

# A Machine Learning Refresher

Q & A on the main concepts and terminology

Huascar Sanchez

[github.com/hsanchez](https://github.com/hsanchez)

Home Research Lab

May 6, 2020

The views expressed do not necessarily reflect the position of my employer.

## Q. What is Machine Learning?

*"Machine Learning (or **ML**) is fitting a function to examples<sup>1</sup> and using that function to generalize and make predictions about new examples."*

*Derek Jedamski, GitHub*

Machine Learning, by large, falls into three categories:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

And solves two types of problems: Classification and Regression.

---

<sup>1</sup>An example, or a sample, is a data point that belongs to some data

## Q. What is Supervised Learning?

In **Supervised Learning** (or **SL**), you are given a bunch of examples and their labels (e.g., A or B) and the goal is to classify, when you are given a new example, to which label we should assign the new example.

You could think of these labels as the names of the **classes or clusters** to which certain portions of the data belong.

## Q. Can you give an example of Supervised Learning?

**Support vector machines or SVM**, is a ML algorithm that takes some data as input and returns as output and optimal separating *hyperplane* between pieces of data; e.g., a plane separating A from B.

- In this example, A and B sit in some high dimensional space, and the idea is to construct a plane that maximizes the margins<sup>2</sup> between the plane and the data.
- If a new datum sits closer to one area of the data, say A, then we assign this new datum to A.

(For historical reasons) This algorithm is called **support vector machines** because the vectors that lie on the margin of the plane are called the *support* vectors.

This is a method for constructing a device to discriminate. If we're having a supervised learning problem then this method gives me an optimal form of discrimination.

---

<sup>2</sup>distance between points closest to the line and the actual line

## Q. What more can you say about SVM?

**SVM** works on both linearly and non-linearly separable data. In latter case, it use the kernel trick<sup>3</sup> to convert these data to linearly separable data in a higher dimension. This transformation is called kernel.

A hyperplane in an  $n$ -dimensional Euclidean space is a flat,  $n - 1$  dimensional subset of that space that divides the space into two disconnected parts.

---

<sup>3</sup>adds one more dimension and call it  $z$ -axis – governed by the constraint  $z = x^2 + y^2$  and  $z$  is the squared distance of the points from origin.

## Q. Tuning SVM Hyper-parameters?

### Hyperparameter and hyper-parameter tuning:

**Kernel:** The main function of the kernel<sup>4</sup> is to transform the given dataset input data into the required form. There are various types of functions such as linear, polynomial, and radial basis function (RBF). Polynomial and RBF are useful for non-linear hyperplane.

**Regularization:**  $C$  is regularization parameter that controls the trade-off between smooth decision boundary<sup>5</sup> and classifying training points correctly. A large  $C$  means you'll get more training points correctly.

**Gamma:** Gamma defines how far the influence of a single training example reaches. A large gamma means the decision boundary is just going to be dependent upon the points that are very close to the line.

---

<sup>4</sup>Kernels can lead to more accurate classifiers.

<sup>5</sup>a decision boundary or decision surface is a hyper-surface that partitions the underlying vector space into two sets, one for each class.

## Q. Advantages and Disadvantages SVM?

### Advantages

SVM Classifiers offer **good accuracy and perform faster prediction** compared to Naive Bayes algorithm. They also use less memory because they use a subset of training points in the decision phase. SVM works well with a clear margin of separation and with high dimensional space.

### Disadvantages

SVM is **not suitable for large datasets** because of its high training time and it also takes more time in training compared to Naive Bayes. It **works poorly with overlapping classes** and is also **sensitive** to the type of **kernel** used.

## Q. What is Unsupervised Learning?

In **Unsupervised Learning** (or **UL**), you are given a bunch of data and you are not told they fall naturally into clusters, and also you are not told what these clusters are.

The goal is identify the clusters within data, how many clusters there are, and then be able to assign new things to these different clusters.



## Q. Can you give an example of Unsupervised Learning?

Principal Component Analysis (or **PCA**) is a classical UL algorithm.

In **PCA**, the way this works, we construct a *covariance matrix*, and my covariance matrix is just the following object:  $C = \sum_j \vec{v}_j \vec{v}_j^+$ , where  $\vec{v}_j^T$  is the transpose of  $\vec{v}_j$ .

- In other words, we construct  $C$  from the data by taking these vectors  $\vec{v}_j$  and multiply them by their transpose  $\vec{v}_j^+$ .

In **PCA**, you diagonalize  $C$  and say  $C = \sum_k P_k \vec{\omega}_k \vec{\omega}_k^+$ <sup>6</sup>,  $P_k$  is piece of the data with size  $k$ , and  $\vec{\omega}_k$  are the set of vectors you need to find.

If and only if a small set of  $P_k \gg 0$ , then  $C$  is effectively low-rank, and the corresponding  $\vec{\omega}_k$  are the principal components. In other words, you need find the eigenvectors that have the largest eigenvalues – i.e., your principal components.

---

<sup>6</sup> $C$  can be decomposed into a product of matrices involving eigenvalues and eigenvectors

## Q. What else can you say about PCA?

In general terms, this PCA process is about finding the underlying patterns of the data and also gives you a method for data compression.

That is, if there is only a few principal components then this means that all my vectors can be written as the sum of a few  $w$ s.

PCA is all about diagonalizing the covariance matrix.

PCA is an exercise in linear algebra on very high dimensional vector spaces.

# Q. When to use PCA?

## When to use it

PCA should be used if one to figure out if there are latent features driving the patterns in the data. E.g., The big shots at SRI International.

Dimensionality reduction. E.g., helps you visualize high dimensional data, helps you reduce noise in your data, make other ML algos work better (regression, classification) because fewer inputs.

## When not to use it

PCA is not suitable in many cases: For example, if all the components of PCA have quite a high variance, there is no *good* universal stopping rule that allows you to discard some exact  $k$  Principal Components, meaning no good data compression.

Not good when working with fine-grained classes<sup>7</sup>.

---

<sup>7</sup>hard-to-distinguish object classes

## Q. Recap: Advantages and Disadvantages of PCA?

### Advantages

- Dimensionality reduction

- Allows estimation of probabilities in high-dimensional data.

- It renders a set of components that are uncorrelated.

### Disadvantages

- It has a high computational cost, meaning it'll struggle with very large datasets.

- Non-linear structure is hard to model with PCA.

## Q. What is Reinforcement Learning?

Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

In reinforcement learning an algorithm learns to do something by being rewarded for successful behavior and/or being punished for unsuccessful behavior.

## Q. Can you give an example of Reinforcement Learning?

### Policy Gradient (PG)

In this method, we have the policy  $\pi$  that has a parameter  $\theta$ . This  $\pi$  outputs a probability distribution of actions.

Then we must find the best parameters ( $\theta$ ) to maximize (optimize) a score function  $J(\theta)$ , given the discount factor  $\gamma$  and the reward  $r$ .

Main steps: (1) Measure the quality of a policy with the policy score function, and (2) Use policy gradient ascent to find the best parameter that improves the policy.

# Examples of Classification and Regression problems

## Q. Can you give an example of a Classification Problem?

E.g., trying to write a machine learning program that will be able to detect cancerous tumors in lungs. It takes in images of lung x-rays as input and determines if there is a tumor. If there is a tumor, we'd like the computer to output "yes" and if there is not a tumor, we'd like the computer to output "no." We'd like the computer to output the correct answer as much as possible.

Say the training set for this algorithm consists of several images of x-rays, half of the images contain tumors and are labelled "yes" and the other half do not contain tumors and are labelled "no."



## Q. Can you give an example of a Regression Problem?

In regression problem the goal of the algorithm is to predict real-valued output.

E.g., Let's consider the problem of predicting the marks of a student based on the number of hours he/she put for the preparation.

Assume for the sake of understanding that the marks of a student ( $M$ ) do depend on the number of hours ( $H$ ) he/she put up for preparation.

The following formula can represent the model:  $M = m * H + c$

## Q. Classification vs Regression?

Classification is the task of predicting a discrete class label.

Regression is the task of predicting a continuous quantity.

There is some overlap between the algorithms for classification and regression; for example

- A classification algorithm may predict a continuous value, but the continuous value is in the form of a probability for a class label.

- A regression algorithm may predict a discrete value, but the discrete value is in the form of an integer quantity.

# Data representation

## Q. How do you represent data in ML?

In general, the given data is expressed in a form of a bunch of vectors  $\vec{v}_j \in \mathbb{R}^d$  that belong to some high dimensional vector space.

For instance, in image recognition, the vector<sup>8</sup> of an each image is a set of pixels<sup>9</sup> (i.e., a pixelated version of the image).

If you have a notion of distance  $\Delta(\vec{v}_i, \vec{v}_j)$ , then you can compare which vectors are close to each other in this high dimensional vector space; e.g., the norm  $\|\vec{v}_i - \vec{v}_j\|^2$ .

---

<sup>8</sup>a.k.a., feature vector; it could be numerical (e.g., height of tree) or descriptive (e.g., eye color)

<sup>9</sup>each entry in the vector (e.g., pixel) represents a feature.

## Q. What is it important about measuring distances?

Because the shorter the distance between two feature vectors the closer in character are the two samples they represent.

There are many ways to measure distance:

- Manhattan distance ( $L^1$  norm). The sum of the absolute values of the different between entries in the vector. (Preferred distance when dealing with data in high dimensions.)
- Euclidean distance ( $L^2$  norm). Square the distances between vector entries, sum these and square root.
- Cosine similarity. Cosine of the angle between two vectors<sup>10</sup>. Just take the *dot* product of two vectors and divide by the two lengths.

---

<sup>10</sup>Two vectors might be similar if they are pointing in the same direction even if they are of different lengths.

## Q. Any other issues we should care to learn?

The curse of dimensionality. This phenomena involves data in high dimensions.

Suppose you are working in  $M$  dimensions, that is you data has  $M$  features. And suppose you have  $N$  data points. Having a large number of data points is good, the more the merrier. BUT what about number of features?

Think how these  $N$  data points might be distributed in  $M$  dimensions. Suppose that the numerical data for each feature is 0 or 1. There will therefore be  $2^M$  possible combinations.

If  $N$  is less than  $2^M$  then you run the risk of having every data point being on a different corner of the  $M$ -dimensional hypercube.

# Building a Machine learning model

## Q. What does this process looks like?

**Explore and clean the data.** Explore the data to really understand what those features look like, then we use some of these learnings to clean the data.

**Split data into train/validation/test data sets.**

**Fit an initial model (baseline) and evaluate using 5-fold cross-validation.**

**Tune hyper-parameters using GridSearch + 5-fold CV** to find the best models<sup>11</sup>.

**Evaluate best models on validation set** and select the top model of each algorithm, and we evaluate them against each other on the validation set.

**Select and evaluate the final model on the test set.** We select the top model based on the validation set and we will evaluate it on the test set to get an unbiased view of the model performance on completely unseen data.

---

<sup>11</sup>those models that beat our baseline



# Measuring success

## Q. What does it mean to measure success of the model?

Now that we have a decent idea of how to explore and clean our data, we want to start setting the stage for building a model by understanding what it means to measure success of the model.

- The first step is walking through why it's necessary to split up our data before we build a model. Please recall the definition of ML that we settled on back in the first section of these slides. Particularly, recall the part on generalizing to make predictions about new examples. This part is about the model needing to learn the underlying pattern (or correlation) that will allow it to make predictions about future examples.
- So the question is how do we make sure the model is learning the underlying pattern and not just memorizing the examples? We can do this by splitting our data into three data sets: the training, validation, and testing data sets. Next section discusses these in detail.

## Q. Why do we split the data?

To make sure that a model is learning the underlying pattern and not just memorizing the examples. We don't know how well the model will generalize because we don't have any additional data to test this.

We can solve that by splitting our dataset into three separate segments; training (60%), validation (20%), and testing (20%) data sets.

training data, or examples, that the model will learn those general patterns from.

validation data is used to select the best model (optimal algorithm and hyper-parameter settings).

test data is used to provide an unbiased evaluation of what the model will look like in its real environment.

## Q. Why does the evaluation process look like with these datasets?

You start train your ML algorithm on the training data, evaluate it using the validation data, then at this point.

If none of your models are any good based on the performance on the validation set, then you need to revisit the training phase and consider some new variables or new models.

If the performance is quite good, then you can select your best model and pass it onto the testing phase.

Now, if you evaluate it on the test set and performance is what you expect, then that model is ready to go.

## Q. What is cross-validation? I.e., Holdout test set

The previous process how to use training, validation and test set to measure our model's ability to generalize.

Now, we're going to add one additional layer on top of this process; i.e., something called cross validation. Before describing this process, we need to define a few terms first:

**Holdout test set:** A generalization of the test set. It's just any data set that was not used in fitting a model that is set aside for evaluating the model's ability to generalize.

**K-Fold Cross-Validation:** Data is divided into  $k$  subsets and the holdout method is repeated  $k$  times. Each time, one of the  $k$  subsets is used as the test set and the other  $k - 1$  subsets are combined to be used to train the model.

## Q. 5-Fold Cross-Validation Example?

E.g., 5-fold CV: Given 10,000 examples, you partition into 5 buckets, each consisting of 2,000 examples<sup>12</sup>.

On trial 1, we set aside the last bucket (holdout test set) and the other 4 buckets are the training set, then compute its predictive performance. On trial 2, we set aside the penultimate bucket, and so on. At the end we *average* the performance of model.

---

<sup>12</sup>this is sampling without replacement; no single example will appear in two different subsets and all original 10,000 are still accounted for in these subsets

## Q. Establishing our Evaluation framework?

A cohesive framework for evaluating our model, consisting of two components:

**Evaluation metrics:** how are gauging the accuracy of the model?

**Process (how to split the data):** how do we leverage a given data set to mitigate the likelihood of overfitting and underfitting.

**There isn't a one-size-fits-all metric.**



## Q. What are the evaluation metrics that we'll use?

The metric(s) chosen to evaluate a ML model depends on various factors:

Is it a regression or a classification task? E.g., MSE vs Accuracy.

What is the business objective? E.g., precision vs recall.

What is the distribution of the target variable?

Examples of other metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), R-squared, Confusion Matrix and related metrics (Precision, Recall, Accuracy).

## Q. Metrics for classification problem?

So, for a classification problem, we're going to use three commonly used evaluation metrics. The first is accuracy:  $\frac{\# \text{predicted correctly}}{\text{total} \# \text{ of examples}}$

The next evaluation metric is something called precision:  $\frac{\# \text{predicted as class A that actually is class A}}{\text{total} \# \text{ predicted to be in class A}}$

The last evaluation metric is something called recall:  $\frac{\# \text{predicted as class A that actually is class A}}{\text{total} \# \text{ that we actually are in class A}}$

## Q. Can you explain what precision and recall are?

**Recall** is a measure of completeness or quantity, whereas

**Precision** is a measure of exactness or quality:

$$\underbrace{Recall = \frac{TP}{TP + FN}}$$

What proportion of actual positives  
was identified correctly?

$$\underbrace{Precision = \frac{TP}{TP + FP}}$$

What proportion of positive identifications  
was actually correct?

In simple terms, **high** precision means your algorithm has returned substantially *more relevant results than irrelevant ones*, while **high** recall means your algorithm has returned *most of the relevant results*.

## Q. What is a confusion matrix?

A **Confusion Matrix** is a simple way of understanding **how well an algorithm is doing at classifying data**. It is just the idea of **false positives** and **false negatives**

		Predicted	
		Yes	No
Actual	Yes	TP	FN
	No	FP	TN

## Q. Can you explain what are false positives and false negatives?

There are two errors that often rear their head when you are learning about hypothesis testing -- false positives and false negatives, technically referred to as type I error and type II error respectively.

**False positives** are incorrect classifications of the presence of a condition when it is actually absent. A **false positive** is **when you reject a true null hypothesis**.

**False negatives** are incorrect classifications of the absence of a condition when it is actually present. A **false negative** is **when you accept a false null hypothesis**.

**Q. Provide examples when false positives are more important than false negatives, false negatives are more important than false positives and when these two types of errors are equally important**

E.g., cancer screening. It is much worse to say that someone doesn't have cancer when they do (false negative) rather than saying that someone has it when in fact they don't (false positive).

E.g., (from a psychological perspective) pregnancy test. Given the null hypothesis: "I am not pregnant." A false negative for someone who really does not want a child, is not ready for one and when assuring themselves with a negative result.

E.g., forecasting (health weather). During the covid-19 outbreak. Tracking rate of false positives and false negatives in March 2020 in the United States of America.

## Q. Can you list some of metrics derived from Confusion Matrix?

Confusion matrix related metrics

Accuracy rate:  $\frac{TP+TN}{Total}$  where Total is  $TP + TN + FP + FN$ ,

Error rate:  $1 - \frac{TP+TN}{Total}$ ,

False positive rate:  $\frac{FP}{TN+FP}$ ,

Recall:  $\frac{TP}{TP+FN}$ , Precision:  $\frac{TP}{TP+FP}$ ,

Specificity:  $1 - \frac{FP}{TN+FP}$ ,

Prevalence:  $\frac{TP+FN}{Total}$

**Q. How do you measure how good your classification algorithm is?  
You have unbalanced numbers, with our class being much larger or smaller than others**

You use the Matthews correlation coefficient. The number it yields is between plus or minus one. Plus one means perfect prediction, zero means no better than random.

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$



## Q. Is there another way to measure how well an algorithm is classifying data? Also given an example

Another way of looking at how well an algorithm is classifying is the **receiver operating characteristic** or **ROC** curve.

E.g., suppose there is a threshold (or parameter) in your classification algorithm that determines whether you have an apple or a non apple object.

- Plot the *true positive* rate against the *false positive* rate as the threshold (or parameter) varies. Points above the  $45^\circ$  diagonal are good, and the further into the top left of the plot the better.
- The area under the ROC curve (the **AUC**<sup>13</sup>) is then a measure of how good different algorithms are. The closer to one (the max possible) the better.

---

<sup>13</sup>AUC is used to rank kaggle competitions

# Process (how to split the data)

## Q. Process description?

Run fivefold cross-validation and select the best models.

Re-fit models on full training set, evaluate those models on the validation set and pick the best one.

Evaluate that best model on the test set to gauge its ability to generalize to unseen data.

**What do we mean by training, best fit,  
performance? Any risks?**

## Q. But, what is training in ML anyway?

Most ML algorithms need to be trained. That is, you give them data and they look for patterns, or best fits, etc.

They know they are doing well when perhaps a loss function has been minimized, or the rewards have been maximized.

## Q. Best fits? What do you mean by best fits? Performance?

First of all, when we say “fitting” in ML we are talking about model fitting.

Model fitting is a measure of how well a machine learning model generalizes<sup>14</sup> to similar data on which it was *trained*.

A model that is well-fitted produces more accurate outcomes<sup>15</sup>.

---

<sup>14</sup>Yes, we are talking about ML model performance.

<sup>15</sup>A model that is overfitted matches the data too closely. A model that is underfitted doesn't match closely enough.

## Q. How do you measure the performance of a ML model?

We do that by using a cost function or a loss function.

In ML, a cost function is used to represent how far away a mathematical model is from the real data.

A common way to do this is via the quadratic cost function<sup>16</sup>:

$$J(\theta) = \frac{1}{2N} \sum_{n=1}^N (h_{\theta}(x^{(n)}) - y^{(n)})^2.$$

We are interested in the parameters that minimize this quadratic cost function. This is called ordinary least squares (OLS).

One adjusts the mathematical model usually by varying parameters within the model, so as to *minimize the cost function*. This is interpreted as given the best model, of its type, that fits the data.

---

<sup>16</sup>This is just the sum of the squares of the vertical distances between the points and the straight line

## Q. What are the risks of not splitting the data?

Overfitting or underfitting to the data

Inaccurate representation of how the model will generalize



## Q. Any caveats on training and testing?

Over many epochs, if the test error begins to rise (and it's much bigger than the training error) then you have overfitted.

To help avoid overfitting sometimes we divide up our original data into three sets. The third data set is the validation data set. (One can use cross validation to evaluate performance of model using training, test, and validation data sets.)

# Model optimization

## Q. Model optimization outline?

We'll discuss the bias-variance trade-off.

We'll cover what we mean by bias and variance from a conceptual level.

## Q. Bias and Variance in Machine learning?

Bias<sup>17</sup>, in ML, is the algorithm's tendency to consistently learn the wrong thing by not taking into account all the information in the data. (results in inaccurate predictions).

Variance<sup>18</sup> is an algorithm's sensitivity to small fluctuations in the training data set.

---

<sup>17</sup>High bias is the result of the algorithm missing the relevant relations between features and target outputs

<sup>18</sup>High variance is a result of the algorithm fitting to random noise in the training data

## Q. Can you be more specific about Bias and Variance?

**Bias** is how far away (or error) the trained model is from the correct result *on average*. Where “on average” means over many goes at training the model, using different data:

$$\text{Bias}(\hat{f}(x')) = \underbrace{\mathbb{E}[\hat{f}(x)]}_{\text{Average error}} - f(x')$$

**Variance** is a measure of the magnitude of that error.

$$\text{Var}(\hat{f}(x')) = \mathbb{E}[\hat{f}(x)^2] - \mathbb{E}[\hat{f}(x')]^2$$

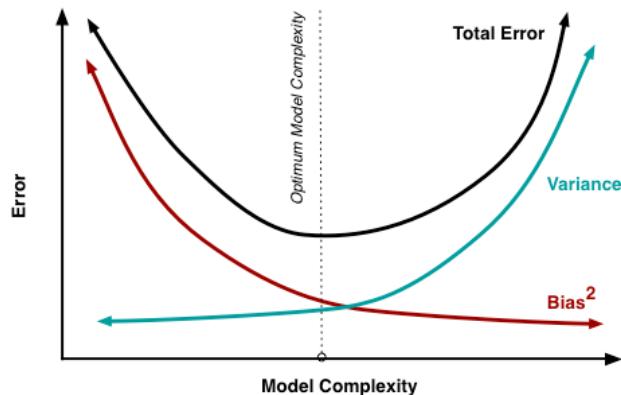
We often find there is a trade-off between bias and variance. As one is reduced, the other is increased<sup>19</sup>.

---

<sup>19</sup>This the matter of over-and-underfitting

## Q. Bias and Variance tradeoff?

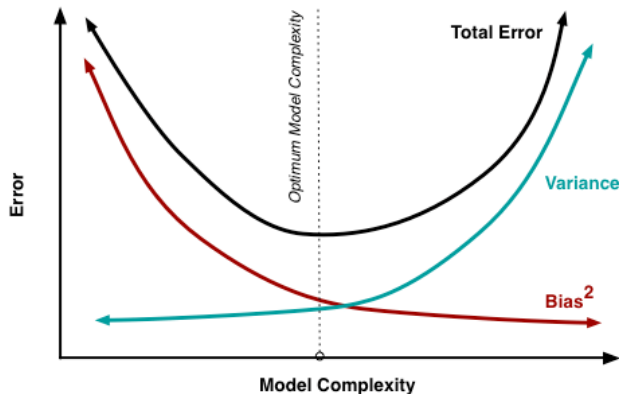
Total error = (Bias + Variance) + Irreducible Error



Model complexity is across the x axis and model error across the y axis. More complexity means higher variance. Lesser complexity means higher bias.

## Q. Bias and Variance tradeoff?

Total error = (Bias + Variance) + Irreducible Error

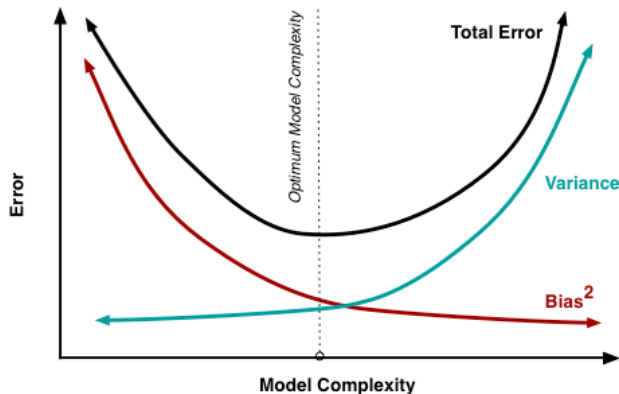


Model complexity is across the x axis and model error across the y axis. More complexity means higher variance. Lesser complexity means higher bias.

So this what bias/variance tradeoff is all about: **finding the right model complexity** that minimizes both bias and variance (I mean the total error as much as possible).

## Q. Bias and Variance tradeoff?

Total error = (Bias + Variance) + Irreducible Error



Model complexity is across the x axis and model error across the y axis. More complexity means higher variance. Lesser complexity means higher bias.

So this what bias/variance tradeoff is all about: **finding the right model complexity** that minimizes both bias and variance (I mean the total error as much as possible).

Total error is very high for very simple models and a very complex model, and then it bottoms out in the middle.



## Q. What is Underfitting?

(Recall high bias.) Underfitting occurs when an algorithm cannot capture the underlying trend of the data. This happens when the model is too simple with high bias and low variance, and results in high total error.

Underfitting: High bias + low variance (The left side of this plot represents underfitting).

## Q. What is Overfitting?

(Recall high variance.) Overfitting occurs when an algorithm fits too closely to a limited set of data (i.e., training set). In other words, the model might just memorize the examples that it has seen in the training data.

Then when it sees a new example that looks a lot like an example it has seen before it can make a very accurate prediction.

And when it sees a new example that looks nothing like any other examples it has seen before it will make a very poor prediction.

Overfitting: Low bias + high variance (The right side of this plot represents overfitting).

**The actual process?**

## Q. How do you find the optimal tradeoff?

The goal is to find something in the middle (a model with medium complexity). This would learn the true pattern in the data w/o memorizing every example in the training data.

Optimal tradeoff: Low bias + Low variance. OKAY, but how do you identify underfit and overfit?

With underfit, we'll have high training error and high test error.

Optimal tradeoff, we'll have low training error and low test error.

With overfit, we'll have low training error and high test error.

## Q. How do you tune a model for optimal complexity?

There are two methods to tune a model for optimal complexity:

- Hyper-parameter tuning – choosing a set of optimal hyper-parameters for fitting a ML algorithm (e.g., linear regression)

- Regularization – a technique used specifically to reduce overfitting by discouraging overly complex models in some way.

## Q. What is a hyper-parameter?

A model **parameter** is a configuration variable that is internal to the model and *whose value can be estimated from data*.

A model **hyper-parameter** is a configuration that is external to the model and *whose value cannot be estimated from data, and whose value guides how the algorithm learns parameter values from the data*. E.g., depth of a decision tree is a model hyper-parameter vs ticket price or ticket class are model parameters.

## Q. What is Regularization?

Regularization is a form of regression, which constrains/regularizes or shrinks the (learned) coefficient estimates towards zero. In other words, this technique reduces overfitting by discouraging learning a more complex model in some way.

The goal of Regularization is to allow enough flexibility for the algorithm to learn the underlying patterns in the data but provides guardrails so it doesn't overfit. See Occam's razor – whenever possible, choose the simplest answer (or model) to a problem.

## Q. Can you provide Regularization examples?

**Ridge regression and lasso regression:** adding a penalty to the loss function (See Model fitting slide) to constrain coefficients.

**Dropout:** some nodes are ignored during training which forces the other nodes to take on more or less responsibility for the input-output.



# Epoch vs Batch size vs Iteration

## Q. What is the difference between these terms?

To find out the difference between these terms you need to know some of the machine learning terms like Gradient Descent to help you better understand.

## Q. What is Gradient Descent?

Gradient descent is an *iterative* optimization algorithm used to minimize some *convex* function by *iteratively* moving in the direction of steepest descent as defined by the negative of the gradient.

In ML, we use gradient descent to update the parameters of a ML model.

Start with an initial guess for each parameter  $\theta_k$ . Then move  $\theta_k$  in the direction of the slope.

Update all  $\theta_k$  simultaneously (known as batch gradient descent), and repeat until convergence.

## Q. Unpacking Gradient Descent?

Gradient means the rate of inclination or declination of a slope.

Descent means the instance of descending.

The algorithm is iterative means that we need to get the results **multiple times** to get the most optimal result (minima of a curve).

## Q. What is Stochastic Gradient Descent?

Similar to batch gradient descent except that you only update using one of the data points each time.

And that data point is chosen randomly. Hence the stochastic.

In other words, stochastic gradient descent pick an  $n$  at random and then update using one of the data points. Repeat, picking another data point at random, etc.

## Q. What is an epoch?

In some ML methods, one uses the same training data many times, as the algorithm gradually converges, for example, in stochastic gradient descent. Each time the whole training set of data is used in the training that is called an **epoch**<sup>20</sup>.

One might see a decreasing error as the number of epochs increases. But that doesn't mean your algorithm is getting better, it could easily mean that you are overfitting<sup>21</sup>. To test for this you introduce a test data set, the data that you've held back.

---

<sup>20</sup>Typically you won't get decent results until convergence after many epochs

<sup>21</sup>This could happen if the learning algorithm has seen the training data so many times or epochs

## Q. So, what is the right numbers of epochs?

Unfortunately, there is no right answer to this question. The answer is different for different data sets but you can say that the numbers of epochs is related to how diverse your data is. For example, do you have only black cats in your data set or is it much more diverse dataset?

## Q. What is a batch?

As I said, you can't pass the entire data set into a ML algorithm at once. So, you divide data set into a number of batches or sets or parts.



## Q. What is an iteration?

An iteration is the number of batches needed to complete one epoch.

Let's say we have 2000 training examples that we are going to use.

We can divide the dataset of 2000 examples into batches of 500 then it will take 4 iterations to complete 1 epoch.

# End-to-End Pipeline

(starting from fit initial model)

Run fivefold cross-validation and select the best models.

## Q. Fit a basic model using cross validation?

Goal: To understand what the baseline performance looks like <sup>22</sup>.

From sklearn, import RandomForestClassifier, and cross-val-score to compute the accuracy of the baseline model.

Next, we will select other models in order to beat our baseline.

---

<sup>22</sup>We'll use only the training set

## Q. Tuning Hyper-parameters?

Run grid search to find the optimal hyper-parameter settings for our models.

Our goal of this step is to find the optimal model that beats that baseline performance.

We use GridSearchCV<sup>23</sup>, which yields multiple combinations of hyper-params values, to find the combinations that improves performance<sup>24</sup>.

---

<sup>23</sup>a wrapper around cross fail score that allows us to run grid search within cross validation.

<sup>24</sup>Recall that a model is a configuration of the ML algorithm on specific params values

Re-fit models on full training set,  
evaluate those models on the validation  
set and pick the best one.

## Q. Evaluating results on Validation set?

Now that we've done some hyper parameter tuning, and we have a good idea of what the best hyper parameter combinations are, let's evaluate these models on the validation set<sup>25</sup>.

The process goes like this:

- 1 Look at additional performance metrics beyond just accuracy (e.g., precision and recall) and also at different ML algorithms.
- 2 Take the 3 best performing models and **re-fit** them using the whole training data<sup>26</sup>.
- 3 Evaluate these models on the validation set<sup>27</sup>.
- 4 Select the best performing model.

---

<sup>25</sup>Now the performance shouldn't deviate too much from the performance we saw with the cross-validation.

<sup>26</sup>Why do we need to do that? Because these models were fit on only 80% of the training data; we need now the entire training data.

<sup>27</sup>This is the truth test. If they are overfit or underfit, then they will fail here.

Evaluate that best model on the test set  
to gauge its ability to generalize to  
unseen data.



## Q. Final model selection and evaluation on test set?

Next, we'll evaluate the best model on the test set. This will give us a truly unbiased view of how it should perform moving forward.

Why do we need to evaluate it on test data, given they have already evaluated on unseen data?

Because, the validation set was created by just randomly distributing examples from the full data set.

So by nature of the fact that testing on the validation set impacted what model was chosen as our final model, it's technically part of the model training process.

So this final test set is purely for evaluation purposes to see that it matches the performance that we've seen before and to give us more confidence in the performance of the model moving forward.

# Machine Learning Algorithms

# Regression algorithms

## Q. What is Regression?

**Regression** is a statistical process for estimating the relationship among variables, often used to make a prediction about some outcome.

## Q. What is linear regression?

**Linear regression** is one type of regression that is used when you have a continuous target variable (output).

For instance, to predict the number of umbrellas sold by the amount of rainfall and the algorithm definition for linear regression is  $y = mx + b$ , where  $y$  is the dependent variable,  $x$  is the independent variable, and  $b$  and  $m$  (i.e., slope) are coefficients dictating the equation<sup>28</sup>.

---

<sup>28</sup>e.g., a model that assumes a linear relationship between the input

## Q. What is logistic regression?

**Logistic regression** is a form of regression where the target variable is binary (zero or one, or true or false). It takes the form:  $\frac{1}{1 + e^{-(mx+b)}}$

Let's say I wanted to examine the relationship between my basketball shooting accuracy and the distance that I shoot from. More specifically, I want a model that takes in "distance from the basket" in feet and spits out the probability that I will make the shot (1 for a make, 0 for a miss).

In this context,  $-(mx + b)$ , labeled as  $z$ , represents  $\log(\text{odds of making shot})^{29}$ , so the probability of making a shot (i.e.,  $y$ ) is  $\frac{1}{1 + e^{-z}}$ . Here,  $m$  and  $b$  are the coefficients we're interested in finding thru optimization, and  $x$  is *distance from basket*.

---

<sup>29</sup>odds =  $P(\text{Event}) / [1 - P(\text{Event})]$

## Q. Logistic regression example?

**Shooting Baskets.** Generally, the further I get from the basket, the less accurately I shoot: when given a small distance, the model should predict a high probability and when given a large distance it should predict a low probability.

In logistic regression the output  $Y$  is in log odds. Let's say I shot 100 free throws and made 70. Based on this sample, my probability of making a free throw is 70%. My odds of making a free throw is 2.33, which we want to bound between 0 and 1 using *log* and the *sigmoid* function, as odds go from 0 to  $INF$ <sup>30</sup>:

We can write our logistic regression equation:  $z = \log(mx + b)$ , where  $m$  and  $b$  are the coefficients to be learned.

And to get probability from  $z$ , which is in log odds, we apply the sigmoid function:  
 $\frac{1}{1+e^{-z}}$ <sup>31</sup> In this case, a high probability means I will be able to shoot and a low probability means I won't.

---

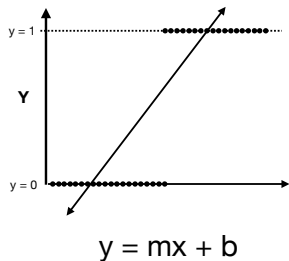
<sup>30</sup>log odds are just probability stated another way

<sup>31</sup>This shows how we can go from a linear estimate of log odds to a probability.

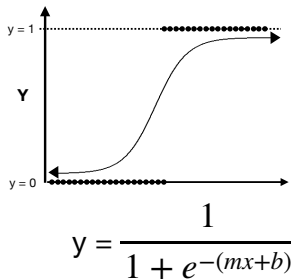
# Q. Why won't Linear Regression work for binary target variables?

In other words, why do we need two regression algorithms?

Linear regression



Logistic regression



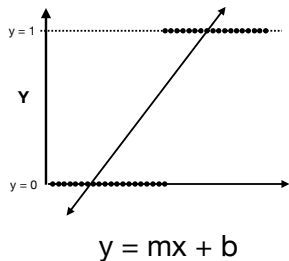
**Linear regression** for binary target variable will have a hard time try to come up with a best fit line that makes any sense.



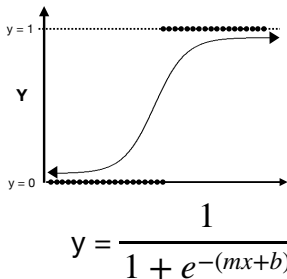
# Q. Why won't Linear Regression work for binary target variables?

In other words, why do we need two regression algorithms?

Linear regression



Logistic regression



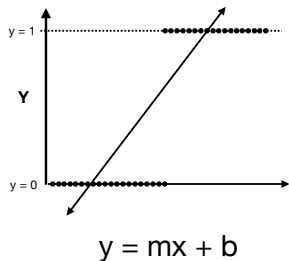
**Linear regression** for binary target variable will have a hard time try to come up with a best fit line that makes any sense.

It'll try to fit a line that fits all of the data and it will end up predicting negative values and values over one, which is impossible.

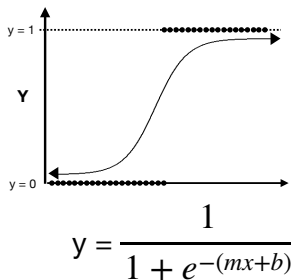
# Q. Why won't Linear Regression work for binary target variables?

In other words, why do we need two regression algorithms?

Linear regression



Logistic regression



**Linear regression** for binary target variable will have a hard time try to come up with a best fit line that makes any sense.

It'll try to fit a line that fits all of the data and it will end up predicting negative values and values over one, which is impossible.

**Logistic regression** is built off of a logistic or sigmoid curve (S shape) and will always be between zero and one, which makes it a better fit for a binary classification problem.

## Q. When should you consider using Logistic Regression?

**When to use?** Consider using it any time you

- You have a binary target variable.
- You're interested in feature importance or having a better understanding of what's going on within the algorithm.
- You have well-behaved<sup>32</sup> data, need a **quick** initial benchmark.

**When not to use?** Consider not using it any time you

- You have a continuous target variable.
- You have a massive amount of data<sup>33</sup>.
- You have unwieldy<sup>34</sup> data, performance<sup>35</sup> is the only thing that matters.

---

<sup>32</sup>e.g., no many outliers, no many missing values, no complex relationships

<sup>33</sup>e.g., lots of features and very few rows, or lots of rows and very few features

<sup>34</sup>e.g., many outliers, many missing values, complex relationships

<sup>35</sup>will usually do pretty well on any given problem, but will rarely be the best.

**Q. What is the general form of multiple regression?**

The general form of the equation for linear regression.

## Q. What is KMeans?

**KMeans** is a unsupervised ML algorithm. You have a bunch of vectors. Then you take a bunch seed-guesses for the centers of the clusters. You assign each vector to the nearest cluster. And then, when everything is done, you calculate the mean values of the clusters. And then you just do it again: you re-assign the nearest mean. Then you keep on going until it converges.