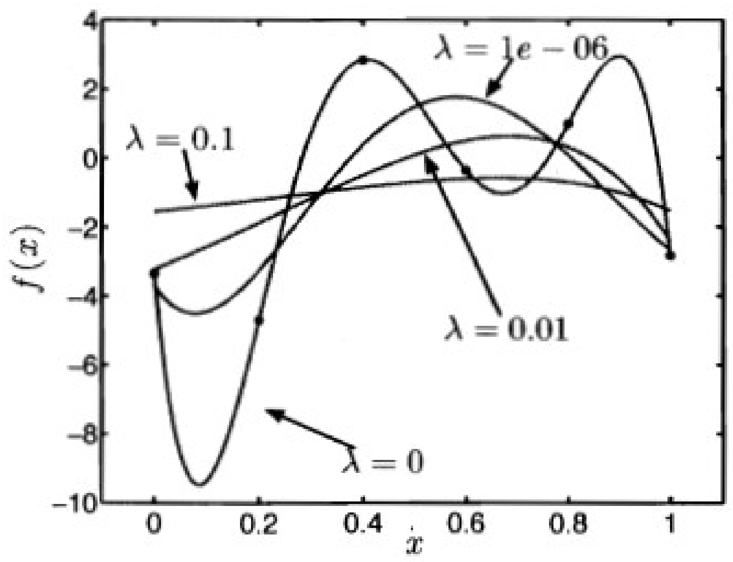
    ($\lambda||w_i||_1$, L1 regularization)

  – Ridge: ($\lambda||w_i||_2$, L2 regularization)

  – Elastic net: LASSO + Ridge

- **The role of the tuning parameter λ**
  – If λ = 0, linear regression (without regularization).
  – The larger λ, more aggressively penalization is, the coefficient is close to zero, resulting in a simpler model.
  – λ should be chosen wisely (using cross-validation) as **it controls** the tradeoff between penalizing not fitting the data and penalizing overly complex models.

The Effects of the Tuning Parameter λ

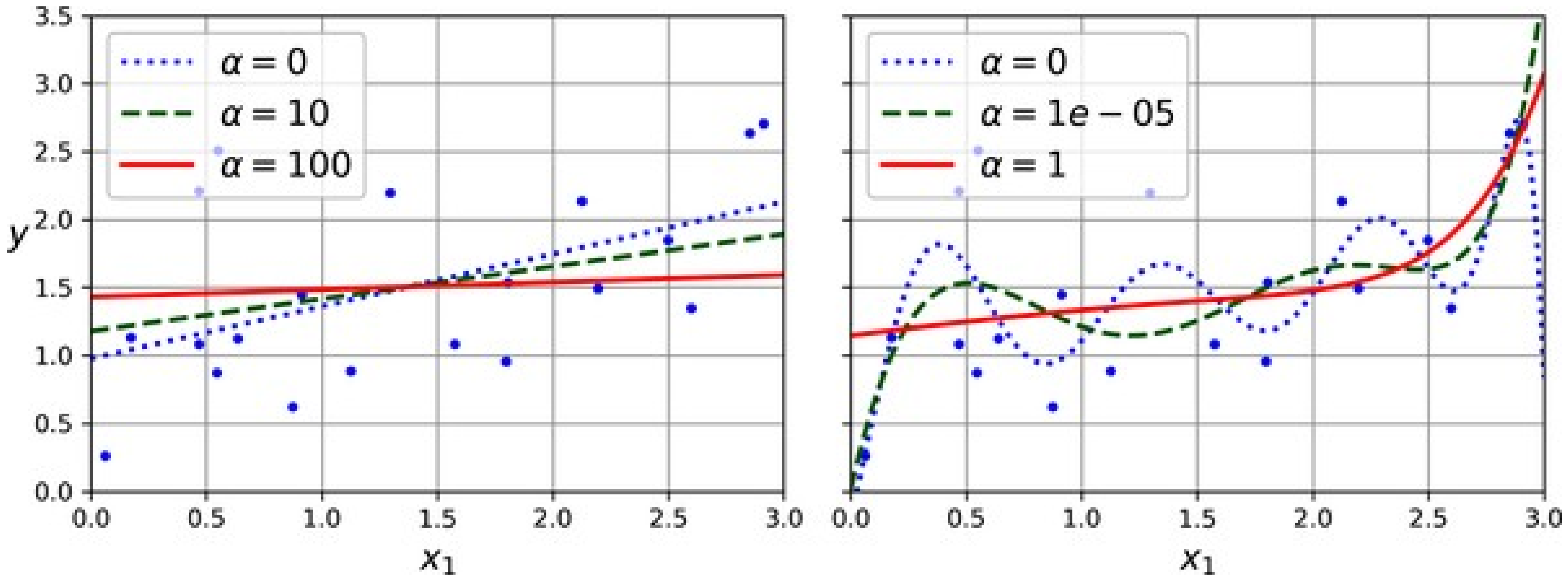


The **higher** the sum in w, the **more complex** the model. To penalize the complex model, keep the w value lower by increasing **λ**.

Varying the regularization parameter **λ** for a 5th order polynomial function

- If **λ = 0**, we get the original solution.
- If λ is too small (1e-06), the model is likely to be too complex, getting closer to the general shape of the exact 5th order polynomial function.
- If λ is too large, less complex but may not capture any useful structure of the data.
- We need to determine the best value of λ.

Ridge regression



Linear (left) and a polynomial (right) models with various levels of ridge regularization

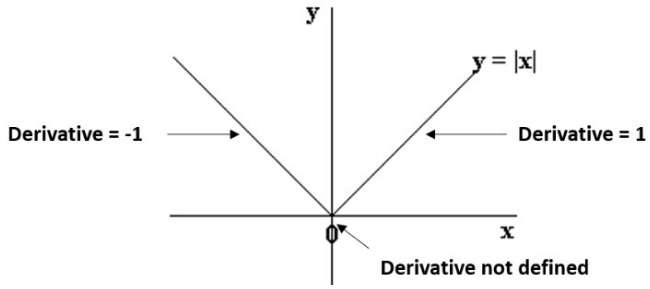
Ridge Regression

- **Ridge regression using the least square method**
  - , solving for w: = 0
  - From , the regularized least square solution is:

    (analytical solution)

- **Ridge regression using the gradient descent method**
  - Adding the regularization term to
  - +]
  - +

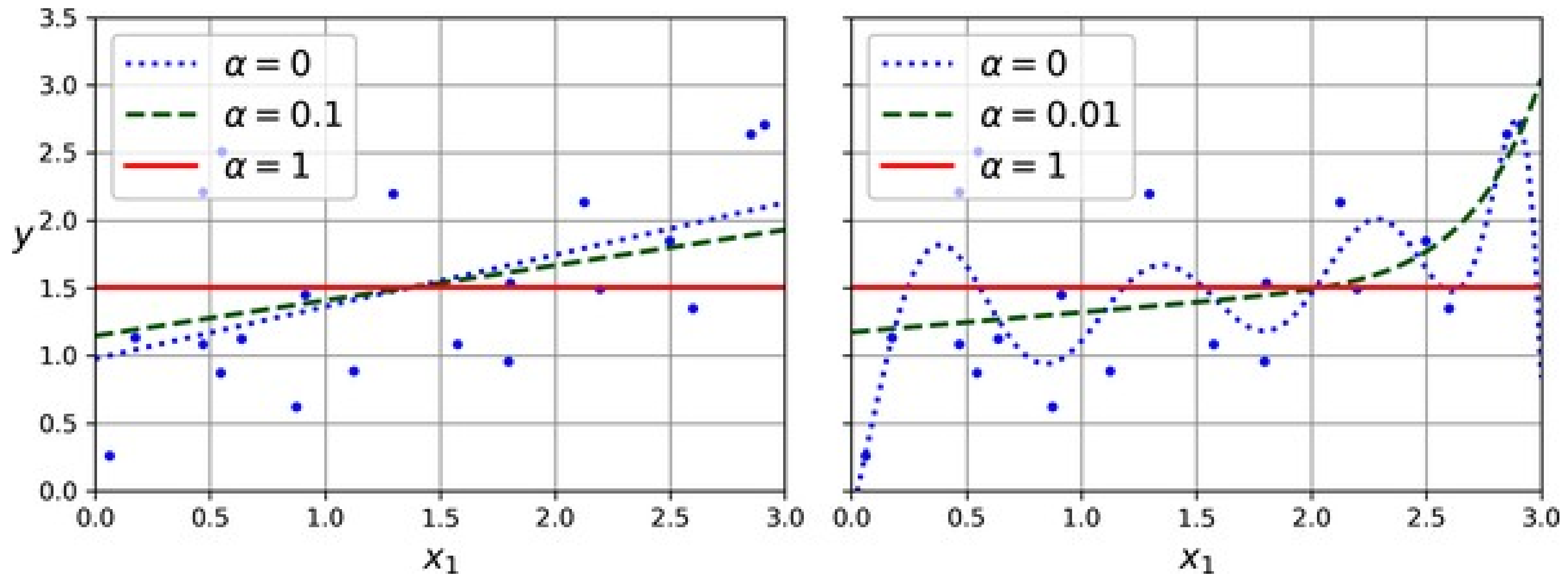
LASSO Regression



- LASSO regression:
  - No derivative at $x = 0$, we use "coordinate descent" that optimizes each parameter separately, holding all the others fixed (based on the concept of subgradients).
  - Unlike Ridge, no analytical solution since the solution is nonlinear in y.

- **Derivation of LASSO regression by Coordinate Descent**
  - RSS = =>
  - Let , then since (if normalized).
  - The regularized term, -λ when <0, [-λ, λ] when =0, and λ when >0 (subgradients)
  - **For minima of** , let **+ = 0**, then solve for :
    - Case < 0: = 0, **= +** . Need <
    - Case = 0: [-2 -λ, -2 +λ] to contain **0**.   Need ≤ ≤
    - Case > 0: = 0, **= -** . Need >

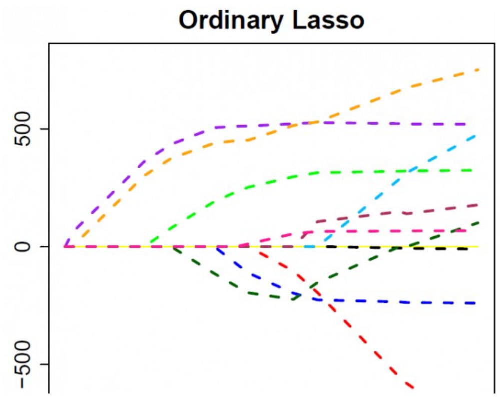
LASSO regression

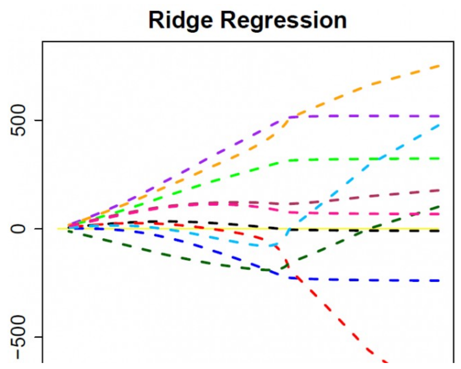


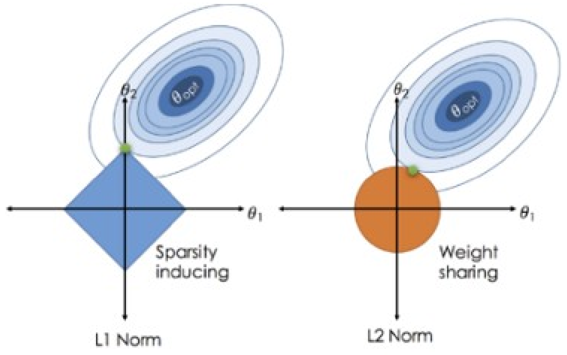
Linear (left) and a polynomial (right) models with various levels of LASSO regularization

Ridge vs LASSO

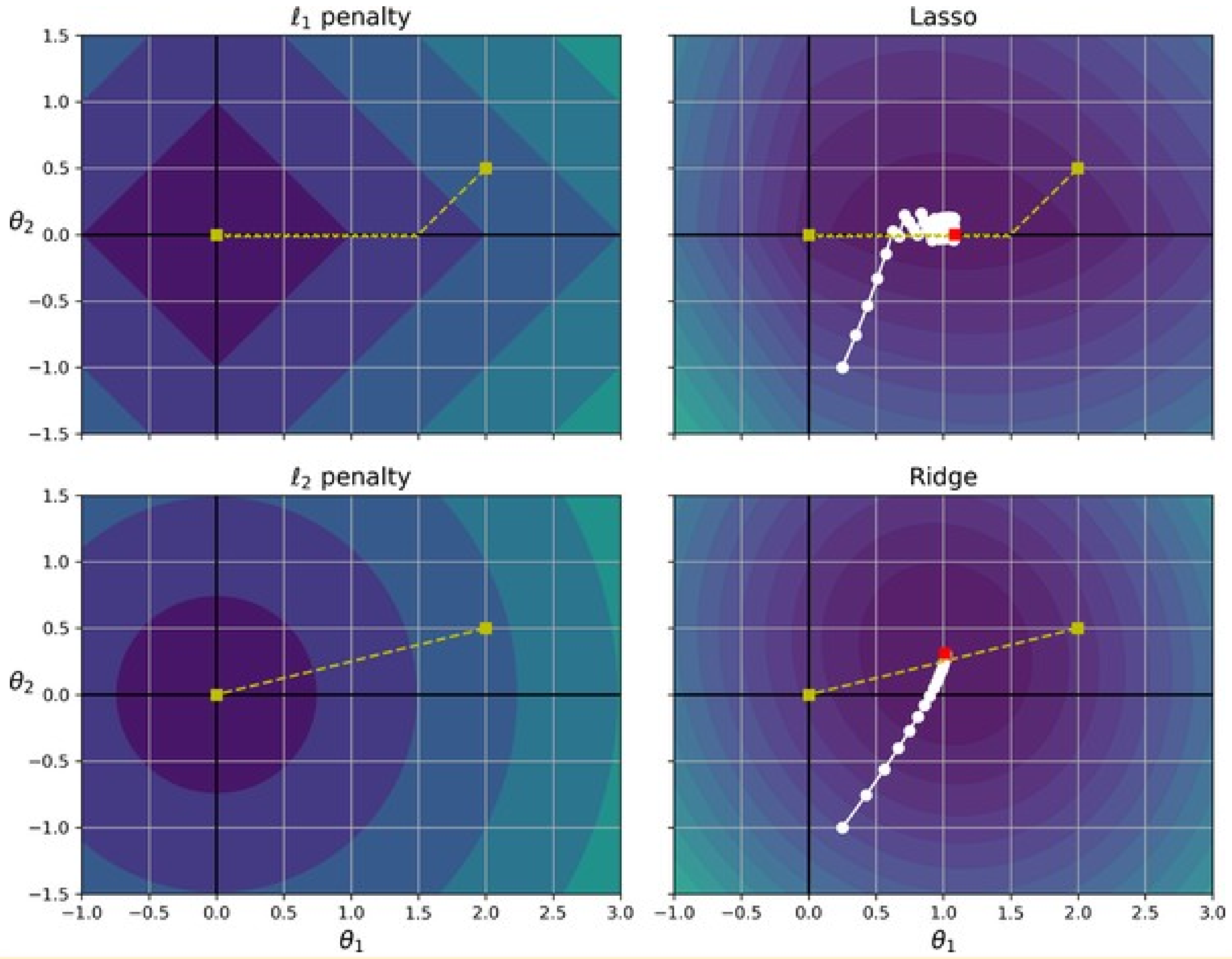
- LASSO and Ridge regressions







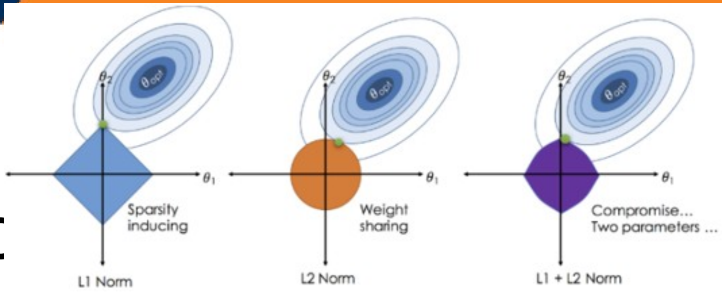
Ridge vs LASSO



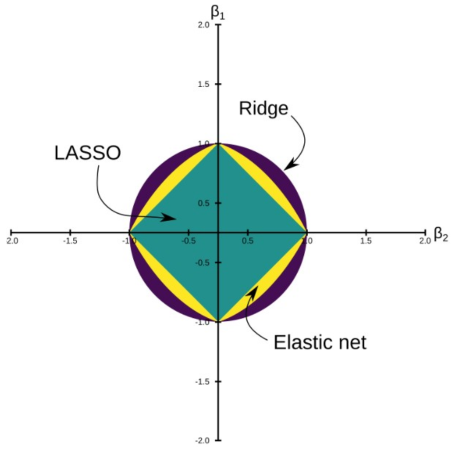
Ridge vs LASSO

- **The effects of λ**
  - As λ increases, **Ridge** can shrink the coefficients asymptotically close to zero, while LASSO can shrink them all the way to zero, eliminating the less important features.
    - If the squared sum in LASSO hits one of the corners, the coefficients are shrunk to zero.
  - For the same λ, **LASSO** has much smaller coefficients and higher RSS compared to Ridge. Many coefficients are zero even for very small λ, “sparsity”.
  - One way to find a proper λ, start with a relatively large λ, then decrease it slowly.
  - The coefficients of correlated predictors are similar in Ridge while one for the correlated predictors has a larger coefficient and the rest are nearly zeroed in LASSO.
  - LASSO works well for a model with high multicollinearity and can be useful for feature selection, but it may be too dependent on data and thus unstable.

- **Overall neither one is better**. **How about LASSO + Ridge?**



Elastic Net Regressic



- **Elastic net** (for both feature selection and regularization)
  - For α = [0,1], $\lambda \geq 0$,

,

    - When α = 1, Elastic net $\equiv$ LASSO.
    - As α shrinks toward 0, it approaches Ridge.
    - For other values of α, the penalty term interpolates between L1 and the L2 of w.

- **Update rule**
  - Find the coefficients of Ridge, then perform the LASSO on the Ridge regression coefficients to shrink the coefficients.

Class work

- Load the iris dataset
- The goal is to model Petal.Width given Petal.Length, Sepal.Length, and Sepal.Width
- Use LASSO regression and experiment with different values of the hyperparameter alpha
- What are the coefficients?
- What is the linear regression equation?
- Is your equation sensitive to the value of alpha?

```
from sklearn.linear_model import Lasso
lasso_reg = Lasso(alpha=0.1)
lasso_reg.fit(X, y)
```

Logistic Regression

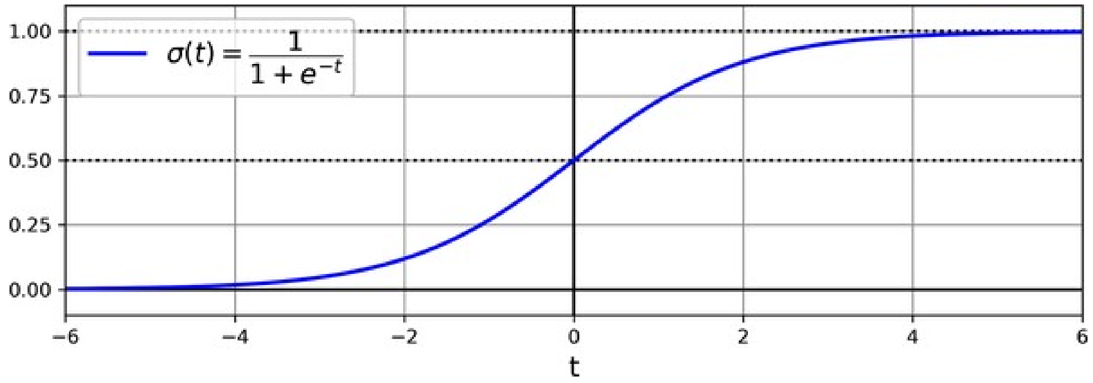
- Modify linear regression to do classification
- Model estimates the probability that an instance belongs to a class.
  - If the estimated probability > 0.5, then predict that the instance belongs to that class (positive)
  - Otherwise (p < 0.5), predict that it does not belong (negative)
- Binary classifier

Logistic Regression

- Model computes
  - a weighted sum of the input features
  - Followed by the logistic of this weighted sum
- Logistic function:
  - A sigmoid function that outputs a number between 0 and 1 for any input

$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma(\boldsymbol{\theta}^\top \mathbf{x})$$

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



Class work

- Load the iris dataset
- The goal is to classify instances as Species="virginica" or not given Petal.Length, Sepal.Length, Sepal.Length, and Sepal.Width
- Use logistic regression and experiment with different combinations of the features
- Use 3-fold cross-validation to evaluate models
  - Which combination of features gives the highest precision/recall?

```
from sklearn.linear_model import LogisticRegression
X = iris.data[["petal width (cm), ???"]].values
y = iris.target_names[iris.target] == 'virginica'

log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
log_reg.predict(..., ...)
```

Textbook code

- Textbook code
- Textbook code on Google Colab
- Open `04_training_linear_models.ipynb`

Acknowledgement

- Many slides from Dr. Christopher Ryu
- Content based on “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow,” Aurélien Géron, 3rd Edition (October 2022), O'Reilly Media, Inc.