Probability for Machine Learning

7-Day Crash-Course

Jason Brownlee

MACHINE LEARNING MASTERY

Disclaimer

Probability for Machine Learning Crash Course

Edition: v1.8

Find the latest version of this guide online at: http://MachineLearningMastery.com

Contents

Before We Get Started...

Probability is a field of mathematics that is universally agreed to be the bedrock for machine learning. Although probability is a large field with many esoteric theories and findings, the nuts and bolts, tools and notations taken from the field are required for machine learning practitioners. With a solid foundation of what probability is, it is possible to focus on just the good or relevant parts. In this crash course, you will discover how you can get started and confidently understand and implement probabilistic methods used in machine learning with Python in seven days. Let's get started.

Who Is This Crash-Course For?

Before we get started, let's make sure you are in the right place. This course is for developers that may know some applied machine learning. Maybe you know how to work through a predictive modeling problem end-to-end, or at least most of the main steps, with popular tools. The lessons in this course do assume a few things about you, such as:

You need to know:

- You know your way around basic Python for programming.
- You may know some basic NumPy for array manipulation.
- You want to learn probability to deepen your understanding and application of machine learning.

You do NOT need to know:

- You do not need to be a math wiz!
- You do not need to be a machine learning expert!

This crash course will take you from a developer that knows a little machine learning to a developer who can navigate the basics of probabilistic methods. This crash course assumes you have a working Python3 SciPy environment with at least NumPy installed. If you need help with your environment, you can follow the step-by-step tutorial here:

- How to Setup a Python Environment for Machine Learning and Deep Learning

Crash-Course Overview

This crash course is broken down into seven lessons. You could complete one lesson per day (recommended) or complete all of the lessons in one day (hardcore). It really depends on the time you have available and your level of enthusiasm. Below is a list of the seven lessons that will get you started and productive with probability for machine learning in Python:

- **Lesson 01**: Probability and Machine Learning.
- **Lesson 02**: Three Types of Probability.
- **Lesson 03**: Probability Distributions.
- **Lesson 04**: Naive Bayes Classifier.
- **Lesson 05**: Entropy and Cross-Entropy.
- **Lesson 06**: Naive Classifiers.
- **Lesson 07**: Probability Scores.

Each lesson could take you 60 seconds or up to 30 minutes. Take your time and complete the lessons at your own pace. Ask questions and even share your results online. The lessons expect you to go off and find out how to do things. I will give you hints, but part of the point of each lesson is to force you to learn where to go to look for help on and about the statistical methods and the NumPy API and the best-of-breed tools in Python. (Hint: I have all of the answers directly on this blog; use the search box.) Share your results online, I'll cheer you on!

Hang in there, don't give up!

Lesson 01: Probability and Machine Learning

In this lesson, you will discover why machine learning practitioners should study probability to improve their skills and capabilities. Probability is a field of mathematics that quantifies uncertainty. Machine learning is about developing predictive modeling from uncertain data. Uncertainty means working with imperfect or incomplete information. Uncertainty is fundamental to the field of machine learning, yet it is one of the aspects that causes the most difficulty for beginners, especially those coming from a developer background. There are three main sources of uncertainty in machine learning; they are:

- **Noise in observations**, e.g. measurement errors and random noise.
- **Incomplete coverage of the domain**, e.g. you can never observe all data.
- **Imperfect model of the problem**, e.g. all models have errors, some are useful.

Uncertainty in applied machine learning is managed using probability.

- Probability and statistics help us to understand and quantify the expected value and variability of variables in our observations from the domain.
- Probability helps to understand and quantify the expected distribution and density of observations in the domain.
- Probability helps to understand and quantify the expected capability and variance in performance of our predictive models when applied to new data.

This is the bedrock of machine learning. On top of that, we may need models to predict a probability, we may use probability to develop predictive models (e.g. Naive Bayes), and we may use probabilistic frameworks to train predictive models (e.g. maximum likelihood estimation).

Your Task

For this lesson, you must list three reasons why you want to learn probability in the context of machine learning. These may be related to some of the reasons above, or they may be your own personal motivations.

Next

In the next lesson, you will discover the three different types of probability and how to calculate them.

Lesson 02: Three Types of Probability

In this lesson, you will discover a gentle introduction to joint, marginal, and conditional probability between random variables. Probability quantifies the likelihood of an event. Specifically, it quantifies how likely a specific outcome is for a random variable, such as the flip of a coin, the roll of a die, or drawing a playing card from a deck. We can discuss the probability of just two events: the probability of event A for variable X and event B for variable Y, which in shorthand is $X = A$ and $Y = B$, and that the two variables are related or dependent in some way. As such, there are three main types of probability we might want to consider.

Joint Probability

We may be interested in the probability of two simultaneous events, like the outcomes of two different random variables. For example, the joint probability of event A and event B is written formally as:

$$P(A \cap B) \tag{1}$$

The joint probability for events A and B is calculated as the probability of event A given event B multiplied by the probability of event B. This can be stated formally as follows:

$$P(A \cap B) = P(A|B) \times P(B) \tag{2}$$

Marginal Probability

We may be interested in the probability of an event for one random variable, irrespective of the outcome of another random variable. There is no special notation for marginal probability; it is just the sum or union over all the probabilities of all events for the second variable for a given fixed event for the first variable.

$$P(X = A) = \sum_{i=1}^{n} P(X = A, Y = y_i) \tag{3}$$

Conditional Probability

We may be interested in the probability of an event given the occurrence of another event. For example, the conditional probability of event A given event B is written formally as:

$$P(A|B) \tag{4}$$

The conditional probability for events A given event B can be calculated using the joint probability of the events as follows:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{5}$$

Your Task

For this lesson, you must practice calculating joint, marginal, and conditional probabilities. For example, if a family has two children and the oldest is a boy, what is the probability of this family having two sons? This is called the *Boy or Girl Problem* and is one of many common toy problems for practicing probability.

Next

In the next lesson, you will discover probability distributions for random variables.

Lesson 03: Probability Distributions

In this lesson, you will discover a gentle introduction to probability distributions. In probability, a random variable can take on one of many possible values, e.g. events from the state space. A specific value or set of values for a random variable can be assigned a probability. There are two main classes of random variables.

- **Discrete Random Variable**. Values are drawn from a finite set of states.
- **Continuous Random Variable**. Values are drawn from a range of real-valued numerical values.

A discrete random variable has a finite set of states; for example, the colors of a car. A continuous random variable has a range of numerical values; for example, the height of humans. A probability distribution is a summary of probabilities for the values of a random variable.

Discrete Probability Distributions

A discrete probability distribution summarizes the probabilities for a discrete random variable. Some examples of well-known discrete probability distributions include:

- Poisson distribution.
- Bernoulli and binomial distributions.
- Multinoulli and multinomial distributions.

Continuous Probability Distributions

A continuous probability distribution summarizes the probability for a continuous random variable. Some examples of well-known continuous probability distributions include:

- Normal or Gaussian distribution.
- Exponential distribution.
- Pareto distribution.

Randomly Sample Gaussian Distribution

We can define a distribution with a mean of 50 and a standard deviation of 5 and sample random numbers from this distribution. We can achieve this using the `normal()` NumPy function. The example below samples and prints 10 numbers from this distribution.

```
# sample a normal distribution
from numpy.random import normal
# define the distribution
mu = 50
sigma = 5
n = 10
# generate the sample
sample = normal(mu, sigma, n)
print(sample)
```

Listing 1: Example of sampling from a Gaussian distribution.

Running the example prints 10 numbers randomly sampled from the defined normal distribution.

Your Task

For this lesson, you must develop an example to sample from a different continuous or discrete probability distribution function. For a bonus, you can plot the values on the x-axis and the probability on the y-axis for a given distribution to show the density of your chosen probability distribution function.

Next

In the next lesson, you will discover the Naive Bayes classifier.

Lesson 04: Naive Bayes Classifier

In this lesson, you will discover the Naive Bayes algorithm for classification predictive modeling. In machine learning, we are often interested in a predictive modeling problem where we want to predict a class label for a given observation. One approach to solving this problem is to develop a probabilistic model. From a probabilistic perspective, we are interested in estimating the conditional probability of the class label given the observation, or the probability of class y given input data X.

$$P(y|X) \tag{6}$$

Bayes Theorem provides an alternate and principled way for calculating the conditional probability using the reverse of the desired conditional probability, which is often simpler to calculate. The simple form of the calculation for Bayes Theorem is as follows:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{7}$$

Where the probability that we are interested in calculating $P(A|B)$ is called the posterior probability and the marginal probability of the event $P(A)$ is called the prior. The direct application of Bayes Theorem for classification becomes intractable, especially as the number of variables or features (n) increases. Instead, we can simplify the calculation and assume that each input variable is independent. Although dramatic, this simpler calculation often gives very good performance, even when the input variables are highly dependent. We can implement this from scratch by assuming a probability distribution for each separate input variable and calculating the probability of each specific input value belonging to each class and multiply the results together to give a score used to select the most likely class.

$$P(y_i|x_1, x_2, \cdots, x_n) = P(x_1|y_1) \times P(x_2|y_1) \times \cdots \times P(x_n|y_1) \times P(y_1) \tag{8}$$

The scikit-learn library provides an efficient implementation of the algorithm if we assume a Gaussian distribution for each input variable. To use a scikit-learn Naive Bayes model, first the model is defined, then it is fit on the training dataset. Once fit, probabilities can be predicted via the `predict_proba()` function and class labels can be predicted directly via the `predict()` function. The complete example of fitting a Gaussian Naive Bayes model (`GaussianNB`) to a test dataset is listed below.

```
# example of gaussian naive bayes
from sklearn.datasets import make_blobs
from sklearn.naive_bayes import GaussianNB
# generate 2d classification dataset
X, y = make_blobs(n_samples=100, centers=2, n_features=2, random_state=1)
# define the model
```

```
model = GaussianNB()
# fit the model
model.fit(X, y)
# select a single sample
Xsample, ysample = [X[0]], y[0]
# make a probabilistic prediction
yhat_prob = model.predict_proba(Xsample)
print('Predicted Probabilities: ', yhat_prob)
# make a classification prediction
yhat_class = model.predict(Xsample)
print('Predicted Class: ', yhat_class)
print('Truth: y=%d' % ysample)
```

Listing 2: Example of a Gaussian Naive Bayes model.

Running the example fits the model on the training dataset, then makes predictions for the same first example that we used in the prior example.

Your Task

For this lesson, you must run the example and report the result. For a bonus, try the algorithm on a real classification dataset, such as the popular toy classification problem of classifying iris flower species based on flower measurements.

Next

In the next lesson, you will discover entropy and the cross-entropy scores.

Lesson 05: Entropy and Cross-Entropy

In this lesson, you will discover cross-entropy for machine learning. Information theory is a field of study concerned with quantifying information for communication. The intuition behind quantifying information is the idea of measuring how much surprise there is in an event. Those events that are rare (low probability) are more surprising and therefore have more information than those events that are common (high probability).

- **Low Probability Event**: High Information (*surprising*).
- **High Probability Event**: Low Information (*unsurprising*).

We can calculate the amount of information there is in an event using the probability of the event.

$$\text{Information}(x) = -\log(P(x)) \tag{9}$$

We can also quantify how much information there is in a random variable. This is called entropy and summarizes the amount of information required on average to represent events. Entropy can be calculated for a random variable X with K discrete states as follows:

$$Entropy(X) = -\sum_{i=1}^{K} P(k_i) \times \log(P(k_i)) \tag{10}$$

Cross-entropy is a measure of the difference between two probability distributions for a given random variable or set of events. It is widely used as a loss function when optimizing classification models. It builds upon the idea of entropy and calculates the average number of bits required to represent or transmit an event from one distribution compared to the other distribution.

$$\text{CrossEntropy}(P, Q) = -\sum_{x \in X} P(x) \times \log(Q(x)) \tag{11}$$

We can make the calculation of cross-entropy concrete with a small example. Consider a random variable with three events as different colors. We may have two different probability distributions for this variable. We can calculate the cross-entropy between these two distributions. The complete example is listed below.

```
# example of calculating cross-entropy
from math import log2

# calculate cross-entropy
def cross_entropy(p, q):
```

```
  return -sum([p[i]*log2(q[i]) for i in range(len(p))])

# define data
p = [0.10, 0.40, 0.50]
q = [0.80, 0.15, 0.05]
# calculate cross-entropy H(P, Q)
ce_pq = cross_entropy(p, q)
print('H(P, Q): %.3f bits' % ce_pq)
# calculate cross-entropy H(Q, P)
ce_qp = cross_entropy(q, p)
print('H(Q, P): %.3f bits' % ce_qp)
```

Listing 3: Example of calculating cross-entropy from scratch.

Running the example first calculates the cross-entropy of Q from P, then P from Q.

Your Task

For this lesson, you must run the example and describe the results and what they mean. For example, is the calculation of cross-entropy symmetrical?

Next

In the next lesson, you will discover how to develop and evaluate a naive classifier model.

Lesson 06: Naive Classifiers

In this lesson, you will discover how to develop and evaluate naive classification strategies for machine learning. Classification predictive modeling problems involve predicting a class label given an input to the model. Given a classification model, how do you know if the model has skill or not? This is a common question on every classification predictive modeling project. The answer is to compare the results of a given classifier model to a baseline or naive classifier model.

Consider a simple two-class classification problem where the number of observations is not equal for each class (e.g. it is imbalanced) with 25 examples for class-0 and 75 examples for class-1. This problem can be used to consider different naive classifier models. For example, consider a model that randomly predicts class-0 or class-1 with equal probability. How would it perform? We can calculate the expected performance using a simple probability model.

$$P(\text{yhat} = \text{y}) = P(\text{yhat} = 0) \times P(\text{y} = 0) + P(\text{yhat} = 1) \times P(\text{y} = 1) \tag{12}$$

We can plug in the occurrence of each class (0.25 and 0.75) and the predicted probability for each class (0.5 and 0.5) and estimate the performance of the model.

$$\begin{aligned} P(\text{yhat} = \text{y}) &= 0.5 \times 0.25 + 0.5 \times 0.75 \\ P(\text{yhat} = \text{y}) &= 0.5 \end{aligned} \tag{13}$$

It turns out that this classifier is pretty poor. Now, what if we consider predicting the majority class (class 1) every time? Again, we can plug in the predicted probabilities (0.0 and 1.0) and estimate the performance of the model.

$$\begin{aligned} P(\text{yhat} = \text{y}) &= 0.0 \times 0.25 + 1.0 \times 0.75 \\ P(\text{yhat} = \text{y}) &= 0.75 \end{aligned} \tag{14}$$

It turns out that this simple change results in a better naive classification model, and is perhaps the best naive classifier to use when classes are imbalanced. The scikit-learn machine learning library provides an implementation of the majority class naive classification algorithm called the `DummyClassifier` that you can use on your next classification predictive modeling project. The complete example is listed below.

```
# example of the majority class naive classifier in scikit-learn
from numpy import asarray
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score
# define dataset
X = asarray([0 for _ in range(100)])
class0 = [0 for _ in range(25)]
class1 = [1 for _ in range(75)]
y = asarray(class0 + class1)
```

```
# reshape data for sklearn
X = X.reshape((len(X), 1))
# define model
model = DummyClassifier(strategy='most_frequent')
# fit model
model.fit(X, y)
# make predictions
yhat = model.predict(X)
# calculate accuracy
accuracy = accuracy_score(y, yhat)
print('Accuracy: %.3f' % accuracy)
```

Listing 4: Example of a majority class classifier.

Run the example and review the confidence interval on the estimated accuracy.

Your Task

For this lesson, you must list two methods for calculating the effect size in applied machine learning and when they might be useful. As a hint, consider one for the relationship between variables and one for the difference between samples.

Next

In the next lesson, you will discover nonparametric statistical methods.

Lesson 07: Probability Scores

In this lesson, you will discover two scoring methods that you can use to evaluate the predicted probabilities on your classification predictive modeling problem. Predicting probabilities instead of class labels for a classification problem can provide additional nuance and uncertainty for the predictions. The added nuance allows more sophisticated metrics to be used to interpret and evaluate the predicted probabilities. Let's take a closer look at the two popular scoring methods for evaluating predicted probabilities.

Log Loss Score

Logistic loss, or log loss for short, calculates the log likelihood between the predicted probabilities and the observed probabilities. Although developed for training binary classification models like logistic regression, it can be used to evaluate multiclass problems and is functionally equivalent to calculating the cross-entropy derived from information theory. A model with perfect skill has a log loss score of 0.0. The log loss can be implemented in Python using the `log_loss()` function in scikit-learn. For example:

```
# example of log loss
from numpy import asarray
from sklearn.metrics import log_loss
# define data
y_true = [1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
y_pred = [0.8, 0.9, 0.9, 0.6, 0.8, 0.1, 0.4, 0.2, 0.1, 0.3]
# define data as expected, e.g. probability for each event {0, 1}
y_true = asarray([[v, 1-v] for v in y_true])
y_pred = asarray([[v, 1-v] for v in y_pred])
# calculate log loss
loss = log_loss(y_true, y_pred)
print(loss)
```

Listing 5: Example of calculating log-loss.

Brier Score

The Brier score, named for Glenn Brier, calculates the mean squared error between predicted probabilities and the expected values. The score summarizes the magnitude of the error in the probability forecasts. The error score is always between 0.0 and 1.0, where a model with perfect skill has a score of 0.0. The Brier score can be calculated in Python using the `brier_score_loss()` function in scikit-learn. For example:

```
# example of brier loss
from sklearn.metrics import brier_score_loss
# define data
y_true = [1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
y_pred = [0.8, 0.9, 0.9, 0.6, 0.8, 0.1, 0.4, 0.2, 0.1, 0.3]
# calculate brier score
score = brier_score_loss(y_true, y_pred, pos_label=1)
print(score)
```

Listing 6: Example of calculating Brier Score.

Your Task

For this lesson, you must run each example and report the results. As a bonus, change the mock predictions to make them better or worse and compare the resulting scores. This was the final lesson in the mini-course.

Final Word Before You Go...

You made it. Well done! Take a moment and look back at how far you have come. You discovered:

- The importance of probability in applied machine learning.
- The three main types of probability and how to calculate them.
- Probability distributions for random variables and how to draw random samples from them.
- How Bayes theorem can be used to calculate conditional probability and how it can be used in a classification model.
- How to calculate information, entropy, and cross-entropy scores and what they mean.
- How to develop and evaluate the expected performance for naive classification models.
- How to evaluate the skill of a model that predicts probability values for a classification problem.

This is just the beginning of your journey with probability for machine learning. Keep practicing and developing your skills. Take the next step and check out my book on *Probability Methods for Machine Learning*.

How Did You Go With The Crash-Course?

Did you enjoy this crash-course?
Do you have any questions or sticking points?

Let me know, send me an email at: **jason@MachineLearningMastery.com**

Take the Next Step

Looking for more help with Probability for Machine Learning?

Grab my new book:
Probability for Machine Learning
https://machinelearningmastery.com/probability-for-machine-learning/