Supervised Learning

Week 7
Linear Regression
Logistic Regression

Linear Regression

Features or variables
Linear regression formulation Generalisation and complexity

Relevance and Covariance among features or variables

- We can measure the linear relationship between the variable x (could be of many dimensions) and output y (y is only one dimensional), using covariance.
- To put it simply, covariance measures the amount of information a specific x_i can provides for x_i. $Cov(x, y)$ is calculated as:

$$Cov(\mathbf{x}, y) = \frac{\sum_{i=1}^{n}(\mathbf{x}_i - \bar{x})(y_i - \bar{y})}{n-1}$$

where, $\bar{x}$ is the mean of x_i and $\bar{y}$ is the mean of y_i

- Three possible values for $Cov(x, y)$:
  - $Cov(\mathbf{x}, y) > 0 \quad \rightarrow \quad \mathbf{x}$ and y are positively correlated; if x is increasing, y is increasing.
  - $Cov(\mathbf{x}, y) < 0 \quad \rightarrow \quad \mathbf{x}$ and y are inversely correlated; if x is increasing, y is decreasing.
  - $Cov(\mathbf{x}, y) = 0 \quad \rightarrow \quad \mathbf{x}$ and y are independent.

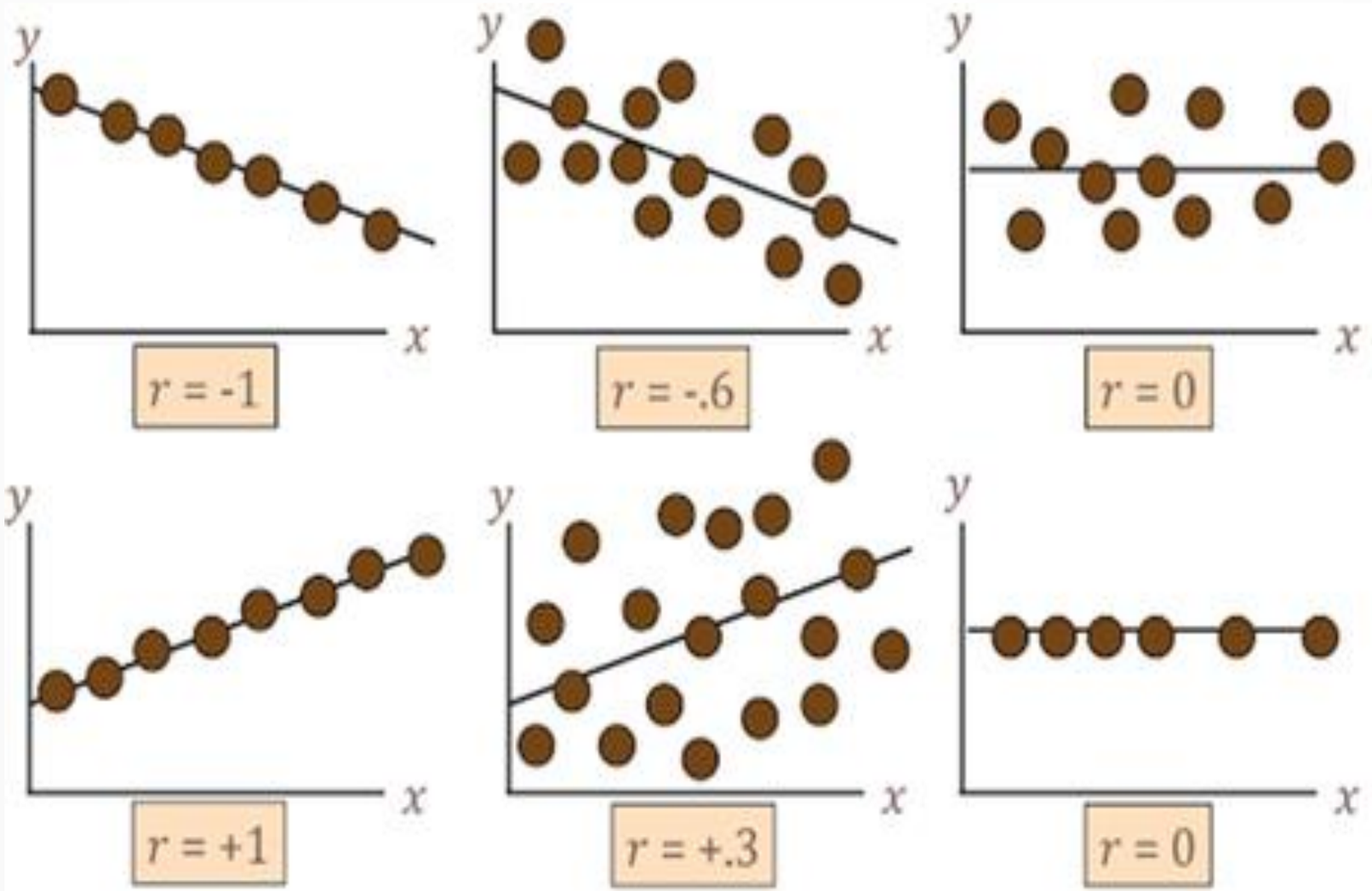
Relevance and Covariance:
Pearson's Correlation Coefficient

- Pearson's Correlation Coefficient is a measure of the linear correlation between two variables x and y

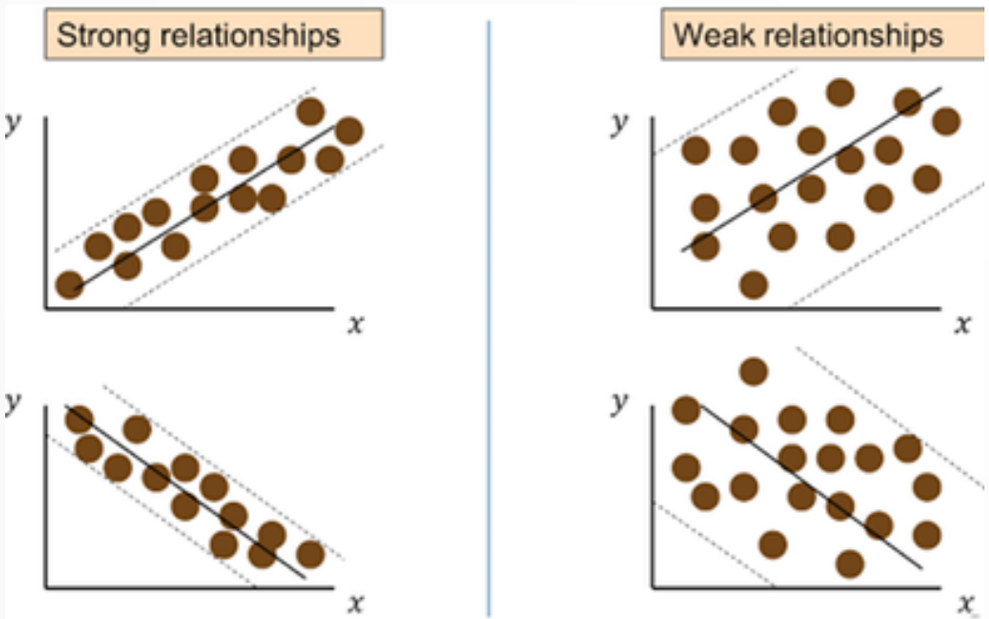
$$r = \frac{cov(x, y)}{\sqrt{var(x)var(y)}}$$

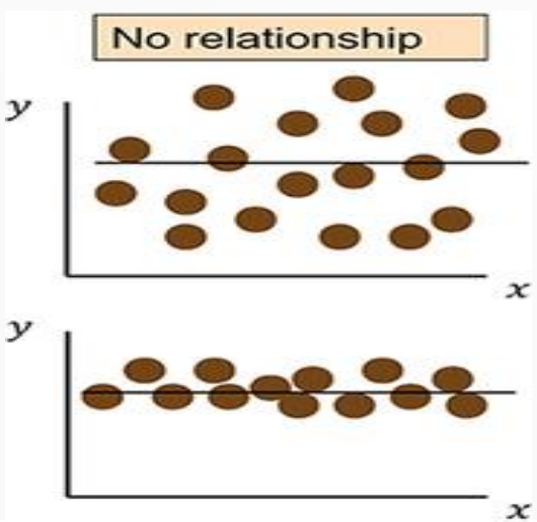
- It has a value between +1 and −1, where
  - 1 or closer values indicate positive and strong linear correlation,
  - 0 or closer values indicant no or weak linear correlation
  - −1 or closer values show negative and strong linear correlation.
- Let's say $y = x^2$, should we expect high values of Pearson's Correlation Coefficient? No! Because they are not linearly related. So notice Pearson's Correlation Coefficient is all about linear relationship.

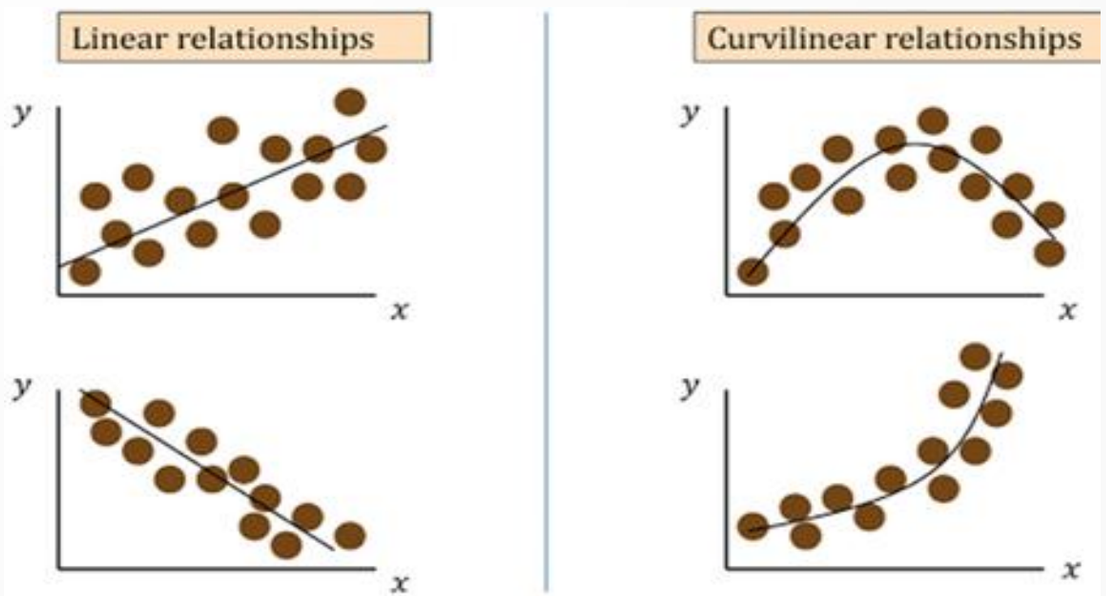
Relevance and Covariance:
Pearson's Correlation Coefficient…



Relevance and Covariance:
Pearson's Correlation Coefficient…







Linear Regression

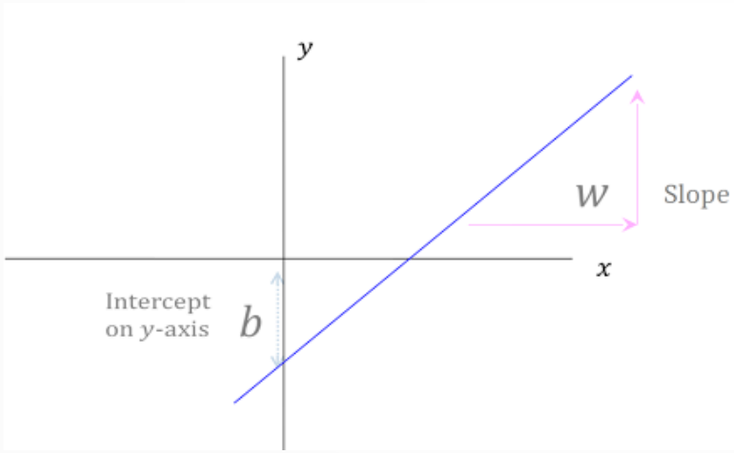
- Linear regression attempts to model the relationship between two variables by fitting a linear equation to the observed data.
- In linear regression, training data is of the form of

$$\mathbf{x}_i, y_i, \quad i = 1, ..., n$$

- For each data point (feature vector) x_i ,there is an output y_i, which can be any real-valued number.
  - Looking for particular relationship between a feature and the output.

Linear regression formulation

- Let's define the linear equation as equation of a line: $y = h(\mathbf{x}) = w\mathbf{x} + b$



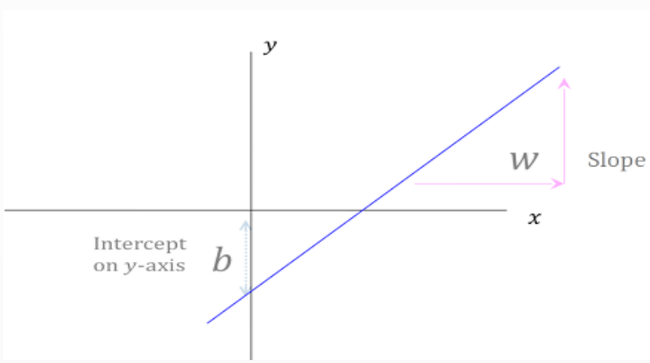
- In linear regression, we would like to find a line similar to h so that the obtained model allows us to summarize and study relationships between two continuous (quantitative) variables.

- So how can we find this line?

Linear regression formulation: Linear hypothesis

$$y = h(\mathbf{x}) = w\mathbf{x} + b$$

- This line has two parameters w and b.
  - By having these two, we can find our proper line based on the data.
  - Then by having a point like x_i we can predict the value of $\hat{y}(x_i)$, which is an estimation of $y(x_i)$.
    - The line predicts $\hat{y}(x_i)$ for x_i.



- What if x is not just a single dimension value?

Linear regression formulation: Linear hypothesis...

- In that case we write the linear regression as:

$$\begin{aligned}\hat{y}(\mathbf{x}_i) &= w_0 x_{i0} + w_1 x_{i1} + w_2 x_{i2} + \ldots + w_d x_{id} \\ &= b + w_1 x_{i1} + w_2 x_{i2} + \ldots + w_d x_{id}\end{aligned}$$

  - where x_{i0} is just a dummy feature with the value of $x_{i0} = 1$ and also $w_0 = b$.

- Same as the single dimension form except
  - the multiplication of w and x is done in all d dimensions.

Linear regression formulation: Linear hypothesis…

- Using the vector notation, we can write the above as $\hat{y}_i = \mathbf{x}_i^T w$ using $d + 1$ dimensional vectors (why?).
- For i=1,…,n we have: $\hat{y}_1 = \mathbf{x}_1^T w, \quad for\ i = 1$

  $\hat{y}_n = \mathbf{x}_n^T w, \quad for\ i = n$
- Therefore, collectively we can write: $\hat{y} = Xw$ where, $\hat{y} = [\hat{y}_1, \ldots, \hat{y}_n]^T$ and $w = [w_0 \ldots w_d]^T$

Linear regression formulation: How can we fit this line to datapoints?

- By miminmising empirical risk!! How?

- The difference between what we predicted and the true value or output of that point, is considered to be the error.
- We show the error for data point i with e_i: $e_i = y_i - \hat{y}_i$
- The linear model seeks for minimizing the empirical risk $R(w)$ via the squared loss $(y_i - \hat{y}_i)^2$ as:

$$\min_w \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Linear regression formulation: How can we fit this line to datapoints?

- We can also rewrite the formula as:

$$\min_w \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i^T \mathbf{w})^2$$

since, $\hat{y}_i = \mathbf{x}_i^T w$.

- The above formulation is just the mean of the square error function, which is defined as the loss function for linear regression:

$$L(y_i, \mathbf{x}_i^T \mathbf{w}) = (y_i - \mathbf{x}_i^T \mathbf{w})^2$$

- But how can we solve this minimisation problem?

Linear regression formulation: How can we fit this line to datapoints?

- Same as other optimsation problems with a closed form function:
  - we can take the derivative of the error function with respect to w

  and
  - equate it to 0.
- Then we are able to find the w which can minimize this error.
- why the derivative is with respect to w and not x?

Linear regression formulation: How can we fit this line to datapoints?

- The answer is really easy
  - x is the feature vector and we do not want to find best fit feature value!!!
  - Rather, we are looking for proper w to fit the line on feature vectors.
- So by taking the derivative of the error function and equating it to 0, we will find that: $$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$
  - the matrix (right hand side) is known as Moore-Penrose pseudo-inverse of the matrix X and often denoted as $X^\dagger$.

Linear regression summary

- For Linear regression we use squared loss functions, i.e. $L(y_i, \mathbf{x}_i^T \mathbf{w}) = (y_i - \mathbf{x}_i^T \mathbf{w})^2$

- W can be learned in single step using closed form solution:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- Once, w is calculated, for any X_i the predicted value can be calculated as: $\hat{y}_i = \mathbf{x}_i^T w$

Logistic Regression

Logistic regression

- When to use logistic regression?
  - For categorical dependent variable.
    - y_i is categorical e.g. for binary 0 or 1

- Still to fit a linear function to predict y_i from $\boldsymbol{X}_i$
  - Using training data to estimate the regression coefficient vector w.
- Instead of squared error function (used in linear regression), we will use maximum likelihood estimation (MLE) to estimate w.
  - The likelihood function of w using data (x_i, y_i) is given as:

$$l(\mathbf{w}) = \frac{1}{1 + exp(-y_i \mathbf{x}_i^T \mathbf{w})}$$

In the above, we assuming a Bernoulli distribution on y_i because of the binary forms of the outputs.

Training a logistic regression model

- Assuming training data with n independent instances $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)\}$ have joint likelihood as:

$$l(\mathbf{w}) = \frac{1}{1 + exp(-y_1 \mathbf{x}_1^T \mathbf{w})} \times ... \times \frac{1}{1 + exp(-y_n \mathbf{x}_n^T \mathbf{w})}$$
$$= \Pi_{i=1}^{n} \frac{1}{1 + exp(-y_i \mathbf{x}_i^T \mathbf{w})}$$

- So the Joint likelihood function while having n independent samples using training data is the multiplication of the likelihood of each point.

$$l(\mathbf{w}) = \Pi_{i=1}^{n} \frac{1}{1 + exp(-y_i \mathbf{x}_i^T \mathbf{w})}$$

- Maximum likelihood estimation method maximises $l(w)$ with respect to w.

Training a logistic regression model: Logistic Loss Function

- Maximising likelihood is equivalent to
  - maximising log of the likelihood function
    - because both provide same solution for w.
- Remember by taking the log of the function you are still able to find the maximum or minimum of the function since
  - the logarithmic functions are monotone increasing functions.
- Thus, Log of the likelihood function can be written by taking the log of $l(w)$ as:

$$L(\mathbf{w}) = log\ l(\mathbf{w}) = -\sum_{i=1}^{n} log(1 + exp(-y_i \mathbf{x}_i^T \mathbf{w}))$$

- The above function is also called the Logistic Loss function $L(w)$

Training a logistic regression model: Logistic Loss Function…

- Maximising $\log l(w)$ is equivalent to minimising $-\log l(w)$, which brings us to the following minimisation problem:

$$\min_{w} \sum_{i=1}^{n} log(1 + exp(-y_i \mathbf{x}_i^T \mathbf{w}))$$

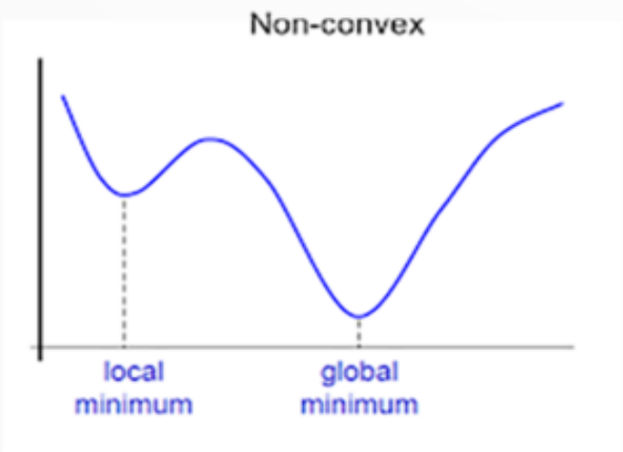
- But How to minimise the above function with respect to w?

Training a logistic regression model: Logistic Loss Function…

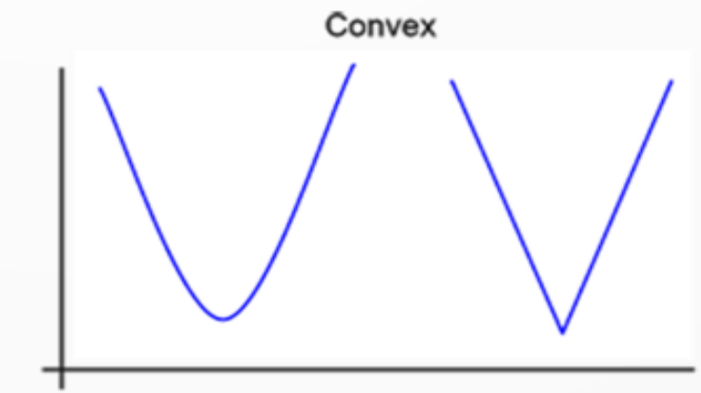
- The usual approach is to take derivative and equate it to zero to solve for w.
  - However, in this case, the solution does not have a closed form.
  - Therefore, we need to solve the problem iteratively.
- It is important to remember that sometimes, we can derive a closed form formula for the minimiser (e.g. linear regression) meaning we can compute the minimiser in one step.
  - Otherwise, we take multiple steps iteratively to reach to the minimum (e.g. logistic regression and Kmeans).
  - So in this case, we would like to perform Coordinate-wise Gradient Descent Optimisation.

Training a logistic regression model: Computing the minimum

- But he question still remains, how can we find the minimum?
- Before answering this question
  - Let's recall difference of two types of functions Convex and Non-convex.

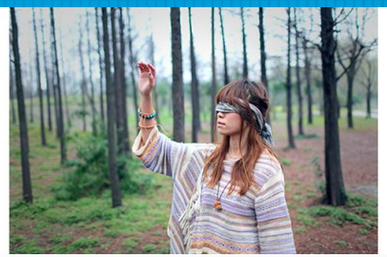


Example: Kmeans objective function



Example: Objective functions of linear and logistic regression

Training a logistic regression model: Computing the minimum…



- Strategies for finding your way forward (figure)
  - Assume, you're blindfolded, but you can see out of the bottom of the blindfold to the ground right by your feet.
  - I drop you off somewhere and tell you that you're in a convex shaped valley and escape is at the bottom/minimum.
- How do you get out?

Training a logistic regression model: Computing the minimum…

- The most simple way is to look for steepest slopes!
  - So basically you start walking and you look for going down through slopes, or the steepest slopes.
- In math, we call the slopes, derivatives!
- Two popular methods when we can compute gradient (derivatives) of the objective function:
  - Gradient descent (uses first derivative)
  - Newton's method (uses second derivative)

Training a logistic regression model: Computing the minimum...

- Gradient Descent (uses first derivative):
  - To minimise $L(w)$, we use the iterative update:

$$w_{t+1} \leftarrow w_t - \eta_t \frac{\partial}{\partial w} L(w)$$

  - You choose your new position of *w* by $w_t - \eta_t \frac{\partial}{\partial w} L(w)$
  - which is made of **η** (your step size) and $\frac{\partial}{\partial w} L(w)$ the slope or derivative of the function.
  - So, it simulates the same concept we defined with a convex shaped valley and escape at the bottom/minimum.

Training a logistic regression model: Computing the minimum…

- Newton's method (uses second derivative):
  - To minimize $L(w)$, we use the iterative update

$$w_{t+1} \leftarrow w_t - H^{-1} \frac{\partial}{\partial w} L(w)$$

    - Where H is the Hessian matrix with H_{ij} being $\frac{\partial^2}{\partial w_i \partial w_j} L(w)$
  - So Newton's method is an iterative method for finding the roots of a differentiable function.

Training a logistic regression model: Computing the minimum…

- Remember, Gradient descent maximises a function using knowledge of its derivative.
- While Newton's method, a root finding algorithm, maximises a function using knowledge of its second derivative.
  - This can be faster when the second derivative is known and easy to compute.
  - However, the analytic expression for the second derivative is often complicated or intractable, requiring a lot of computation.

Training a logistic regression model: Coordinate-wise Gradient Descent Optimisation

- Now, lets get back to Coordinate-wise Gradient Descent Optimisation.
- In order to fulfil this task first randomly initialise w.
- Fix all the variables except for one, i.e., for each j, optimise w_j fixing $w_1, ..., w_{j-1}, w_{j+1}, ..., w_d$
- Then, we need to minimise the objective function with respect to w_j using Gradient descent as:

$$w_j \leftarrow w_j + \eta \frac{\partial}{\partial w_j} \left(\sum_{i=1}^{n} log(1 + exp(-y_i \mathbf{x}_i^T \mathbf{w})) \right)$$

Training a logistic regression model: Coordinate-wise Gradient Descent Optimisation

- Similarly optimise for other w_j's (j is from 1 to d).
- Continue until the objective function stops changing.
- The solution is unique (due to the convexity of the objective function), irrespective of the initialisation of w.
- But, there is always a chance of getting stuck in local minimums rather than the global one.
  - run the Gradient descent with many different random initialization
    - escaping from the local minimum.

Logistic regression summary

- For Linear regression we use squared loss functions, i.e.

$$L(y_i, \mathbf{x}_i^T \mathbf{w}) = log(1 + exp(-y_i \mathbf{x}_i^T \mathbf{w}))$$

- W can be learned iteratively since closed form solution is not there.
  - Gradient decent
  - Newton's method

Model Complexity

Model Complexity

- Overfitting happens when we are finding an over complex model on the data.
- On the other hand, underfitting is the result of an extremely simple model
-

Overfitting: How it happens?

- We train a model and find that it is explaining the data better now but still not good enough.
  - Add more variables.
    - Improves the model
      - Leads to overfitting, i.e. it will probably have poor prediction on unseen data.
      - It has learnt too much specifics of training data and has probably learnt the background noise.

Model Complexity: Bias Variance Decomposition

- Let us assume our data (x, y) has the true relation $y = f(x) + \epsilon$, where ϵ is a measurement noise in y with mean zero and variance σ_ϵ^2
- Also assume that we are fitting a hypothesis function (or model) $h_D(x)$ using dataset D.
- Then the expected loss (or risk) has three components.

$$Risk = \{E_D[h_D(x) - f(x)]\}^2 + E_D[\{h_D(x) - E_D[h_D(x)]\}^2] + \sigma_\epsilon^2$$

Model Complexity: Bias Variance Decomposition...

$$Risk = \{E_D[h_D(x) - f(x)]\}^2 + E_D[\{h_D(x) - E_D[h_D(x)]\}^2] + \sigma_\epsilon^2$$

- where, $\{E_D[h_D(x) - f(x)]\}^2$ is the $(bias)^2$
- $E_D[\{h_D(x) - E_D[h_D(x)]\}^2]$ is the variance
- σ_ϵ^2 is the noise (Irreducible error)
- Let's see more details about these separate parts:
  $(bias)^2 : \{E_D[h_D(x) - f(x)]\}^2$
  - This term shows how accurate the hypothesis function (or your designed model, $h_D(x)$) is
  - As you can see, the E (expectation) means average out this error to find out the expectation of error regarding this hypothesis ($h_D(x)$) and the true function output ($f(x)$)
  - As long as you are building an accurate model with low error rate, $(bias)^2$ is a small value and possibly close to 0.

Model Complexity: Bias Variance Decomposition...

$$variance: E_D[\{h_D(x) - E_D[h_D(x)]\}^2]$$

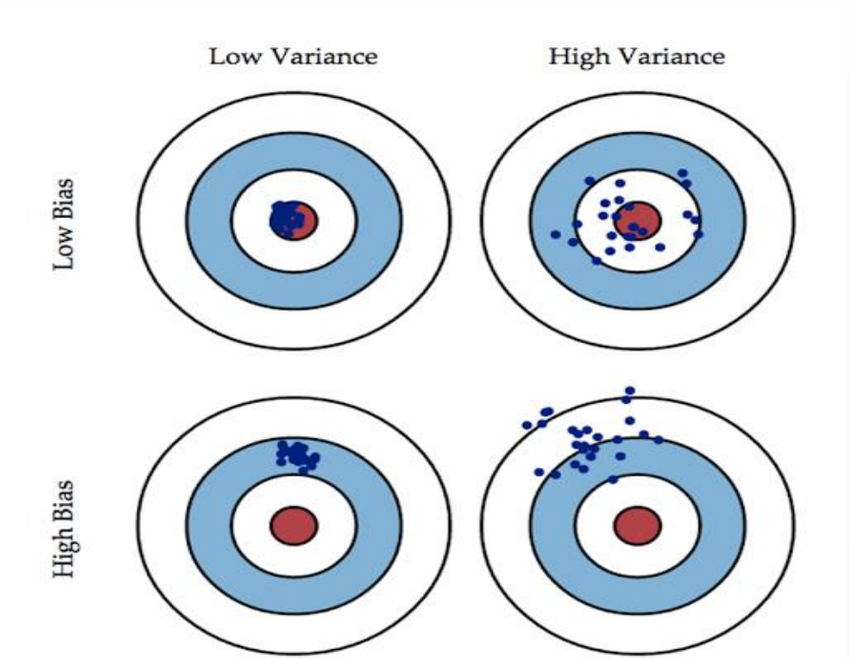
- As this term does not have $f(x)$ inside it, it solely relies on your model which is $h_D(x)$.
- To put it simply, this model measures the tolerance of your calculated model while changing just the data set D.
- If it varies too much that is a problem!
- The E or the expectation of this term measures the complexity of your model.
- The higher the variance the more complex the model.

Model Complexity: Bias Variance Decomposition...

- Hence, you can see that increasing the variance of a model means lower bias and the model becomes more complex.
- On the other hand you can see that the low complexity for a model will result in high bias and low variance.
- So higher bias results in lower variance and high variance results in lower bias.
- This illustrates another trade-off problem in machine learning.
- We can't control the other term, the noise σ_ϵ^2
  - Noise is just related to the observations from the function.

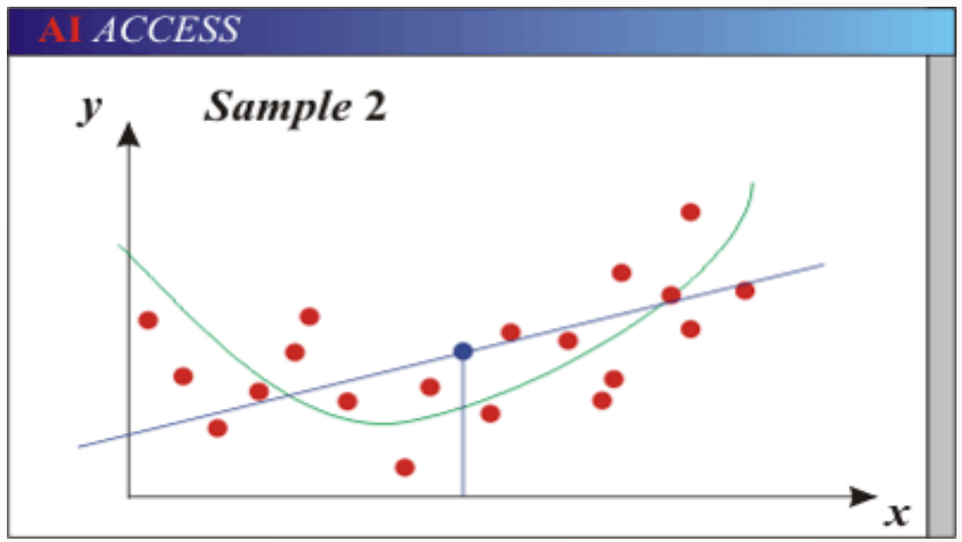
Model Complexity: Bias Variance Decomposition…

- To better illustrate the variance-bias trade-off, let's see the figure:
  - the best model is a model with low variance and low bias.
    - It means the model is not that complex but also it is properly accurate.
- The worst model could be the one with high bias which means not accurate based on the training data, and also high variance which means too far complex.

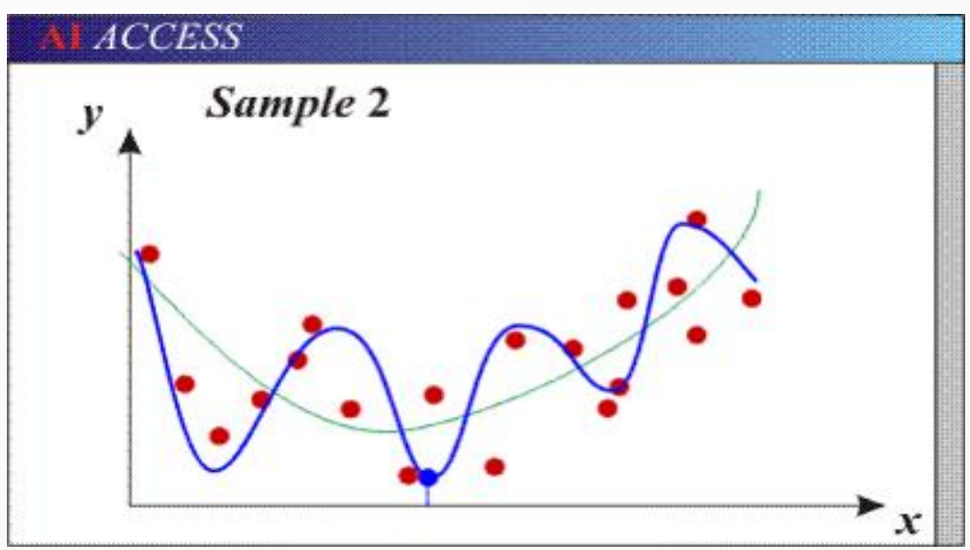


Model Complexity: Bias Variance Decomposition…

- Models with too few parameters are inaccurate because of a large bias (not enough flexibility): the case of under-fitting.



- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample): the case of over-fitting.



Model Complexity: Summary

- Based on the above-mentioned information about bias-variance trade-off, we know that:
  - Low bias implies high variance, and high bias implies low variance
- We need to find the sweet spot where Risk is minimum

$$\text{Risk} = (bias)^2 + \text{variance} + \text{noise}$$

  i.e., minimum error at the right model complexity.

- Can we overfit using linear models?!!! Depends!
  - model complexity.
- In linear models, the model complexity grows with the number of features.
  - Using all data dimensions as features may fit the model on not only true patterns (signal) but also on background noise.

Regularised Linear Model

Regularised linear models

- What is regulariser?
  - An additional term in the loss function to avoid overfitting.
  - Tries to keep the parameters more normal or regular i.e.,
    - it does not allow regression coefficients (or weights) to take excessively large value. What is wrong with this?
      - It implies the model is highly dependent on certain features

what if this feature is a noise or even highly affected by noisy observations.

Regularised linear models…

- Loss function has another term which is λ Regulariser(w). We can consider this term as complexity of the model.

$$\underset{w}{\text{minimize}} \frac{1}{n}\sum_{i} L(y_i, \boldsymbol{x}_i^T \boldsymbol{w}) + \lambda\, Regularizer(\boldsymbol{w})$$

- How do regularisers works in linear models?
  - Remember the linear model is $y = w_0 + \sum_{j=1}^{d} w_j x_j$
  - Should we allow all possible weights? Or impose any preferences? What makes a simpler linear model?

Regularised linear models: Our preference

- We do not want huge weights (i.e. do not want to over-rely on any one feature).
- If weights are huge, a small change in a feature would result in a large change in the prediction!
- In fact, since we may even have irrelevant features, we want some of the weights to be zero to discard some features.

$$\min_{w} \frac{1}{n} \sum_{i} L(y_i, \mathbf{x}_i^T \mathbf{w}) + \lambda \, Regularizer(\mathbf{w})$$

Regularised linear models: Our preference...

- But how do we penalize large weights or encourage small/zero weights ?
- There are two popular regulariser functions:
- Option 1:

$$Regularizer(\mathbf{w}) = \sum_j |w_j| = ||\mathbf{w}||_1, (l_1 - norm)$$

This encourages 0 weights (sparsity). This function implies the closed form function of a square.

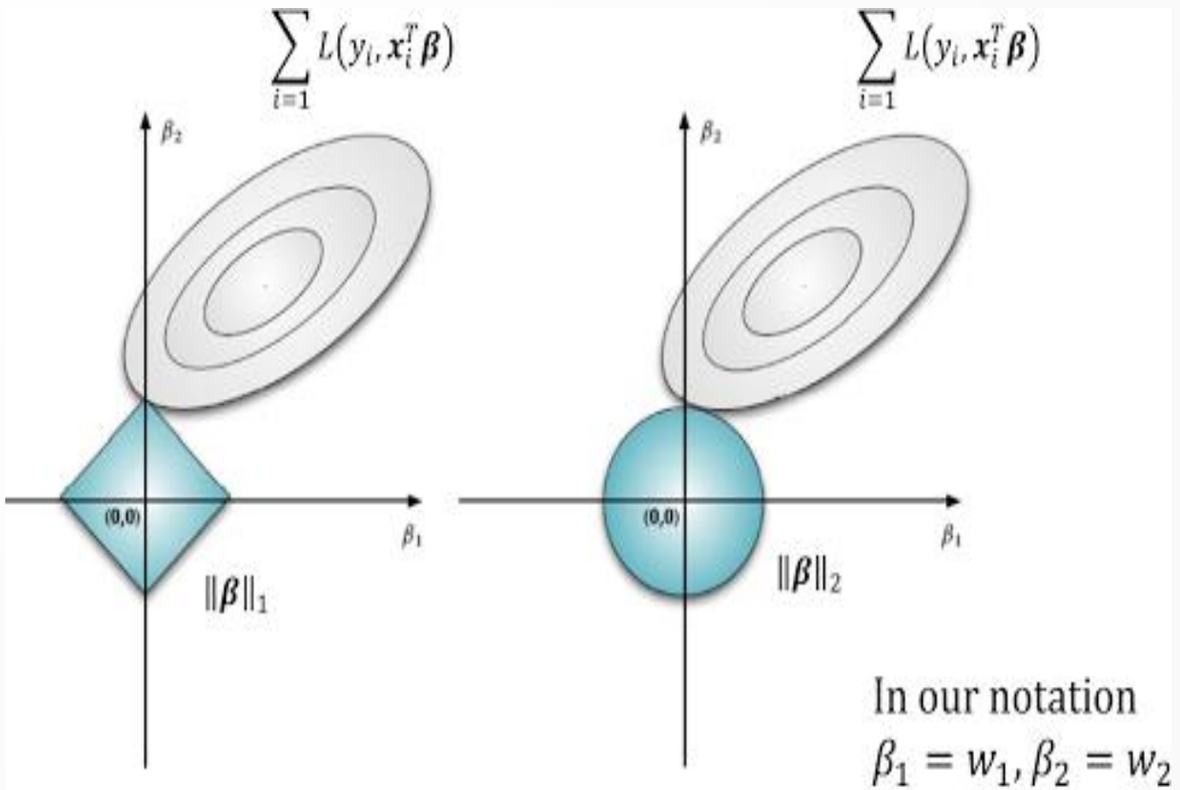
- Option 2:

$$Regularizer(\mathbf{w}) = \sum_j |w_j|^2 = ||\mathbf{w}||_2, (l_2 - norm)$$

This penalizes large weights. This function implies the closed form function of a circle.

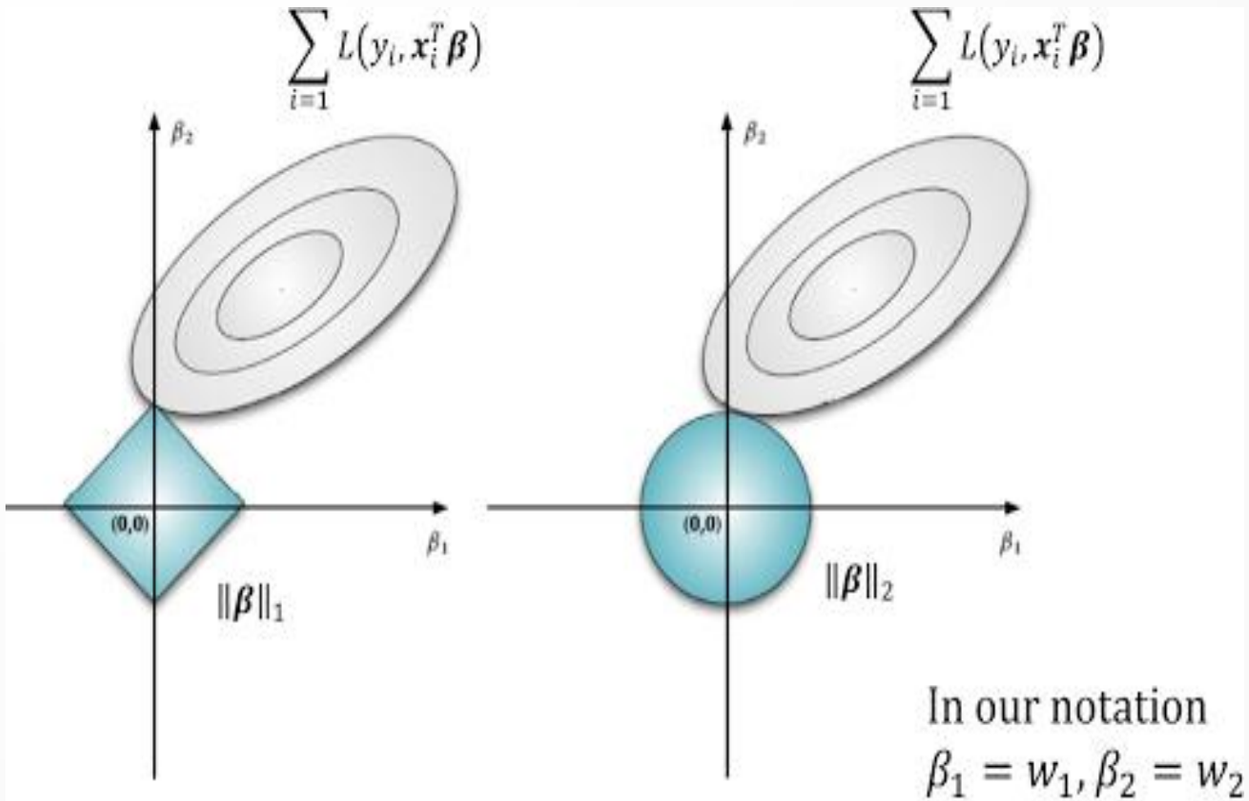
Regularised linear models: Our preference...

- The L1-norm forms a square shape, assume the loss function in form of ellipses in the plot.
- Since we are minimizing the loss function which actually has l1-norm regularization inside it, we need to find a sweet spot which is the intersection of these two regions.
- If you keep drawing the ellipses, you can find the intersection in the image.



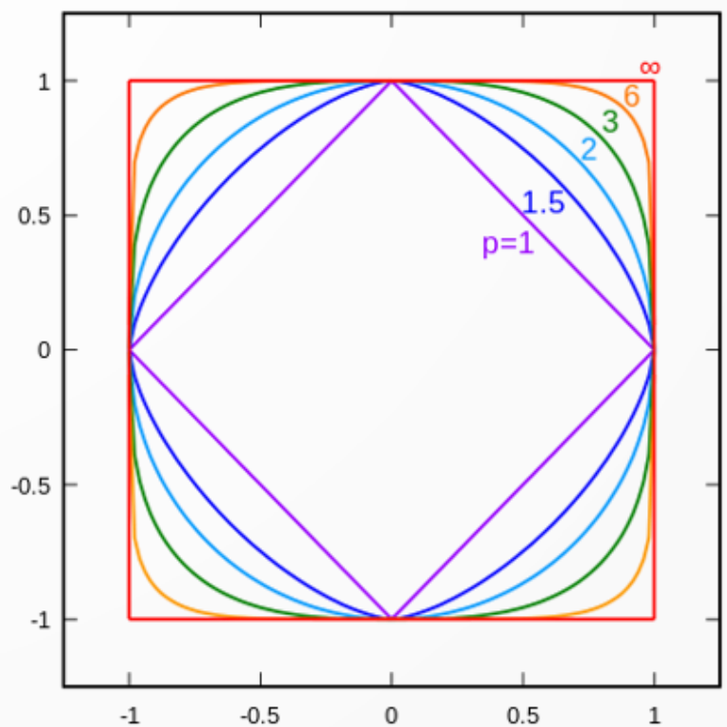
Regularised linear models: Our preference...

- L2-norm is the circle shape.
- There is more chance of intersection in this occasion.
- There is less chance of having 0 weights for β_1 or w_1 and β_2 or w_2.
- We have more options for selection of w_1 and w_2.



Regularised linear models: Our preference...

- The figure below shows the visualization of $l_p - norm$
- Also the ∞−norm regulariser is in form of square (superelipse).
- Remember that all of these L_p-norm regularisers are aiming of reducing the complexity of the model.
- All L_p-norms penalize larger weights.



Regularised linear models: L1 Regularisation (LASSO)

- Least Absolute Shrinkage and Selection Operator (LASSO)
  - performs
    - variable selection
    - regularization
  - enhances
    - prediction accuracy
    - interpretability of the model
- Formulation:

$$\min_{w} \frac{1}{n} \sum_{i} L(y_i, \mathbf{x}_i^T \mathbf{w}) + \lambda_1 ||\mathbf{w}||_1$$

Regularised linear models: L2 Regularization (Ridge) and Elastic net

- Ridge regularisation

$$\min_w \frac{1}{n}\sum_i L(y_i, \mathbf{x}_i^T\mathbf{w}) + \lambda_2||\mathbf{w}||_2^2$$

- Elastic net

$$\min_w \frac{1}{n}\sum_i L(y_i, \mathbf{x}_i^T\mathbf{w}) + \lambda_1||\mathbf{w}||_1 + \lambda_2||\mathbf{w}||_2^2$$

  - LASSO and ridge regularization are then special cases if Elastic net for λ_2=0 and λ_1=0.
  - Elastic net overcomes a limitation of LASSO for $d > n$ case:
    - LASSO selects at most n variables before it saturates.
    - Elastic net can select more number of variables despite the number of data points.

Regularised linear models - Summary

- For Linear regression we use squared loss functions,

$$L(y_i, \mathbf{x}_i^T \mathbf{w}) = (y_i - \mathbf{x}_i^T \mathbf{w})^2$$

- For Logistic regression we use logistic loss function,

$$L(y_i, \mathbf{x}_i^T \mathbf{w}) = log(1 + exp(-y_i \mathbf{x}_i^T \mathbf{w}))$$

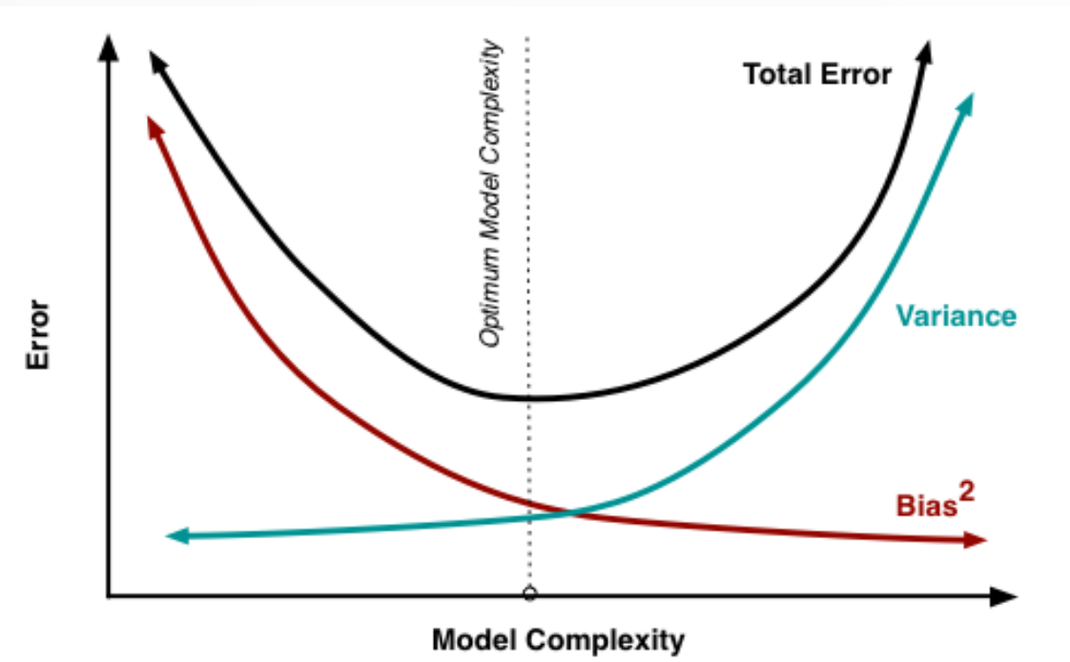
- So all in all, we solve the following optimization

$$\min_w \frac{1}{n} \sum_i L(y_i, \mathbf{x}_i^T \mathbf{w}) + \lambda_1 ||\mathbf{w}||_1 + \lambda_2 ||\mathbf{w}||_2^2$$

- For LASSO and Elastic net
  - Iterative optimization
    - Since L1 norm is non-differentiable and non-smooth, proximal gradient should be used to optimize this loss function.

Regularised linear models - Summary

- what are the effects of Regularization on Bias and Variance?
  - Increases bias
  - Decreases variance
- it is useful when the net effect (i.e. $bias^2 + variance$) reduces.

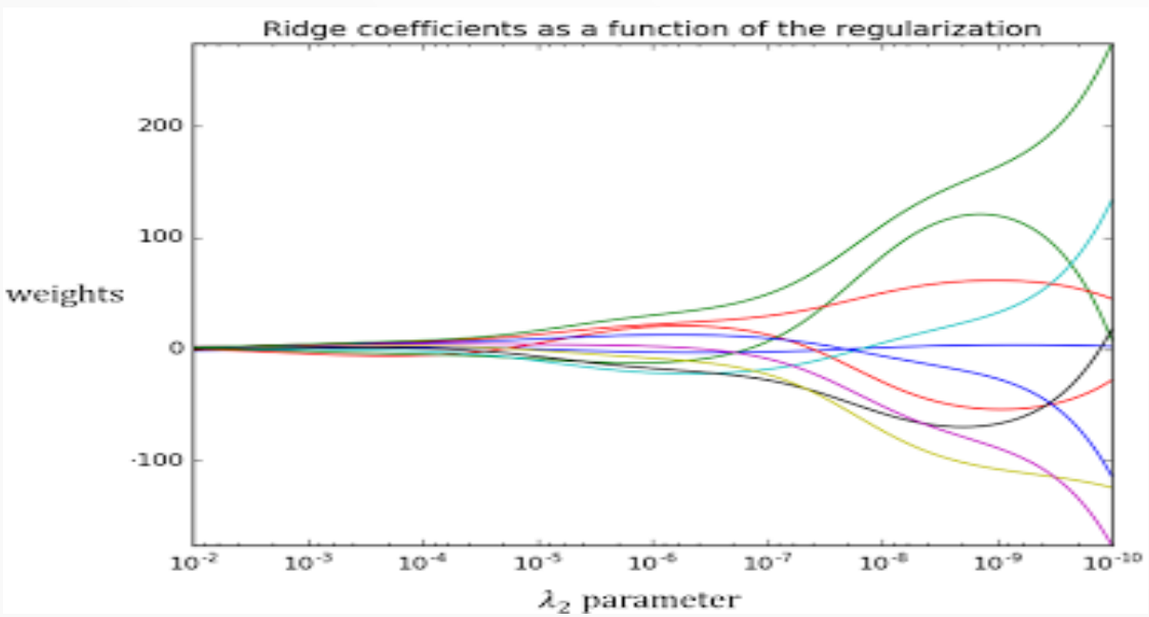


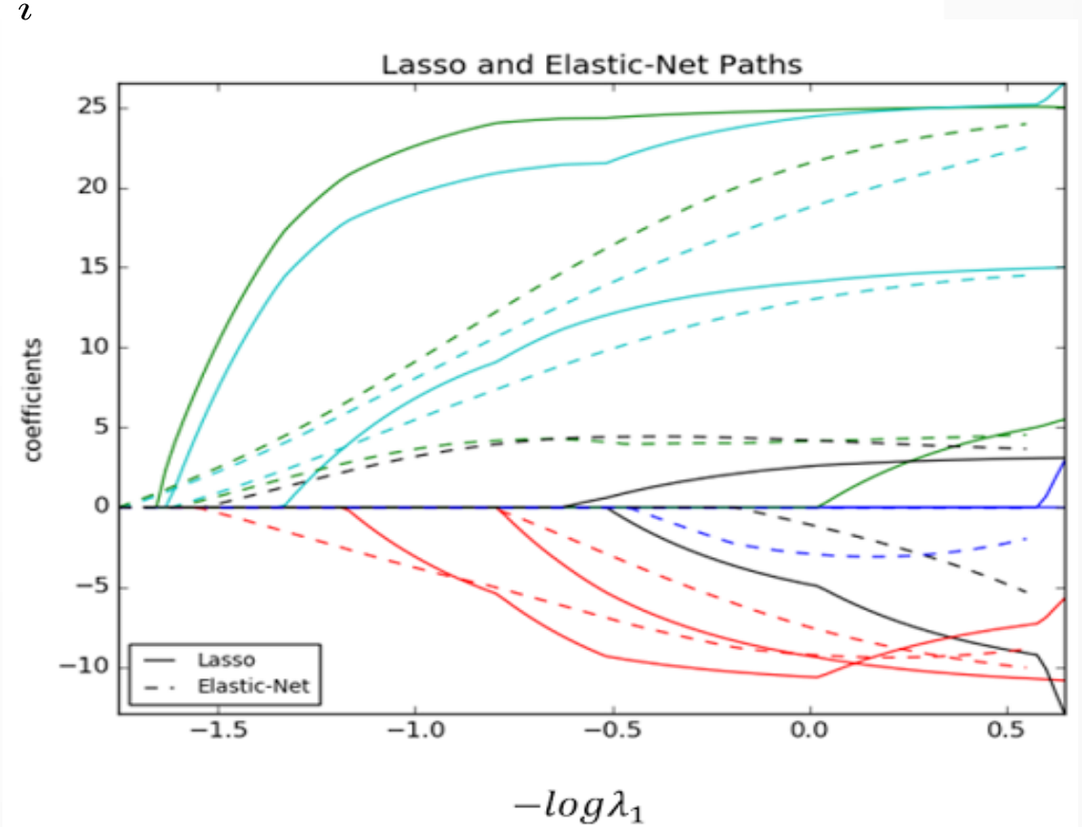
There is a trade-off or optimum model complexity.

Regularised linear models - Summary

$$\min_w \frac{1}{n} \sum_i L(y_i, \mathbf{x}_i^T \mathbf{w}) + \lambda_1 ||\mathbf{w}||_1 + \lambda_2 ||\mathbf{w}||_2^2$$

- Another important element is λ's.
- With larger values of λ_1 or λ_2
  - smaller values of weights.
- With larger values of λ_1 or λ_2
  - bigger values for weights.





Thank You.