4.1.2: Types of learning

Topic Progress: (https://gurus.pyimagesearch.com/topic/what-is-image-classification/)

(https://gurus.pyimagesearch.com/topic/types-of-learning/)

← Back to Lesson (https://gurus.pyimagesearch.com/lessons/a-high-level-overview-of-image-classification/)

In our **previous lesson (https://gurus.pyimagesearch.com/topic/what-is-image-classification/)** we reviewed the concept of machine learning and image classification, allowing us to assign a label to an image from a pre-defined set of categories using specialized algorithms. We also explored some of the challenges associated with image classification and why it's such a hard task that has yet to be "solved".

However, much of what we discussed in the previous lesson is known as **supervised learning**, where our input data consists of the *image data **and** the labels associated with each image*, allowing us to train/teach our classifier what each category looks like. We act as teachers to our algorithms, let them make predictions as to what they think the image contains, and then correct them when a mistake is made.

But what if we only had the raw images themselves without their associated categories? Could we still classify the contents of the image without the pre-defined labels even though there is no training step? What about if we only had a small subset of our image data labeled — could we apply machine learning and still classify and categorize the image?

It turns out that we can — we just need a specialized set of machine learning algorithms in each particular case.

Objectives

In this lesson, we will be reviewing the three primary types of machine learning:

- Supervised learning
- Unsupervised learning
- Semi-supervised learning

Supervised learning

Imagine this: you've just graduated from college with your Bachelors of Science in Computer Science. You're young. Broke. And looking for a job in the field — perhaps you even feel a bit lost in your job search.

BAM! Before you know it, a Google recruiter finds you on LinkedIn and offers you a position working on their GMail software.

Are you going to take it? Most likely.

A few weeks later, you pull up to Google's spectacular campus in Mountain View, California, overwhelmed by the breathtaking landscape, the fleet of Tesla's in the parking lot, and the almost never-ending rows of gourmet food in the cafeteria.

You finally sit down at your desk in a wide-open workspace among hundreds of other employees — and then you find out your role in the company. You've been hired to create a piece of software to *automatically classify* email as *spam* or *not-spam*.

How in the world are you going to accomplish this goal?

Well, think back to yesterday's lesson on **image classification (https://gurus.pyimagesearch.com/topic/what-is-image-classification/)**. Would a rule-based approach work? Could you write a series of "if/else" statements looking for certain words and then determining if an email is spam based on these rules?

That might work… to a degree. But this approach would also be easily defeated and be near impossible to maintain.

Instead, what you *really need* is machine learning.

You need a *training set* consisting of the emails themselves along with their *labels*, in this case: *spam* or *not-spam*.

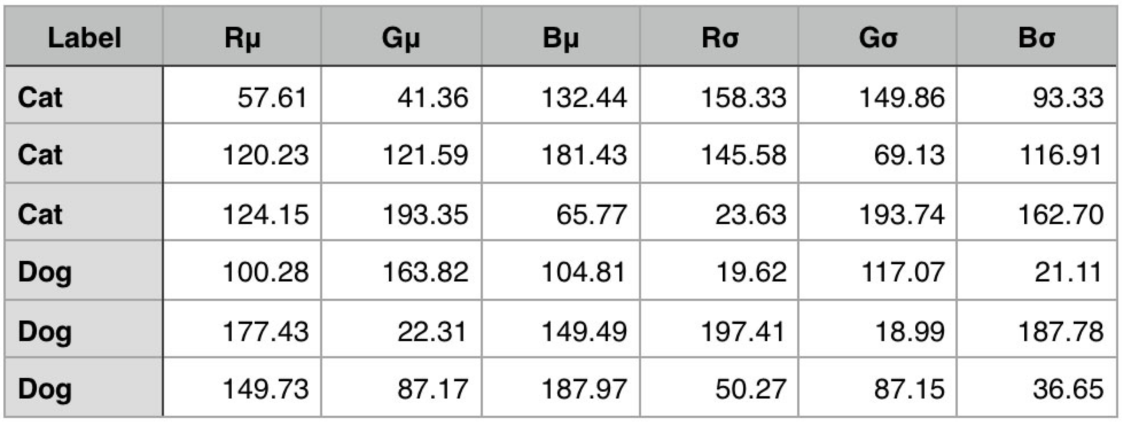
Given this data, you can analyze the text (i.e. the distribution of words) of the email and utilize the spam/not-spam labels to teach a machine learning classifier what words occur in a spam email and which do not — all without having to manually create long and complicated series of "if/else" statements.

This example of creating a spam filter system is an example of **supervised learning**.

Supervised learning is arguably the most well known and studied type of machine learning. Given our training data, a model (or "classifier") is created through a training process where predictions are made on the input data and then corrected when the predictions are wrong. This training process continues until the model achieves some desired stopping criterion, such as a low error rate or a maximum number of training iterations. Common supervised learning algorithms include Logistic Regression, Support Vector Machines, and Random Forests.

In the context of **image classification**, we assume that our image dataset consists of the images themselves along with a corresponding *label* that we can use to teach our machine learning classifier what each category looks like. If our classifier makes an incorrect prediction, we can then apply methods to correct its mistake.

My favorite way to visualize **supervised learning** is probably looking at a spreadsheet:



Label	Rμ	Gμ	Bμ	Rσ	Gσ	Bσ
Cat	57.61	41.36	132.44	158.33	149.86	93.33
Cat	120.23	121.59	181.43	145.58	69.13	116.91
Cat	124.15	193.35	65.77	23.63	193.74	162.70
Dog	100.28	163.82	104.81	19.62	117.07	21.11
Dog	177.43	22.31	149.49	197.41	18.99	187.78
Dog	149.73	87.17	187.97	50.27	87.15	36.65

(https://gurus.pyimagesearch.com/wp-content/uploads/2015/06/features_table_supervised_learning.jpg)

FIGURE 1: WHEN PERFORMING SUPERVISED LEARNING, OUR DATA MUST CONTAIN ***BOTH*** THE LABEL/CATEGORY OF THE IMAGE AND THE ASSOCIATED IMAGE DATA OR FEATURE VECTORS.

The first column of our spreadsheet is the label associated with a particular image. The remaining six columns correspond to our feature vector — in this case the mean and standard deviation of each RGB color channel, as we discussed **here (https://gurus.pyimagesearch.com/lessons/color-channel-statistics/)**.

Our supervised learning algorithm will make predictions on each of these feature vectors, and if it makes an incorrect prediction, we'll correct it by telling it what the correct label actually is. This process will then continue until a desired stopping criterion has been met, such as accuracy, number of iterations of the learning process, or simply an arbitrary amount of time.

Unsupervised learning

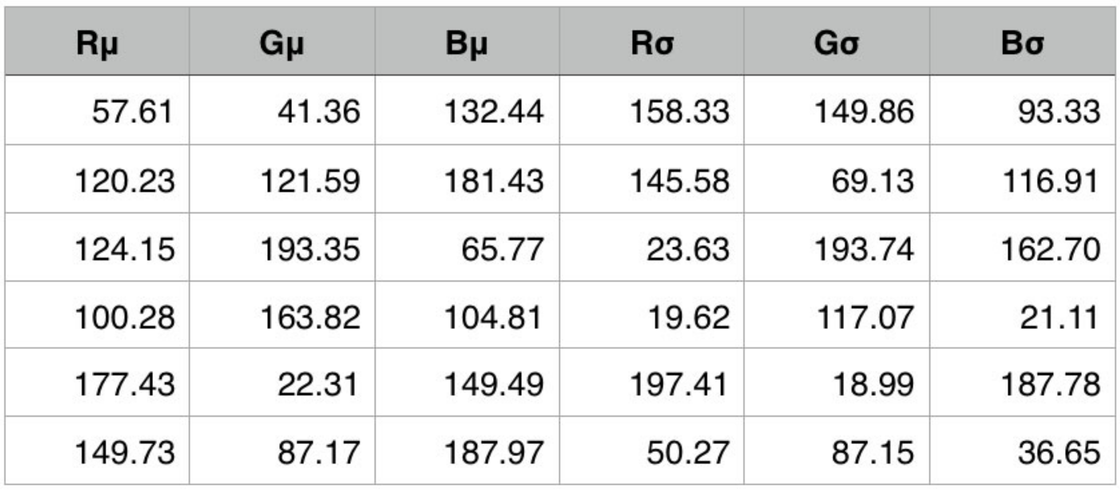
In contrast to supervised learning, **unsupervised learning** has no labels associated with the input data, and thus we cannot correct our model if it makes an incorrect prediction. Thus, most unsupervised learning methods are focused on deducing structure present in the input data.

Remember back to our lesson on **extracting color histograms (https://gurus.pyimagesearch.com/lessons/color-histograms/)**, where we clustered our vacation photos into two separate groups using the k-means algorithm?

In that example, we only had the images themselves — *we had no labels associated with these images to help us guess which category each respective image belongs to.*

However, by applying the k-means algorithm, we were able to automatically cluster our vacation photo images into their respective groups based on only the color histograms extracted from the images. Using these color histograms and the k-means algorithm, we found two natural "structures" (i.e. clusters) in the data.

Going back to our spreadsheet example, we can think of unsupervised learning data as removing the "label" column of the spreadsheet, leaving us with only the data itself:



Rμ	Gμ	Bμ	Rσ	Gσ	Bσ
57.61	41.36	132.44	158.33	149.86	93.33
120.23	121.59	181.43	145.58	69.13	116.91
124.15	193.35	65.77	23.63	193.74	162.70
100.28	163.82	104.81	19.62	117.07	21.11
177.43	22.31	149.49	197.41	18.99	187.78
149.73	87.17	187.97	50.27	87.15	36.65

(https://gurus.pyimagesearch.com/wp-content/uploads/2015/06/features_table_unsupervised_learning.jpg)

FIGURE 2: IN UNSUPERVISED LEARNING WE DO NOT HAVE THE IMAGE LABELS — ONLY THE IMAGE DATA OR ASSOCIATED FEATURE VECTORS.

Other popular unsupervised learning algorithms include the Gaussian processes, the *A Priori* for association rule learning, and Principal Component Analysis (PCA) for dimensionality reduction, which is especially useful for the task of face identification in the Eigenfaces algorithm (https://gurus.pyimagesearch.com/lessons/the-eigenfaces-algorithm/).

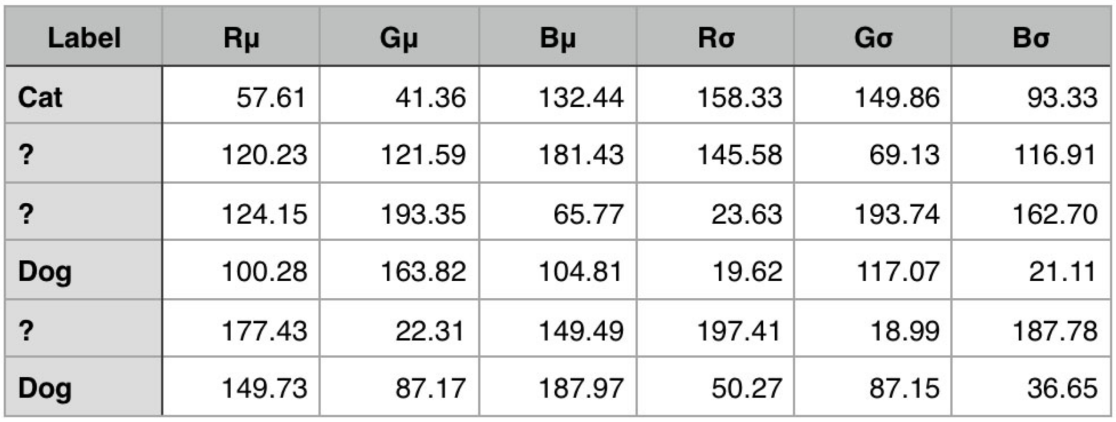
In this course, we'll be making heavy use of the k-means algorithm, especially when we start discussing the bag-of-visual-worlds model (https://gurus.pyimagesearch.com/lessons/what-is-content-based-image-retrieval/) for Image Classification and Content-Based Image Retrieval, so this is not the last time you will see unsupervised learning.

Semi-supervised learning

So what happens if we only have *some* of the labels associated with our data and *no labels* for the other? Is there a way that we can apply some hybrid of supervised and unsupervised learning and still be able to classify each of our data points?

It turns out the answer is ***yes*** — we just need to apply semi-supervised learning.

Going back to our spreadsheet example, let's say that we only had labels for a small fraction of our input data:



Label	Rμ	Gμ	Bμ	Rσ	Gσ	Bσ
Cat	57.61	41.36	132.44	158.33	149.86	93.33
?	120.23	121.59	181.43	145.58	69.13	116.91
?	124.15	193.35	65.77	23.63	193.74	162.70
Dog	100.28	163.82	104.81	19.62	117.07	21.11
?	177.43	22.31	149.49	197.41	18.99	187.78
Dog	149.73	87.17	187.97	50.27	87.15	36.65

(https://gurus.pyimagesearch.com/wp-content/uploads/2015/06/features_table_semisupervised_learning.jpg)

FIGURE 3: WHEN PERFORMING SEMI-SUPERVISED LEARNING, WE ONLY HAVE THE LABELS FOR A SUBSET OF THE IMAGES/FEATURE VECTORS AND MUST TRY TO LABEL THE OTHER DATA POINTS TO UTILIZE THEM AS EXTRA TRAINING DATA.

Our semi-supervised learning algorithm would take the known pieces of data, analyze them, and then try to label each of the unlabeled data points for use as extra training data. This process can then repeat for many iterations as the semi-supervised algorithm learns the "structure" of the data to make more accurate predictions and generate more reliable training data. The overall goal here is to generate more training data, which the algorithm can use to make itself "smarter".

Semi-supervised learning is especially useful in computer vision where it is often time consuming, tedious, and expensive (at least in terms of man hours) to label each and every single image in our training set.

In cases where we simply do not have the time or resources to label each individual image, we can label only a tiny fraction of our data and utilize semi-supervised learning to label and classify the rest of the images.

Semi-supervised learning algorithms often trade smaller labeled input datasets for some tolerable reduction in classification accuracy. Normally, the more accurately-labeled training data a supervised learning algorithm has, the more accurate predictions it can make. As the amount of training data goes down, accuracy inevitably suffers. Semi-supervised learning takes this into account and attempts to keep classification accuracy within tolerable limits while dramatically reducing the amount of training data required to build the model — the end result is an accurate classifier (but normally not as accurate as a supervised classifier) with less effort and training data.

Summary

Feedback

In this lesson, we reviewed the three primary types of learning: *supervised learning*, *unsupervised learning*, and *semi-supervised learning*.

In *supervised learning*, we have both the *image data* (in either raw image format or the extracted feature vectors) along with the *label/category* associated with each image, so we can "teach" our algorithm what each image category looks like.

On the other end of the spectrum we have *unsupervised learning*, where all we have is the image data itself — we have no associated label or category that we can use to teach our algorithm to make an accurate predictions or classifications. Because of this, we mainly use unsupervised learning algorithms to uncover the underlying structure of our data, such as natural forming clusters or groupings based on the extracted feature vectors (just like we did with **color histograms (https://gurus.pyimagesearch.com/lessons/color-histograms/)**).

Finally, *semi-supervised learning* attempts to be a middle-ground between the two. Here, we have a small sub-set of our image data labeled, and we utilize this labeled data to generate more training data from the unlabeled data.

This course will primarily focus on supervised learning methods; however, we'll be using unsupervised learning algorithms such as k-means and PCA quite regularly, especially once we start constructing a bag-of-visual-words model for Image Classification and Content-Based Image Retrieval.

Quizzes		Status
1	Types of Learning Quiz (https://gurus.pyimagesearch.com/quizzes/types-of-learning-quiz/)	

← Previous Topic (https://gurus.pyimagesearch.com/topic/what-is-image-classification/)

Feedback

Course Progress

Ready to continue the course?

Click the button below to **continue your journey to computer vision guru**.

I'm ready, let's go! (/pyimagesearch-gurus-course/)

Resources & Links

- PyImageSearch Gurus Community (https://community.pyimagesearch.com/)
- PyImageSearch Virtual Machine (https://gurus.pyimagesearch.com/pyimagesearch-virtual-machine/)
- Setting up your own Python + OpenCV environment (https://gurus.pyimagesearch.com/setting-up-your-python-opencv-development-environment/)
- Course Syllabus & Content Release Schedule (https://gurus.pyimagesearch.com/course-syllabus-content-release-schedule/)
- Member Perks & Discounts (https://gurus.pyimagesearch.com/pyimagesearch-gurus-discounts-perks/)
- Your Achievements (https://gurus.pyimagesearch.com/achievements/)
- Official OpenCV documentation (http://docs.opencv.org/index.html)

Your Account

- Account Info (https://gurus.pyimagesearch.com/account/)
- Support (https://gurus.pyimagesearch.com/contact/)
- Logout (https://gurus.pyimagesearch.com/wp-login.php?action=logout&redirect_to=https%3A%2F%2Fgurus.pyimagesearch.com%2F&_wpnonce=5736b21cae)

Search

Feedback