### **<https://github.com/ShuaiW/data-science-question-answer/blob/master/README.md>**

### 

### **Cross Validation**

Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a validation set to evaluate it. For example, a k-fold cross validation divides the data into k folds (or partitions), trains on each k-1 fold, and evaluate on the remaining 1 fold. This results to k models/evaluations, which can be averaged to get a overall model performance.

### **Would adding more data address underfitting**

Underfitting happens when a model is not complex enough to learn well from the data. It is the problem of model rather than data size. So a potential way to address underfitting is to increase the model complexity (e.g., to add higher order coefficients for linear model, increase depth for tree-based methods, add more layers / number of neurons for neural networks etc.)

[back to top](https://github.com/ShuaiW/data-science-question-answer/blob/master/README.md#data-science-question-answer)

### **Activation Function**

For neural networks

* Non-linearity: ReLU is often used. Use Leaky ReLU (a small positive gradient for negative input, say, y = 0.01x when x < 0) to address dead ReLU issue
* Multi-class: softmax
* Binary: sigmoid
* Regression: linear

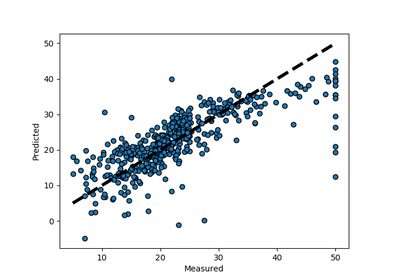
### **Generative vs discriminative**

* Discriminative algorithms model *p(y|x; w)*, that is, given the dataset and learned parameter, what is the probability of y belonging to a specific class. A discriminative algorithm doesn't care about how the data was generated, it simply categorizes a given example
* Generative algorithms try to model *p(x|y)*, that is, the distribution of features given that it belongs to a certain class. A generative algorithm models how the data was generated.

Given a training set, an algorithm like logistic regression or the perceptron algorithm (basically) tries to find a straight line—that is, a decision boundary—that separates the elephants and dogs. Then, to classify a new animal as either an elephant or a dog, it checks on which side of the decision boundary it falls, and makes its prediction accordingly.

Here’s a different approach. First, looking at elephants, we can build a model of what elephants look like. Then, looking at dogs, we can build a separate model of what dogs look like. Finally, to classify a new animal, we can match the new animal against the elephant model, and match it against the dog model, to see whether the new animal looks more like the elephants or more like the dogs we had seen in the training set.

* How to learn the parameter: minimize the cost function
* How to minimize cost function: gradient descent
* Regularization:
  + L1 (Lasso): can shrink certain coef to zero, thus performing feature selection
  + L2 (Ridge): shrink all coef with the same proportion; almost always outperforms L1
  + Elastic Net: combined L1 and L2 priors as regularizer
* Assumes linear relationship between features and the label
* Can add polynomial and interaction features to add non-linearity



### **Logistic regression**

* Generalized linear model (GLM) for binary classification problems
* Apply the sigmoid function to the output of linear models, squeezing the target to range [0, 1]
* Threshold to make prediction: usually if the output > .5, prediction 1; otherwise prediction 0
* A special case of softmax function, which deals with multi-class problems

### **Naive Bayes**

* Naive Bayes (NB) is a supervised learning algorithm based on applying [Bayes' theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem)
* It is called naive because it builds the naive assumption that each feature are independent of each other
* NB can make different assumptions (i.e., data distributions, such as Gaussian, Multinomial, Bernoulli)
* Despite the over-simplified assumptions, NB classifier works quite well in real-world applications, especially for text classification (e.g., spam filtering)
* NB can be extremely fast compared to more sophisticated methods