accusoft®

Machine Learning and the Death of Old-School Classifiers

John Reynolds
Software Engineer II

An initial query for readers out there. What is this text below? Did my cat walk across the keyboard as I was typing this blog article? Is this simply modem line noise?

Many, I'm sure, will recognize this text block as a regex, specifically a regex that validates whether or not a particular block of text is a valid email. For its part, it does a fantastic job, but clearly a non-trivial amount of work was put into the construction of this regex and many other regexes like it.

```
(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\
x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-
]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}
(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\
x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])
```

Whoever wrote it had a clear understanding of RFC-5322 and the intricacies therein. If I were to write my own email regex validator, it would likely be far too restrictive and there are a host of potential problems and pitfalls that I would probably fall into. There is a great deal of domain-specific knowledge that goes into developing these, and many developers can run afoul of problems introduced by the potentially high complexity involved.

Also, this regex is an example of a strict classifier. It represents a boolean way of separating whether or not a particular string of text is in one class or another, specifically the set of: {is_a_valid_email_address, is_not_a_valid_email_address}.

Strictly binary classifiers (a hard true or false) are very useful for validation tasks, but what I'm interested in investigating are the changes to "fuzzier" classifiers. Those classifiers that seek to ask, under ambiguous circumstances: "How likely is this text to be an email? How likely is this picture to be a dog? Where in this image a barcode?" In cases like this, strict classifiers are not the tool we want to work with.

The question I seek to answer is: How has the industry previously solved these questions, and how is this changing?

How the Industry Used to Do Things

Most of the products I work on have to do with, broadly speaking, image recognition and detection. Let's begin there. I'll start with an example that's near and dear to my heart - barcodes.

I'm sure most people out there have seen these before. QR codes are a two-dimensional barcode that were invented in 1994 by Japanese auto manufacturers. They've since exploded in popularity and you'll see them all over the place: soda cans, fliers, magazine articles, etc. You'll scan them with your cell phone, and an app might take you to a website, or show some metadata for the QR code.

What our API needs to do is find instances of QR codes in an image whether it be a fax, scanned document, or photo, and it needs to do it quickly and accurately. Now, as a software developer, this has represented some particular challenges over the years. How might we identify areas of an image that contain QR codes?

The biggest and most obvious feature we can see are concentric rings of the three position patterns, so let's focus on these and do some free thinking on how to find them. We might do some connected component analysis, or perhaps do some run-length calculations to see if we can find instances of the 1:1:3:1:1 ratios of the pattern.

We might also decided to run an edge-detection filter on the image to find the lines of the pattern. If we look at enough images of QR codes, we'd note that the ratio of white to black blocks tends toward 1:1, and we could use that as a heuristic to guide our generalized search of the image.

All of these methods have varying degrees of difficulty in implementation and high complexity. These approaches, and those that we have used in our software, have taken years to develop and are highly specialized. I, myself, have been working on them for over 10 years now. They've been written with an intimate understanding of the various specifications of the barcodes we read.

Now let's throw another wrinkle in things. Let's imagine you've implemented the algorithms above based on what the QR code specification says, and now you run it on data from actual customers.

When dealing with the real world, your expectations can be thrown awry. You'll often go in expecting sane inputs, but what you can end up with are blurry perspective warped noisy messes. Now you have to bring even more advanced concepts to the party: things like despeckling algorithms, 2D homographies, etc.

This is how the software industry has largely done things in the past over many different contexts (eg. facial recognition technology). It sounds pretty grim, that to get that long tail of hard-to-read images, you need exceedingly intricate and complex domain-specific algorithms.

Thankfully, this has been changing lately. Now, companies have begun to leverage their large amounts of data, often supplied by customers or synthetically generated, to create general-purpose algorithms to solve their needs.

Enter: Machine Learning

You've heard of it, I've heard of it… even my wonderful, yet computer averse, mother has heard of it.

[fireworks] **Machine Learning** [fireworks]

Is it the real deal? Is it new tech-fad buzzword? To be honest, it's a lot of column A and a little of column B. There's a great deal of potential in the concepts of machine learning. In this particular case, I'll be discussing specifically deep learning and the way that the industry's been changing in leaps and bounds from these new technologies. Yet, we must also discuss the fact that companies have been rushing into the field to not be left behind.

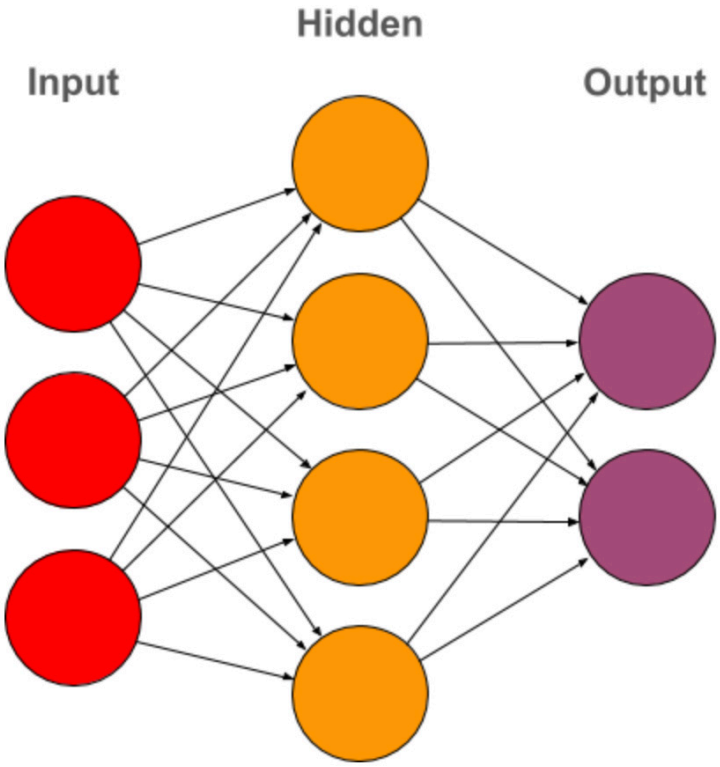
source: ***XKCD***

Hopefully, I'll be able to dispel some myths about the field and illustrate how we here at Accusoft have been navigating these changes, and how your software might also benefit from the field.

Background: Neural Networks

The backbone of most machine learning software programs is likely to be neural networks. If I were to try and describe, in lay fashion, what the process of training neural networks is like, I might envision a scenario of hot and cold as such:

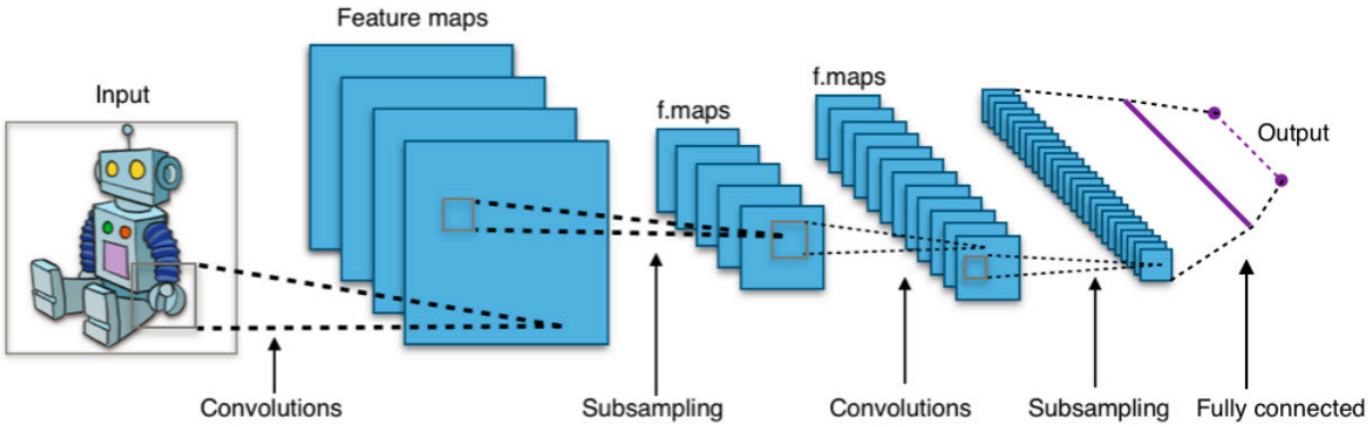
Imagine Bob is inside a room and is looking for a device that is beeping. He takes into account observations of the room (sight, sound, etc) and makes an educated guess as to which direction the device is in, and makes a step in that direction. Alice knows where the device is. If he steps in the right direction, she lets him know he's getting hotter; a step in the wrong direction lets him know he's getting colder. As Bob navigates the room, he learns what to make of his observations and how to react to different circumstances.



The reality is, naturally, more complicated than this. It involves calculus, linear algebra, and patience. Broadly speaking though, neural networks function by taking in your inputs, in the form of scalar values (numbers), and sends them through a network of non-linear activation functions to finally arrive as the numeric outputs you desire. Of note, is that the non-linear nature of the network is **really important**. Most of the problems that we seek to solve in the business world are not problems that are linearly separable and thankfully the non-linear nature of neural networks allows us to solve all of the wacky patterns that can exist in our data.

CNNs: An Imaging Sonic Screwdriver

Because we're dealing with image data, I'd like to introduce one of the most powerful tools for image recognition that has been created in the field of machine learning: the convolutional neural network. Convolutional neural networks (or CNNs for short) take a cue from biology and operate similarly to how the human eye works (more or less). In our eyes, we have over 100 million photoreceptor cells that send signals through pathways to neurons. Each neuron operates only on a local area of data (called the receptive field) and passes signals down to the brain.



source: ***Wikipedia***

Similarly, CNNs take in an image as an input and process each pixel by taking into account its immediate neighbors (analogous to how photoshop filters work). Another set of "filters" will then be applied to this result image, and this process repeats itself many layers deep.

What's amazing about this, is that when the network is trained, each bank of filters begins to automatically learn features of the original images. The first layer of filters learned tend to be simple features, like those found in the neurons of our eyes. E.g. What edges are there? Other features that may be learned in the first layer are basic assumptions about color. E.g. Is this a largely dark region, a red region, etc? Subsequent layers then begin to learn how to put together these low-level features. E.g. Is this area a curve? Is this a corner?

The further you get into the network, higher level features begin to emerge from the learning process. There are filters that will begin to detect things like: "Is this a face?" or in our case, "Is this a barcode?" All from just a little linear algebra!

Bringing It All Together

The power in these neural networks is their ability to automatically learn arbitrary functions and features without explicitly writing complex algorithms to look for them. The dynamic of the industry has begun to change from a pure algorithms approach to a data-driven approach. Because we have many years of customer images and data to work with, we're able to leverage the vastly different conditions of real data to shape the ways we find things with our software.

The days of writing complicated algorithms specialized to classify your data are beginning to wane. Neural networks and deep learning have transformed the process from a domain-specific knowledge task to a data-driven task, and this is an exciting time to be in the industry.

Where Can I Learn More?

If you're interested in developing deep learning solutions, the most popular machine learning libraries out there are ***tensorflow*** and ***pytorch***.

For a general overview of how neural networks work, the superlative ***3Blue1Brown video series*** is something that must be watched.

Finally, for a better understanding of how Convolutional Neural Networks work and are applied, the ***CS231n lectures*** from Stanford are an excellent resource for further knowledge.