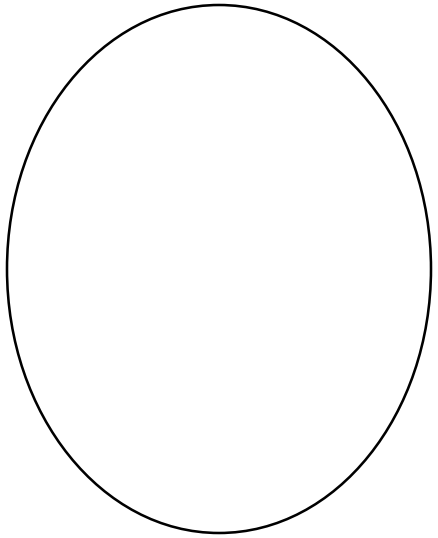
Deep Learning

Approximate



Approximate + Details

Extraction Information from Data

Consider the following data of tossing two fair coins A and B ten times in each experiment:

(B) H T T T H H T H T H (5H, 5T)
(A) H H H H T H H H H H (9, 1)
(A) H T H H H H H T H H (8, 2)
(B) H T H T T T H H T T (4, 6)
(A) T H H H T H H H T H (7, 3)

Q: find probability of getting head for coin A.

Sol.

Probability of getting head from coin A
= 24/(24 + 6) = 4/5 = 0.8

Probability of getting head from coin B
= 9/(9 + 11) = 9/20 = 0.45

Coin A	Coin B
	5 H, 5 T
9 H, 1 T	
8 H, 2 T	4 H, 6 T
7 H, 3 T	
24 H, 6 T	9 H, 11 T

Extraction Information from Data

Consider the following data of tossing two fair coins A and B ten times in each experiment:

i. H T T T H H T H T H (5H, 5T)
ii. H H H H T H H H H H (9, 1)
iii. H T H H H H H T H H (8, 2)
iv. H T H T T T H H T T (4, 6)
v. T H H H T H H H T H (7, 3)

Find probability of getting head for tossing coin A.

$$Y = x^3 - 5x^2 + 4x - 9$$

Extraction Information from Data

Sol.

Let $p_A(h)$=0.6 & $p_B(h)$=0.5 be probabilities of getting heads on A & B from the given data.

For 1st exp.(5, 5), *compute likelihood using the binomial distribution.*

P(k) = $^nC_k\, p^k(1 - p)^{n-k}$ *(here ^nC_k is 1 as only one possibility is considered.)*

Likelihood of A = $(p_A(h))^k\,(1 - p_A(h))^{n-k} = (0.6)^5(1- 0.6)^{10-5} = 0.000796$

Likelihood of B = $(p_B(h))^k\,(1 - p_B(h))^{n-k} = (0.5)^5(1- 0.5)^{10-5} = 0.000976$

Normalizing them gives

0.000796/0.001872=0.45, 0.000976/0.001872=0.55

So, ***$p_A(h) = 0.45,$ $p_B(h) = 0.55$***

For A, no. of heads = 0.45*5 = **2.25** and no. of tails = 0.55*5 = **2.75**

For B, no. of heads = 0.55*5 = **2.75** and no. of tails = 0.45*5 = **2.25**

Extraction Information from Data

For 2nd exp. (9H, 1T), *compute likelihood using binomial distribution*

Likelihood of A = $(0.6)^9(1-0.6)^1 = 0.004031$

Likelihood of B = $(0.5)^9(1-0.5)^1 = 0.000976$

Normalizing them gives 0.004031/0.0015007, 0.000976/0.0015007

$p_A(h) = 0.80,$ $p_B(h) = 0.20$

For A, no. of heads = 0.8*9 = **7.20** and no. of tails = 0.2*1 = **0.20**

For B, no. of heads = 0.2*9 = **1.80** and no. of tails = 0.8*1 = **0.80**

For 3rd exp. (8H, 2T)*, compute likelihood using binomial distribution*

Likelihood of A = $(0.6)^8(1-0.6)^2 = 0.002687$

Likelihood of B = $(0.5)^8(1-0.5)^2 = 0.000976$

Normalizing them gives 0.002687/0.003683, 0.000976/0.003683

$p_A(h) = 0.73,$ $p_B(h) = 0.27$

For A, no. of heads = 0.73*8 = **5.84** and no. of tails = 0.27*2 = **0.54**

For B, no. of heads = 0.27*8 = **2.16** and no. of tails = 0.73*2 = **1.46**

Extraction Information from Data

For 4th exp. (4H, 6T), *compute likelihood using binomial distribution*

Likelihood of A $= (0.6)^4(1-0.6)^6 = 0.000536$

Likelihood of B $= (0.5)^4(1-0.5)^6 = 0.000976$

Normalizing them gives 0.00536/0.001512, 0.000976/0.001512

$p_A(h) = 0.35, \quad p_B(h) = 0.65$

For A, no. of heads = 0.35*4 = **1.40** and no. of tails = 0.65*6 = **3.90**

For B, no. of heads = 0.65*4 = **2.60** and no. of tails = 0.35*6 = **2.10**

For 5th exp. (7H, 3T), *compute likelihood using binomial distribution*

Likelihood of A $= (0.6)^7(1-0.6)^3 = 0.001791$

Likelihood of B $= (0.5)^7(1-0.5)^3 = 0.000976$

Normalizing them gives 0.001791/0.002767, 0.000976/0.002767

$p_A(h) = 0.65, \quad p_B(h) = 0.35$

For A, no. of heads = 0.65*7 = **4.55** and no. of tails = 0.35*3 = **1.05**

For B, no. of heads = 0.35*7 = **2.45** and no. of tails = 0.65*3 = **1.95**

Extraction Information from Data

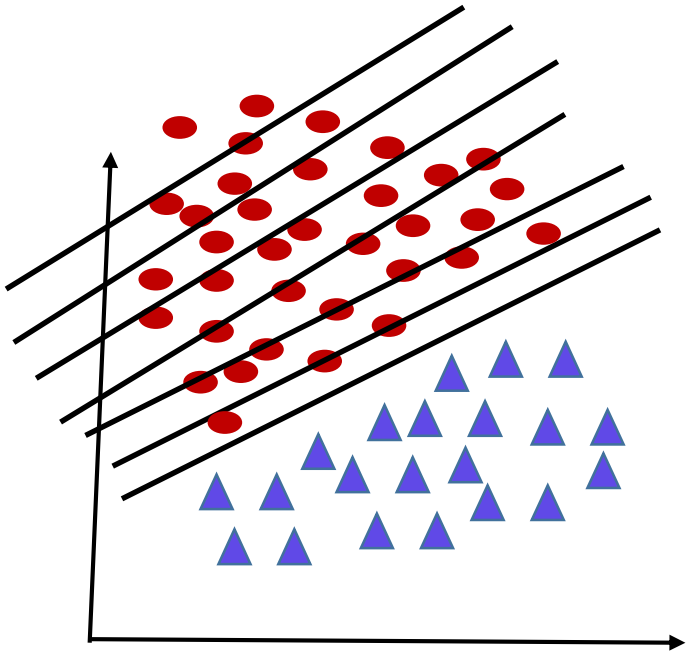
Exp. No.	Coin A		Coin B	
	Heads	Tails	Heads	Tails
1	2.25	2.75	2.75	2.25
2	7.20	0.20	1.80	0.80
3	5.84	0.54	2.16	1.46
4	1.40	3.90	2.60	2.10
5	4.55	1.05	2.45	1.95
Total	21.24	8.44	11.76	8.56

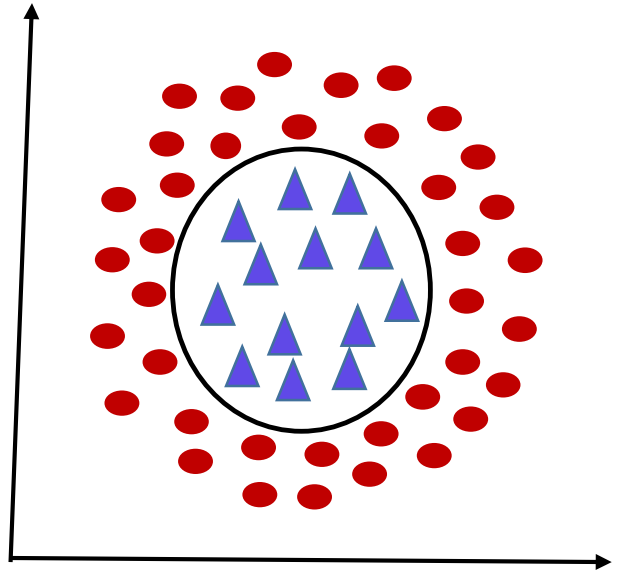
$p_A(h)$ = 21.24/29.68 = **0.7156**; $p_B(h)$ = 11.76/20.32 = **0.57874**

These are the probabilities of getting heads after one iteration.

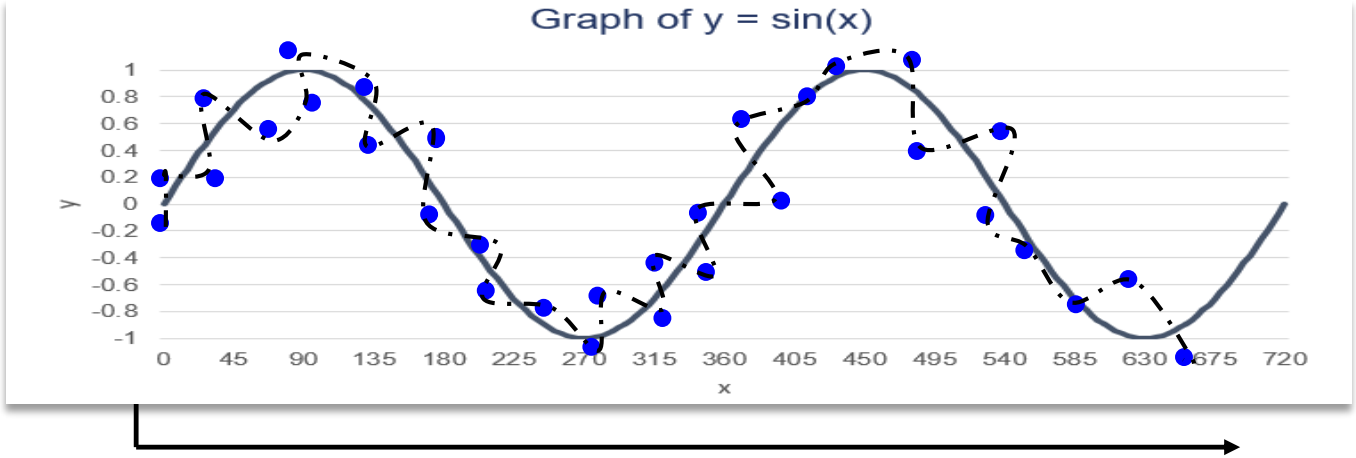
- *Perform same process with these probabilities for all experiments and compute new probabilities.*
- Using the newly computed probabilities, repeat the process till the probabilities are stabilized. They will be the probabilities of getting heads.

How to Classify Data



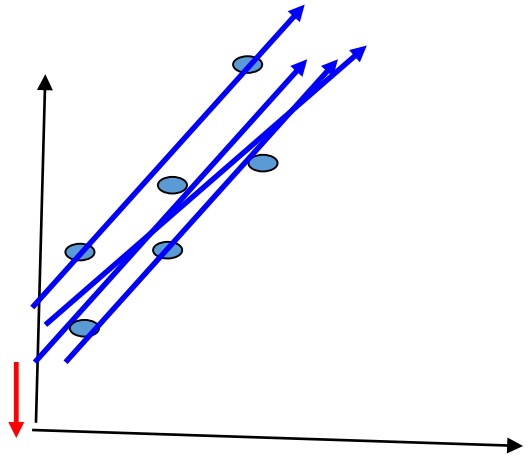


Curve fitting



$$Y = f(x) = a_0 + a_1 x$$

$$E = (y_{actual} - a_0 - a_1 x)^2$$



$a_0^2 <= r \quad g(x) = E + \alpha_0 a_0^2 + \alpha_1 a_1^2$

$a_1^2 <= r$

Machine Learning

- Machine learning is the study of powerful techniques that can learn from experience.
- As a machine learning algorithm accumulates more experience, typically in the form of observational data or interactions with an environment, its performance improves.
- Some core components, no matter what kind of machine learning problem we take on, are:

1. *The data that we can learn from.*
2. *A model of how to transform the data.*
3. *An objective function that quantifies how well (or badly) the model is doing.*
4. *An algorithm to adjust the model's parameters to optimize the objective function.*

Some characteristics of Matrix

- A **matrix A** of size **nxn** is ***diagonalizable*** if it has **n** distinct eigenvalues.
- We can write **A = PDQ,** where **P, Q** are invertible, **$Q = P^{-1}$** and **D** is diagonal matrix that contains eigenvalues of **A**.

$$A = PDP^{-1}$$

- If matrix **A** is singular (**det(A) = 0**), **can we find P, Q, D matrices ???**

Some Characteristics of Matrix

- Consider **A** $= \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix}$.
- We can write **A** $= \frac{1}{\sqrt{5}} \begin{bmatrix} 2 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} \sqrt{10} & 0 \\ 0 & 0 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.
- Here **P** and **Q** are orthogonal matrices and their columns are called **left** and **right singular** vectors.
- The elements of **D** are called **singular values** of **A** and they are **ordered** in decreasing order.

$A_{mxn}X_{nx1} = B_{mx1}$, m>>n, $\sum y = m\sum x + c\sum 1$; $\sum xy = m\sum x^2 + c\sum x$

- $A_{mxn}X_{nx1} = B_{mx1}$, m<n, $(A^T_{nxm}A_{mxn})X_{nx1} = A^T_{nxm}B_{mx1}$
- $X_{nx1} = (A^T_{nxm}A_{mxn})^{-1}(A^T_{nxm}B_{mx1})$
- $X = A^{-1}B$, m = n

Machine Learning

- The change from (comparatively) small to big data is a major contributor to success of modern deep learning
- It is not enough to have lots of data and to process it cleverly. We need the *right* data.
- If the data are full of mistakes, or if the chosen features are not predictive of the target quantity of interest, then the learning is going to fail.
- One common failure occurs in datasets where some groups of people are unrepresented in the training data.
- Most machine learning algorithms involve transforming the data in some sense.
- Deep learning is differentiated from the classical methods mainly by the set of powerful models that it focuses on.

Machine Learning

- Machine learning is generally considered as learning from experience. By learning here, we mean improving at some task over time.
- In order to develop a formal mathematical system of learning machines, we need to have formal measures of how good (or bad) our models are.
- In machine learning, and optimization more commonly, we call these *objective functions*.
- By convention, we usually define the objective functions so that lower is better. This is merely a convention.
- Because lower is better, these functions are sometimes called *loss functions*.
- When trying to predict numerical values, the most common loss function is squared error, i.e., square of the difference between the prediction and ground-truth.

Machine Learning

- For classification, the most common objective function is to minimize the error rate, i.e., the fraction of examples on which our predictions disagree with the ground truth.
- Some objective functions (e.g., squared error) are easy to optimize; others (e.g., error rate) are difficult to optimize directly, due to non-differentiability or other complications.
- In cases of no-differentiability, it is common to optimize a surrogate function, which is a function that approximates another function and it takes very less time to evaluate as compared to the main objective function.
- Typically, the loss function is defined with respect to the model's parameters and it depends upon the dataset.

Machine Learning

- We learn the best values of our model's parameters by minimizing the loss incurred on a set consisting of some number of examples collected for training.
- However, doing well on training data does not guarantee that we will do well on the unseen data.
- So we typically want to split the available data into two partitions: training dataset (or training set, for fitting model parameters) and test dataset (or test set, which is held out for evaluation), reporting how the model performs on both of them.
- It could be thought of training performance as being like a student's scores on practice exams used to prepare for some real final exam.

Machine Learning

- Even if the intermediate results (e.g., minor tests) are encouraging, that does not guarantee the success on the final exam, i.e., the test performance can deviate significantly from the training performance.
- When a model performs well on training set, but fails to generalize to unseen data, we say that it is **overfitting**.
- In real-life terms, this is like flunking the real exam. despite doing well on the practice exams.
- Once we have some data source and representation, a model, and a well-defined objective function, we need an algorithm capable of searching for the best possible parameters for minimizing the loss function.
- Popular optimization algorithms for deep learning are based on an approach, called gradient descent, but no the gradient descent method itself.

Machine Learning

- The gradient descent, at each step, checks to see, for each parameter, which way the training set loss would move by perturbing that parameter just a small amount.
- It then updates the parameter in the direction that may reduce the loss.
- The basic approach is

$$w_{new} = w_{old} + \rho(\hat{y} - y_{actual})x$$

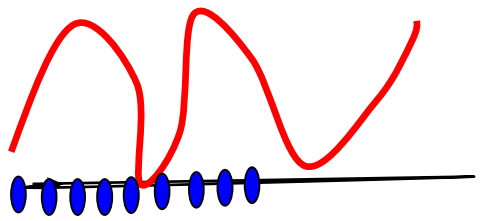
w is weight or parameter of model,

ρ is learning rate

$\hat{y}$ is estimated/computed value of target variable

y_{actual} is actual value (ground truth)

x is value of input variable



Deep Learning

- Machine learning can use data to learn transformations between inputs and outputs, such as transforming audio into text in speech recognition.
- In doing so, it is often necessary to represent data in a way that is suitable for algorithms to transform such representations into the output.
- Though deep learning is a subset of machine learning, the dizzying set of algorithms and applications makes it difficult to assess what specifically the ingredients for deep learning might be.
- This is as difficult as trying to pin down required ingredients for pizza since almost every component is substitutable.

Deep learning

- Deep learning is deep in precisely the sense that its models learn many layers of transformations, where each layer offers the representation at one level.
- For example, layers near the input may represent low-level details of the data, while the layers closer to the classification output may represent more abstract concepts used for discrimination.
- Since representation learning aims at finding the representation itself, deep learning can be referred to as multi-level representation learning.
- The problems such as learning from raw audio signal, raw pixel values of images, or mapping between sentences of arbitrary lengths and their counterparts in foreign languages, are those where deep learning excels and the traditional machine learning methods falter.

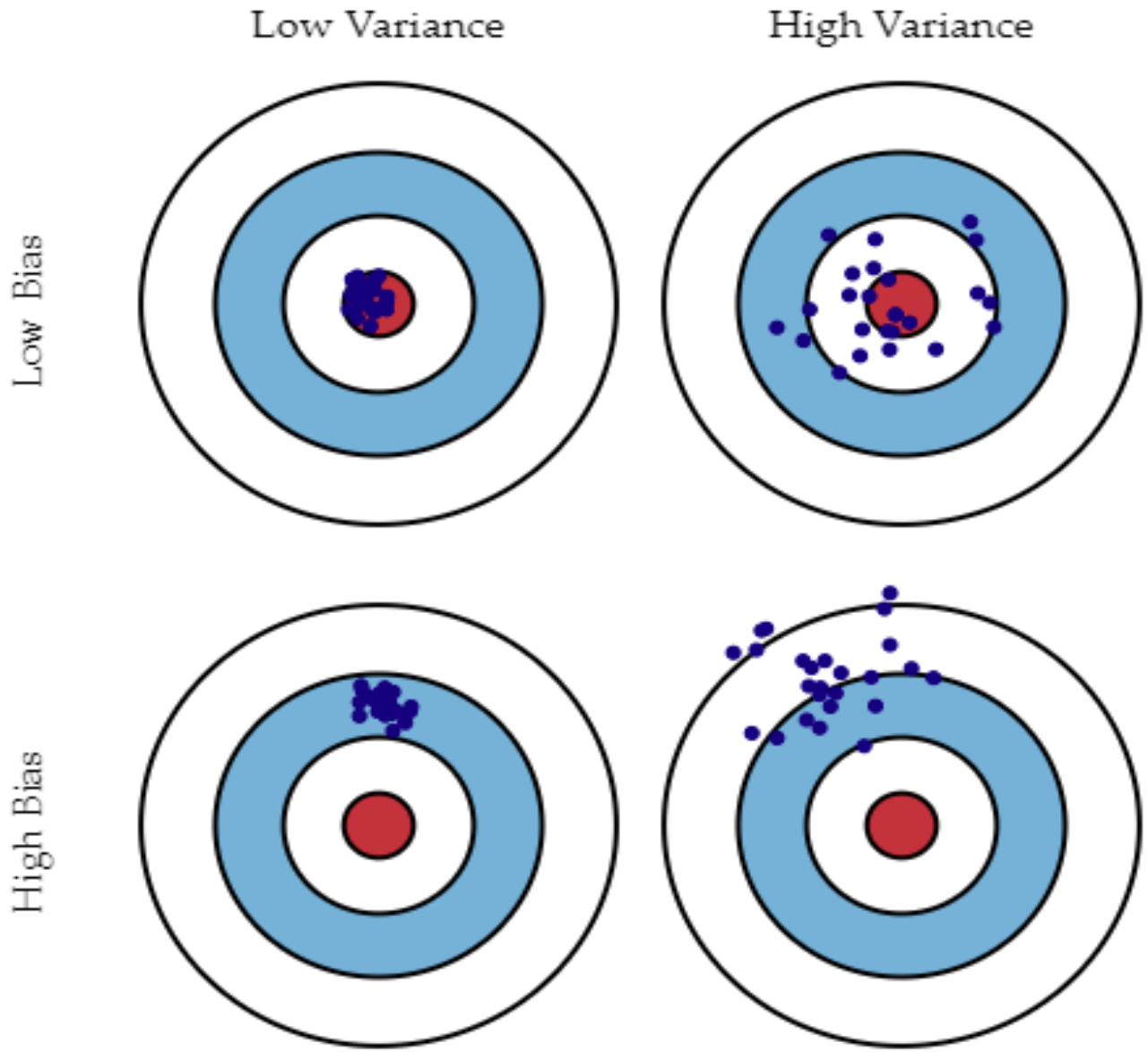
Regularization

- Sometimes a machine learning model performs well on training data, but it doesn't perform well on unseen data.
- It means that the model is not able to predict the output or target for the unseen, and it is called an **overfitted model**.
- By noise we mean those data points in the dataset which don’t really represent true properties of the data, but only due to a random chance (noise).
- So, to deal with the problem of overfitting, we take the help of a regularization technique.
- A regularization technique regularizes/prevents the model from overfitting by adding extra information to it.
- It is a form of regression that shrinks the coefficient estimates towards zero, i.e., it forces the model not to learn a more complex or flexible model, to avoid the problem of overfitting.

Regularization

- The word **regularize** means to make things regular or acceptable. This is exactly why we use it for.
- Regularization refers to the techniques used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting.
- It helps to improve the model to work on the unseen data by ignoring the less important features.
- It minimizes the validation loss and tries to improve the accuracy of the model.
- It avoids overfitting by adding a penalty to the model with high variance, thereby shrinking the coefficients towards to zero.

Bias and variance



Bias and variance

- **Low Bias:** The average prediction is very close to the target value (actual or ground truth)
- **High Bias:** The predictions differ too much from the actual value (ground truth)
- **Low Variance:** The data points are compact and do not vary much from their mean value
- **High Variance:** Scattered data points with huge variations from the mean value and other data points.
- To make a good fit, we need to have a balance of bias and variance.

Regularization

- We try to understand "How flexibility of a model is represented?"
- For regression problems, the increase in flexibility of a model is represented by an increase in its coefficients, which are calculated from the regression line.
- In a regularization technique, we reduce the magnitude of the independent variables by keeping the same number of variables or making less relevant independent variables as zero.
- It maintains accuracy as well as a generalization of the model. Generalization means that the model performs better on the unseen data as well.
- Regularization works by adding a penalty or complexity term or shrinkage term with Residual Sum of Squares (RSS)/mean square error (MSE) to the complex model.

Regularization

- Let Y represent the dependent feature or response which is the learned relation. For simplicity, we consider simple linear regression. Then Y can be approximated as follows:

$$Y = w_0 + w_1X_1 + w_2X_2 + \ldots + w_pX_p$$

here X_1, X_2, …X_p are independent variables/features/predictors for Y, and w_0, w_1,…..w_n denote coefficients estimates for different variables or predictors (X), which signify the weights or magnitude attached to the features, respectively; w_0 is bias.

- In simple linear regression, the optimization function or loss function is the residual sum of squares (RSS), as given below.

$$RSS = \sum_{i=1}^{n} \left(y_i - w_0 - \sum_{j=1}^{p} w_j x_{ij} \right)^2$$

- We choose those set of coefficients w_1,…..w_n, such that the above loss function is minimized.

Regularization

- This will adjust the coefficient estimates based on the training data.
- If there is noise present in the training data, then the estimated coefficients won't generalize well and are not able to predict the future/unseen data.
- This is where regularization comes into picture, which shrinks or regularizes these learned estimates towards zero, by adding a loss function with optimizing parameters to make a model that can predict the accurate value of Y.
- Regularization works by adding a penalty or complexity term or shrinkage term with Residual Sum of Squares (RSS) to the complex model.
- There are two types of regularization techniques:

  - Ridge Regression

  - Lasso (Least Absolute Shrinkage and Selection Operator) **Regression**

Regularization

Ridge Regression

- It is one of the types of linear regression in which we introduce a small amount of bias, known as **Ridge regression penalty**, so that we can get better long-term predictions. In Statistics, it is known as the L_2 norm.
- In this technique, the cost function is altered by adding the penalty term (shrinkage term), which multiplies λ with squared weight of each individual feature.
- The optimization function (cost/loss function) becomes:

$$L(X, y, w) = \sum_{i=1}^{n} \left(y_i - w_0 - \sum_{j=1}^{p} w_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} w_j^2$$

here, λ is tuning parameter that decides how much we want to penalize the flexibility of our model.

- As λ increases, cost function increases, the coefficient of the equation decreases and leads to shrinkage.

Regularization

Ridge Regression

- In ridge regression, the RSS is modified by adding the shrinkage quantity.
- The coefficients are estimated by minimizing this function.
- The increase in flexibility of a model is represented by increase in its coefficients, and if we want to minimize the above function, then these coefficients need be small.
- This is how Ridge regression technique prevents coefficients from rising too high.
- Also, we shrink the estimated association of each variable with the response, except intercept w_0, which is a measure of the mean value of the response when $x_{i_1} = x_{i_2} = x_{i_3} = \ldots = x_{i_p} = 0$.

Regularization

Ridge Regression

- When $\lambda = 0$, the penalty term has no effect, and the estimates produced by ridge regression will be equal to the least squares.
- However, as $\lambda \rightarrow \infty$, the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach zero.
- Selecting a good value of λ is critical. Cross validation comes in handy for this purpose.
- The coefficients produced by the standard least squares method are scale equivariant, i.e., if we multiply each input by c, then the corresponding coefficients are scaled by a factor of 1/c.

Regularization

Ridge Regression

- So, regardless of how the predictor is scaled, the multiplication of predictor and coefficient ($x_j w_j$) remains the same.
- However, this is not the case with ridge regression, and so, we need to standardize the predictors or bring the predictors to the same scale before performing ridge regression.
- The formula used to do this is given below.

$$\bar{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(x_{ij} - \bar{x}_i\right)^2}}$$

Regularization

The loss function is: $L(X, y, w) = SSE(X, y, w) + \frac{1}{2}\lambda w w^T$

Its gradient is: $\nabla_w L(X, y, w) = \nabla_w SSE(X, y, w) + \lambda w$

For a single training step and a learning rate θ, this can be written as

$$\mathrm{w} = (1 - \lambda\theta)w - \theta\nabla_w SSE(X, y, w)$$

The equation effectively shows us that each weight of the weight vector will be reduced by a constant factor on each training step.

Looking at it from the entire training viewpoint, here is what happens:

The L2 regularizer will have a big impact on the directions of the weight vector that don't "contribute" much to the loss function.

It will have a relatively small effect on the directions that contribute to the loss function.

As a result, we reduce the variance of our model, which makes it easier to generalize on unseen data.

Regularization

- We further simplify the analysis by making a quadratic approximation to objective function in neighborhood of the value of weights that obtains minimal unregularized training cost, $w^* = \arg min_w L(X, y, w)$.
- If objective function is truly quadratic, as in the case of fitting a linear regression model with mean squared error, then the approximation is perfect.
- The approximation $L(X, y, w)$ is given by

$$L(X, y, \theta) = SSE(X, y, w^*) + \frac{1}{2}(w - w^*)^T H(w - w^*)$$

It is expansion of SSE w.r.t w^* using Taylor series up to two terms (1st derivative is zero as w^*gives minimum value)

θ is a vector consisting all weights including bias.

w^* is vector when SSE has minimum value;

H is Hessian matrix of L w.r.t. w evaluated at w$*$. It is Hermitian matrix, which is positive definite.

- The minimum of $L(X, y, \theta)$occurs where its gradient is 0:

$$\nabla_w L(X, y, \theta) = H(w - w^*)$$

Regularization

- To study the effect of weight decay, we modify $\nabla_w L(X, y, \theta) = H(w - w^*)$ by adding the weight decay gradient.
- We can now solve for the minimum of the regularized version of $L(X, y, \theta)$.
- We use variable $\hat{w}$ to denote the location of minimum.

$$\lambda\hat{w} + H(\hat{w} - w^*) = 0$$
$$\hat{w}(\lambda I + H) - Hw^* = 0$$
$$\hat{w} = (\lambda I + H)^{-1} Hw^*$$

- As $\lambda \rightarrow 0$, the regularized solution $\hat{w} \rightarrow w*$. But what happens as α grows?
- Since H is real and symmetric, it can be decomposed into a diagonal matrix S and an orthonormal basis of eigenvectors, Q, such that $H = Q\Lambda Q^T$.

$$\hat{w} = (\lambda I + QSQ^T)^{-1} QSQ^T w^*$$
$$\hat{w} = Q(\lambda I + S)^{-1} SQ^T w^*$$

Regularization

- We see that the effect of weight decay is to rescale $w*$ along the axes defined by eigenvectors of H.
- Specifically, the component of $w*$ that is aligned with i^{th} eigenvector of H is rescaled by a factor of $\frac{\alpha_i}{\alpha_i+\lambda}$, α_i is ith Eigen value of H.
- Along the directions where eigenvalues of H are relatively large, for example, where $\alpha_i >> \lambda$, the effect of regularization is relatively small.
- However, the components $\alpha_i \ll \lambda$ will be shrunk to have nearly zero magnitude.

Regularization

Least Absolute Shrinkage and Selection Operator(LASSO)

- Lasso is another variation, in which the following function is minimized.

$$L(X, y, \theta) = \sum_{i=1}^{n} \left(y_i - w_0 - \sum_{j=1}^{p} w_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |w_j|$$

- It adds L_1 penalty, which is the sum of absolute value of w_j coefficients.
- The ridge regression can be thought of as solving an equation, where summation of squares of coefficients is less than or equal to some constant s that exists for each value of shrinkage factor λ.
- Lasso can be thought of as an equation where summation of modulus of coefficients is less than or equal to s and these equations are also referred to as constraint functions.

Regularization

- The loss function is: $L(X, y, \theta) = SSE(X, y, w) + \lambda|w|$
- Its gradient is: $\nabla_w L(X, y, \theta) = \nabla_w SSE(X, y, w) + \lambda \frac{w}{|w|}$
- The value of $\frac{w}{|w|}$ is simply the sign of w = $sign(w)$.
- The regularization term does not scale linearly, contrary to L2 regularization, but it's a constant factor with an alternating sign.
- Our simple linear model has a quadratic cost function that we can represent via its Taylor series.
- The gradient in this setting is given by

$$\nabla_w L(X, y, \theta) = H(w - w^*)$$

- As L1 penalty does not admit clean algebraic expressions in case of a fully general Hessian, we will also make further simplifying assumption that Hessian is diagonal, H = diag([$H_{1,1}$, . . . ,$H_{n,n}$]), where $H_{i,i} > 0$.

Regularization

- This assumption holds if the data for linear regression problem has been preprocessed to remove all correlation between the input features, which may be accomplished using PCA.
- Quadratic approximation of L1 regularized objective function decomposes into a sum over the parameters:

$$L(X, y, w) = SSE(X, y, w*) + \frac{1}{2}\sum_{i}\left[H_{i,i}(w_i - w_i^*)^2 + \lambda|w_i|\right]$$

- Its 1st derivative is $H_{i,i}\,(w_i - w_i^*) + \lambda sign(w_i) = 0$. It is to note that w_i cannot be negative.
- The problem of minimizing this approximate cost function has an analytical solution (for each dim i), with following form:

$$w_i = sign(w_i^*)max\left\{|w_i^*| - \frac{\lambda}{H_{i,i}}, 0\right\}$$

Regularization

Consider the situation where w_i^*>0 for all i. There are two possible outcomes:

i. $\frac{\lambda}{H_{i,i}} \leq w_i^*$: Here the optimal value of w_i under regularized objective is simply $w_i = 0$.

This occurs because the contribution of penalty term to regularized objective L is overwhelmed - in direction i – by L1 regularization which pushes w_i value to zero.

ii. $\frac{\lambda}{H_{i,i}} > w_i^*$: The regularization does not move optimal value of w_i to zero but it just shifts it in that direction by a distance equal to $\frac{\lambda}{H_{i,i}}$

Similar process happens when w_i^*<0, but with L1 penalty making w_i less negative by $\frac{\lambda}{H_{i,i}}$

Regularization

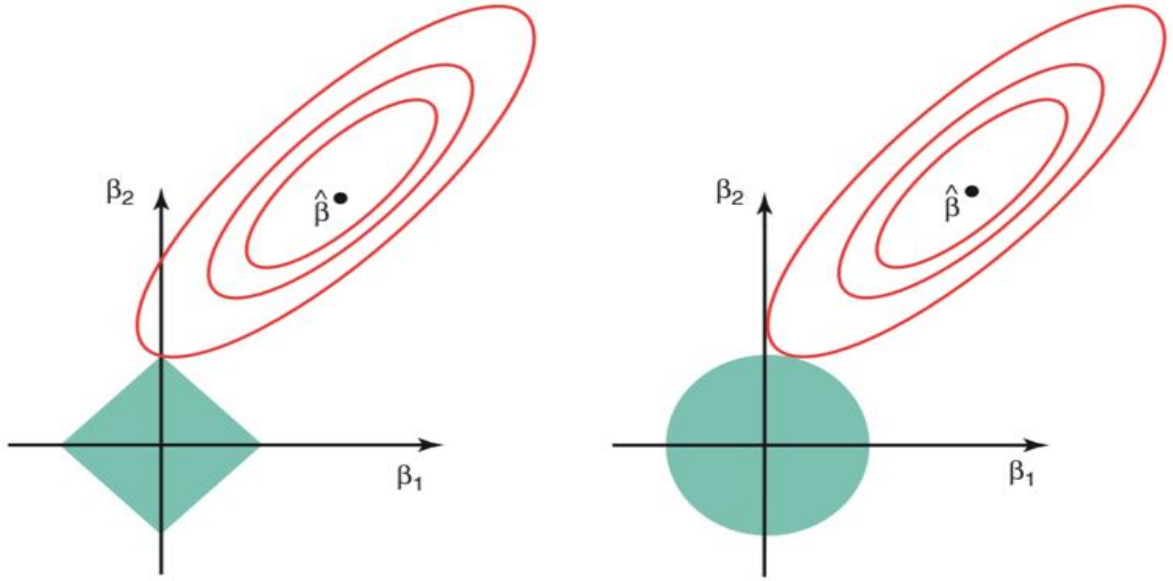
It affects the overall training as:

The L1 regularizer introduces sparsity in the weights by forcing more weights to zero instead of reducing the average magnitude of all weights (as L2 regularizer does).

In other words, L1 suggests that some features should be discarded whatsoever from the training process.

Regularization

- Assuming 2 parameters in a given problem, the ridge regression is expressed by $w_1^2 + w_2^2 \leq s$.
- This implies that ridge regression coefficients have the smallest RSS (loss function) for all points that lie within the circle given by $w_1^2 + w_2^2 \leq s$.
- For Lasso, it becomes $|w_1|+|w_2| \leq s$, it means that lasso coefficients have the smallest RSS (loss function) for all points that lie within the diamond given by $|w_1|+|w_2| \leq s$.



Taken from: ***An Introduction to Statistical Learning by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani***

Regularization

- The above image shows the constraint functions (green areas), for lasso (left) and ridge regression(right), along with contours for RSS (red ellipse).
- Points on the ellipse share the value of RSS.
- The lasso and ridge regression coefficient estimates are given by the first point at which an ellipse contacts the constraint region.
- Since ridge regression has a circular constraint with no sharp points, this intersection will not ***generally*** occur on an axis, and so the ridge regression coefficient estimates will be exclusively non-zero.
- The lasso constraint has corners at each of the axes, and so the ellipse will ***often*** intersect the constraint region at an axis. When it occurs, one of the coefficients will equal zero.
- In higher dim (where parameters are much > 2), many of the coefficient estimates may equal zero simultaneously.

Regularization

- The Ridge will shrink the coefficients for least important predictors, very close to zero. But it will never make them exactly zero.
- Thus, the final model will include all predictors.
- The Lasso has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.
- So, lasso also performs variable selection and is said to yield sparse models.
- A standard least squares model tends to have some variance in it, i.e. this model won't generalize well for a data set different than its training data.

Regularization

- Regularization significantly reduces the variance of the model, without substantial increase in its bias.
- So tuning parameter λ used in regularization techniques, controls the impact on bias and variance.
- As the value of λ increases, it reduces the value of coefficients and thus reducing the variance.
- Till a point, this increase in λ is beneficial as it is only reducing the variance (hence avoiding overfitting), without loosing any important properties in the data.
- But after certain value, the model starts loosing important properties, giving rise to bias in the model and thus underfitting.
- So, the value of λ should be carefully selected.

Regularization

- We now study the trajectory followed by the parameter vector during training.
- For simplicity, set initial parameter vector to origin, $w^{(0)}=0$
- We study the approximate behavior of gradient descent on SSE by analyzing gradient descent on L:

$$w^{(\tau)} = w^{(\tau-1)} - \epsilon \nabla_w L(X, y, w^{(\tau-1)})$$

$$w^{(\tau)} = w^{(\tau-1)} - \epsilon H(w^{(\tau-1)} - w^*)$$

$$w^{(\tau)} - w^* = (I - \epsilon H)(w^{(\tau-1)} - w^*)$$

$$w^{(\tau)} - w^* = (I - \epsilon QSQ^T)(w^{(\tau-1)} - w^*)$$

$$Q^T(w^{(\tau)} - w^*) = (I - \epsilon S)QT(w^{(\tau-1)} - w^*)$$

Learning rate ϵ is chosen to be small enough to guarantee $|1 - \epsilon\alpha_i| < 1$. The parameter trajectory during training after τ parameter updates is as follows:

$$Q^T w^{(\tau)} = [I - (I - \epsilon S)^\tau] Q^T w^*$$

Regularization

$$Q^T w^{(\tau)} - (I - \epsilon S) Q^T w^{(\tau-1)} = Q^T w^* - (I - \epsilon S) QT w^*$$

Putting τ=1,2,…,

$Q^T w^{(1)} - (I - \epsilon S) Q^T w^{(0)} = Q^T w^* - (I - \epsilon S) QT w^*$ (r1)

$Q^T w^{(2)} - (I - \epsilon S) Q^T w^{(1)} = Q^T w^* - (I - \epsilon S) QT w^*$ (r2)

$Q^T w^{(3)} - (I - \epsilon S) Q^T w^{(2)} = Q^T w^* - (I - \epsilon S) QT w^*$ (r3)

…. … … … … …

$Q^T w^{(k)} - (I - \epsilon S) Q^T w^{(k-1)} = Q^T w^* - (I - \epsilon S) QT w^*$ (rk)

$w^{(0)} = 0$, multiplying eqn. (r1) by $(I - \epsilon S)$ and then adding (r2), we get

$$Q^T w^{(2)} = Q^T\ w^* - (I - \epsilon S) Q^T w^* + (I - \epsilon S) Q^T w^* - (I - \epsilon S)^2 Q^T w^*$$

$Q^T w^{(2)} = Q^T\ w^* - (I - \epsilon S)^2 QT w^*$ (i1)

multiplying eqn. (i1) by $(I - \epsilon S)$ and then adding (r3) gives

$$Q^T w^{(3)} = (I - \epsilon S) Q^T w^* - (I - \epsilon S)^3 QT w^* + Q^T\ w^* - (I - \epsilon S) QT w^*$$

$$Q^T w^{(3)} = Q^T w^* - (I - \epsilon S)^3 QT w^*$$

Continuing in this way will give desired result.

Regularization

- But we have $\hat{w} = \mathrm{Q}(\lambda I + \mathrm{S})^{-1}\mathrm{S}\mathrm{Q}^{\mathrm{T}}w^*$

$$\mathrm{Q}^{\mathrm{T}}\hat{w} = (\lambda I + \mathrm{S})^{-1}\mathrm{S}\mathrm{Q}^{\mathrm{T}}w^*$$

$$\mathrm{Q}^{\mathrm{T}}\hat{w} = [I - (\lambda I + \mathrm{S})^{-1}\lambda]\mathrm{Q}^{\mathrm{T}}w^* \qquad \text{®}$$

$$\mathrm{Q}^{\mathrm{T}}\hat{w} = \left[I - \frac{\lambda I}{(\lambda I + \mathrm{S})}\right]\mathrm{Q}^{\mathrm{T}}w^* = \left[\frac{\lambda I + \mathrm{S} - \lambda I}{(\lambda I + \mathrm{S})}\right]\mathrm{Q}^{\mathrm{T}}w^* = \left[\frac{\mathrm{S}}{(\lambda I + \mathrm{S})}\right]\mathrm{Q}^{\mathrm{T}}w^*$$

which is same as given in ®

- Comparing this eqn. and Eqn. from last slide gives

$$I - (I - \epsilon S)^{\tau} = I - (\lambda I + \mathrm{S})^{-1}\lambda$$

$$(I - \epsilon S)^{\tau} = (\lambda I + \mathrm{S})^{-1}\lambda$$

- Taking log and expanding log(1+x), we get, if all α_i are small, i.e., $\epsilon\alpha_i \ll 1 \; and \; \frac{\alpha_i}{\lambda} \ll 1$,

$$\tau \approx \frac{1}{\epsilon\lambda} \quad \text{or} \quad \lambda \approx \frac{1}{\epsilon\tau}$$

Regularization

We will focus on a few factors that tend to influence the generalizability of a model class:

1. *The number of tunable parameters. When the number of tunable parameters, sometimes called the degrees of freedom, is large, models tend to be more susceptible to overfitting.*
2. *The values taken by the parameters. When weights can take a wider range of values, models can be more susceptible to overfitting.*
3. *The number of training examples. It is trivially easy to overfit a dataset containing only one or two examples even if your model is simple. But overfitting a dataset with millions of examples requires an extremely flexible model.*

Principal Component Analysis

Consider 3 points:

$x_1 = (0, 0, 0)^T$, $x_2 = (1, 0, 0)^T$, $x_3 = (1, 1, 0)^T$, $x_4 = (1, 0, 1)^T$.

The mean vector, covariance matrix, and normalized (unit length) eigenvectors of this population are:

$$m_x = \frac{1}{4}\begin{bmatrix}3\\1\\1\end{bmatrix}; \quad m_x m_x^T = \frac{1}{16}\begin{bmatrix}3\\1\\1\end{bmatrix}[3\ 1\ 1] = \frac{1}{16}\begin{bmatrix}9 & 3 & 3\\3 & 1 & 1\\3 & 1 & 1\end{bmatrix}$$

$$x_1 x_1^T = \begin{bmatrix}0\\0\\0\end{bmatrix}[0\ 0\ 0] = \begin{bmatrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix}; \quad x_2 x_2^T = \begin{bmatrix}1\\0\\0\end{bmatrix}[1\ 0\ 0] = \begin{bmatrix}1 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix};$$

$$x_3 x_3^T = \begin{bmatrix}1\\1\\0\end{bmatrix}[1\ 1\ 0] = \begin{bmatrix}1 & 1 & 0\\1 & 1 & 0\\0 & 0 & 0\end{bmatrix}; \quad x_4 x_4^T = \begin{bmatrix}1\\0\\1\end{bmatrix}[1\ 0\ 1] = \begin{bmatrix}1 & 0 & 1\\0 & 0 & 0\\1 & 0 & 1\end{bmatrix};$$

$$C_{xx} = E\{xx^T\} - m_x m_x^T = \frac{1}{4}\sum_{i=1}^{4} x_i x_i^T = \frac{1}{4}\begin{bmatrix}3 & 1 & 1\\1 & 1 & 0\\1 & 0 & 1\end{bmatrix} - \frac{1}{16}\begin{bmatrix}9 & 3 & 3\\3 & 1 & 1\\3 & 1 & 1\end{bmatrix}$$

Principal Component Analysis

Consider 3 points:

$x_1 = (0, 0, 0)^T$, $x_2 = (1, 0, 0)^T$, $x_3 = (1, 1, 0)^T$, $x_4 = (1, 0, 1)^T$.

$$C_{xx} = \frac{1}{16}\begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & -1 \\ 1 & -1 & 3 \end{bmatrix}$$

Its Eigen vectors and Eigen values in decreasing order are

$$e_1 = \begin{bmatrix} 0.8018 \\ 0.2673 \\ 0.5346 \end{bmatrix}; e_2 = \begin{bmatrix} -0.1543 \\ -0.7715 \\ 0.6172 \end{bmatrix}; e_3 = \begin{bmatrix} -0.5774 \\ 0.5774 \\ 0.5774 \end{bmatrix};$$

$\lambda_1 = 4$; $\lambda_2 = 4$; $\lambda_3 = 1$;

Let A = [e1 e2 e3]

Then $A * C_{xx} A^T = \begin{bmatrix} 4.000 & 0 & 0 \\ 0 & 3.999 & 0 \\ 0 & 0 & 1 \end{bmatrix}$