A Machine Learning Refresher

Q & A on the main concepts and terminology

Huascar Sanchez
`github.com/hsanchez`

Home Research Lab

May 5, 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[1] and using that function to generalize and make predictions about new examples.

Machine Learning, by large, falls into three categories:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

And solves two types of problems: Classification and Regression.

[1] 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 name 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**, which goal is to construct the optimal separating *hyperplane* between pieces of data given input data; e.g., a plane separating A from B.

- A and B sit in some high dimensional space, and the idea is to construct a plane that maximizes the margins[2] 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.

[2]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[3] to convert these data to linearly separable data in a higher dimension.

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.

Hyperparameters:

- C is regularization parameter that controls the trade-off between smooth decision boundary and classifying training points correctly. A large C means you'll get more training points correctly.

- *Gamma*. It 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.

[3]adds one more dimension and calls it z-axis – governed by the constraint $z = x^2 + y^2$ and z is the square of distance of the point from origin.

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: $C = \sum_j \vec{v}_j \vec{v}_j^\dagger$, 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 $\vec{v}_j^\dagger$.

In **PCA**, you diagonalize C and say $C = \sum_k P_k \vec{\omega}_k \vec{\omega}_k^\dagger$[4], 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 $P_k >> 0$, 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.

[4] C can be decomposed into a product of matrices involving involving eigenvalues and eigenvectors

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.

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[5] of an each image is a set of pixels[6] (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 $\left\|\vec{v}_i - \vec{v}_j\right\|^2$.

[5] a.k.a., feature vector; it could be numerical (e.g., height of tree) or descriptive (e.g., eye color)
[6] 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[7]. Just take the *dot* product of two vectors and divide by the two lengths.

[7] 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 and evaluate. We will use 5-fold cross validation o the training set to fit an initial model to see what we can expect for the baseline performance of the model.

Tune hyper-parameters. We use the same 5-fold cross validation paired with grid search to explore a variety of different hyper-parameter settings for each algorithm.

Evaluate on validation set. We 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.

Measuring success

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. 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[8] to similar data on which it was *trained*.

A model that is well-fitted produces more accurate outcomes[9].

[8] Yes, we are talking about ML model performance.

[9] 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[10]:

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

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.

[10] 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.)

Our evaluation framework and cross-validation

Q. What are the components of the Evaluation framework?

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.

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

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

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

- Is it a regression or 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: $\frac{\#predictedcorrectly}{total\#ofexamples}$

The next evaluation metric is something called precision: $\frac{\#predictedasclassAthatactuallyclassA}{total\#predictedtobeinclassA}$

The last evaluation metric is something called recall: $\frac{\#predictedasclassAthatactuallyclassA}{total\#thatweactuallyinclassA}$

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:

$$\underbrace{Recall = \frac{TP}{TP + FN}}_{\text{What proportion of actual positives was identified correctly?}} \qquad \underbrace{Precision = \frac{TP}{TP + FP}}_{\text{What proportion of positive identifications was actually correct?}}$$

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: $\frac{TP}{TP+TN}$, 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.

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

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° diagonal are good, and the further into the top left of the plot the better.

- The area under the ROC curve (the **AUC**[11]) is then a measure of how good different algorithms are. The closer to one (the max possible) the better.

[11]AUC is used to rank kaggle competitions

Q. What is cross-validation?

Cross-validation is a technique for assessing how well a model performs on new independent data (holdout test set[12]). (how robust the model is.)

The simplest example of cross-validation is when you split your data into three groups[13]: (1) training data, (2) validation data, and (3) testing data.

both validation and test data sets are a type of holdout set.

[12] sample of the data not used in fitting a model; used to evaluate the model's ability to generalize to unseen examples

[13] e.g., a ˜60%-˜20%-˜20% split

Q. What do you do in (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.

E.g., given $10,000$ examples, we divide them into k or 5 subsets of data; each subset has 2000 examples[14].

No we assign one of these 5 subsets as a test set. Then, we fit a model on the training set of 8000 examples and then we'll evaluate the model on the 2000 example test set (holdout set). The performance metric is then recorded for this first iteration on the holdout set.

On the second iteration, where select another holdout test, and we perform same steps as before.

[14]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

Model optimization

Q. Bias and Variance in Machine learning?

Bias[15] 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[16] refers to an algorithm’s sensitivity to small fluctuations in the training data set.

[15] High bias is the result of the algorithm missing the relevant relations between features and target outputs
[16] 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:

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

Variance is a measure of the magnitude of that error.

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

We often find there is a trade-off between bias and variance. As one is reduced, the other is increased[17].

[17]This the matter of over-and-underfitting

Q. Bias and Variance tradeoff?

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

Total error is very high for very simple models and a very complex model, and then it bottoms out in the middle. 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[18]) as much as possible.

[18]Total error = (Bias + Variance) + Irreducible Error

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

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

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 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 θ_k. Then move θ_k in the direction of the slope.
- Update all θ_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**[19].

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[20]. To test for this you introduce a test data set, the data that you've held back.

[19] Typically you won't get decent results until convergence after many epochs
[20] 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.

Machine Learning Algorithms

Regression algorithms

Q. What is Regression?

Regression is a statistical process for estimating the relationship among variables, often 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 continuous target variable (output).

Linear regression attempts to model the relationship between two continuous variables – a scalar response (or dependent variable) and and one or more explanatory variables (or independent variables) – by fitting a linear equation to observed data.

The governing algorithm or equation for linear regression is also quite simple. It takes the form: $y = mx + b$ where y is the dependent variable, x is the independent variable, and b and m (i.e., slope) are coefficients dictating the equation.

e.g. a model that assumes a linear relationship between the input variables x and the single output variable y.

Q. What is logistic regression?

Logistic regression is another type of regression where the target variable is binary (zero or one, or true or false). It takes the form:

$$\frac{1}{1+e^{-(mx+b)}}$$

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

In other words, why do we need two regression algorithms? i.e., linear and logistic regression.

Because **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 will 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 which looks an S shape. This 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?

In other words, why do we need two regression algorithms? i.e., linear and logistic regression.

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.