Machine Learning

Miles Turpin

Professor Hubert Bray

MATH89S

Introduction

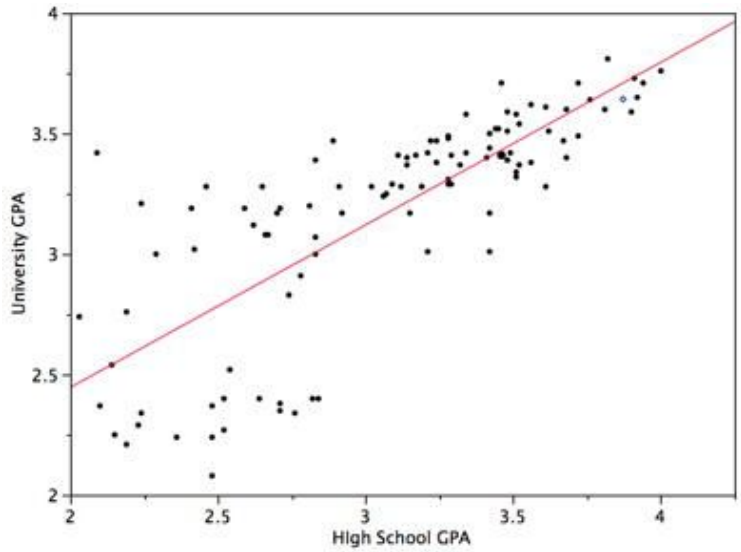
Artificial intelligence (AI) is no longer science fiction. It is becoming increasingly integrated in our daily lives and experts estimate that artificial intelligence of human level or greater intelligence will be developed by the end of the century (Bostrom). Given the impact that the technology has had already and its potential to reshape the world in the future, it is critical to understand how it works.

So what makes a program AI and not just any other kind of program? The key here is *learning*. Without being explicitly programmed, the program adapts over time to better accomplish some goal. For most programs, the range and accuracy of their outputs are limited to how they were coded initially. Imagine a program that takes an image as an input, and returns whether the image is of a dog or a car (assume we only give it images of a dog or a car). A program without AI might mistakenly identify a car as a dog, but is unable to change to correct for this in the future. If a program with AI capability also mistakenly identified a car as a dog, it is designed to learn from past mistakes and adjust internal parameters to return the right answer in the future. While learning seems like a pretty simple idea, the implementation can turn out to be much more complex. I will try to shed light on machine learning overall, gradient descent, neural networks, and applications of machine learning.

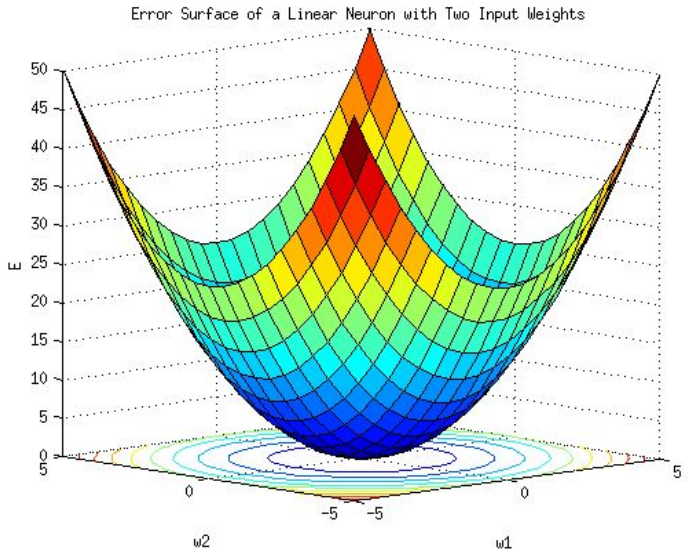
Machine Learning

So the big picture for machine learning is this: we have some data, we want to find a mathematical model that best fits the data, and as we interact with the model (essentially adding more data) the model will update itself to even better fit the data. One of the most simple cases of machine learning is *linear regression*. Whenever you use your TI-84 or excel to fit a line to data you are using machine learning. I found a random data set relating high school and university GPA. In this case of linear regression, where your model is of the form $y = ax + b$, x and y are called the *features* of the model,

while a and b are the *parameters*. We had data on high school and university GPA's and the machine learning program adjusted the parameters of the model until it found the best fit.



Now to make this process more rigorous... to find the model that best fits the data, we have to find a way to quantify how well a certain model does just that. *Cost functions* quantify how well the model fits the data with its current parameters. When we're finding a best fit line (or best fit surface for more variables), the cost function is defined that we simply add up the difference between the expected values (what the model predicts) and actual values (what the data says). The idea is that the best fitting model will have the lowest cost. Whichever set of parameters gives us this lowest cost is then the best fitting model (Goodfellow).



For a model with two inputs the cost function might look something like this bowl-like shape to the left. Imagine that this is the cost function for the example of the high school and university GPA. This image has the two parameters for *a* (slope of the best fit line) and

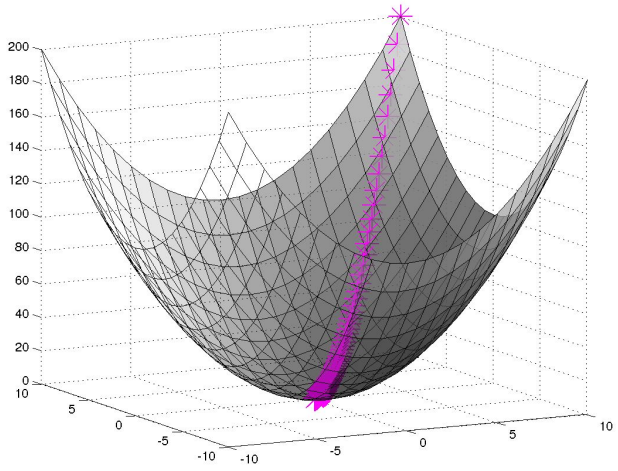
b (y intercept of the best fit line) on the x and y axes, and the cost value on the vertical z axis. This graph is showing the cost value for every permutation of parameters - each data point on this surface is a line on the graph above. For example, the point (2,3,15) on this cost function graph refers to the line $y = 2x + 3$ on the high school vs university GPA graph above, and this line has a calculated cost of 15.

From here it is a simple optimization problem, finding the combination of parameters with the lowest cost (which looks to be point (0,0,0) in this cost function above). Unfortunately, finding the minimum cost is not as simple as calculating the value for every permutation of input parameters and selecting the smallest one. For large numbers of parameters, the number of cost values one would have to calculate becomes astronomical. Instead, we rely on an algorithmic technique called *gradient descent*.

Gradient Descent

The best way to anthropomorphize gradient descent is to imagine as if you were standing blindfolded on a hill and you are trying to get to the bottom. You cannot see so you feel with your feet around you which direction is going downhill, then you take a step in that direction. You continue this process following a path that winds down the hill until you eventually reach the bottom.

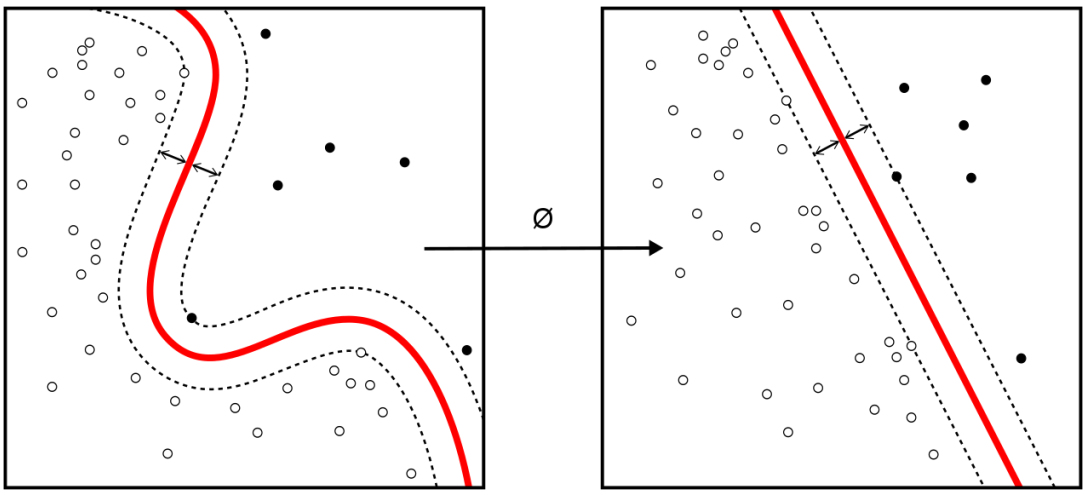
Gradient descent is an analogous process. We start by picking a random point, which are some random parameters for the model. Just as how you felt around with your feet, we can use calculus to find the derivative of the surface (called the *gradient* in 3 or more



dimensions, hence gradient descent), which tells us which direction points towards the minimum of the model. We take a step in the direction of the gradient giving us a set of new parameters closer to the minimum. We calculate the gradient at the new point and repeat the process until the gradient reaches zero. Gradient descent thus gives us a method to find which combination of parameters fits our data the best while saving us a significant number of computations (LeCun).

Classification

You might have been wondering how a model similar to the one used to predict university GPA from high school GPA would be used to tell dogs from cars. While the GPA model was a continuous model, the dogs vs. cars one is an example of a *classification model*. Where you use a continuous model to get a continuous range of outputs for every input (e.g. a predicted university GPA from every high school GPA), a classification model gives a finite range of outputs for every input (returning a value of "dog" or "car" per each input). This uses the same process illustrated before except the model is not used as a line of best fit for the data, but rather as a *decision boundary* that separates the 2 (or more) kinds of outputs (Goodfellow).



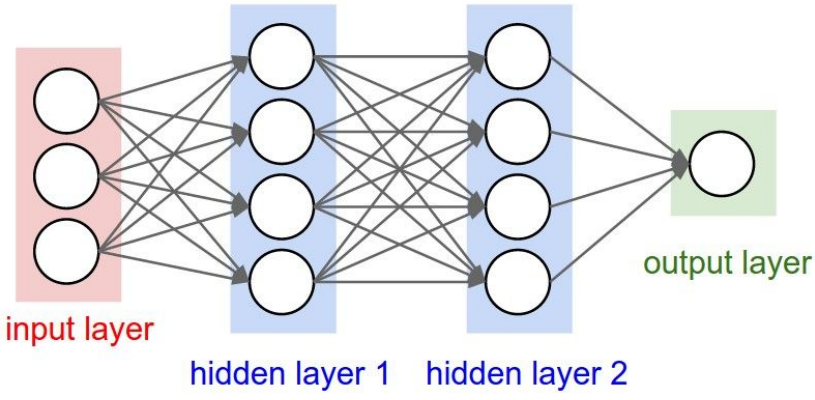
With an understanding of this linear regression and classification models, it's easy to see how powerful these techniques could be with more parameters and more complex data.

Neural Networks

Neural Networks were first introduced in the 1980's as a technique for machine learning, but they never took hold. That is partly because for them to run effectively, a lot of computational power is required as well as a wealth of data to train the network. In just the past 5 years or so, neural networks have made a huge comeback and are now at the cutting edge of AI (Brownlee).

What is so appealing to us about neural networks is that they continuously improve with more data, bigger models, and more computation. Other data science technologies seem to plateau after a certain point, while neural networks have not yet showed signs of performance decline (Brownlee).

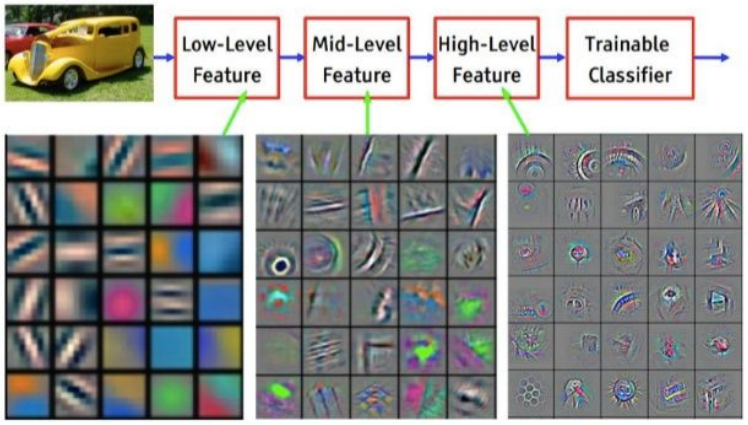
As the name suggests, neural networks are indeed loosely based on the idea of neurons. The inspiration comes from the fact that neurons themselves are incredibly simple - they have inputs and outputs, and if their inputs reach a certain threshold level, they trigger their outputs. Even with this simple behavior on a low-level scale, when connected into circuits and coordinated with other neurons, they demonstrate incredible complexity on a high-level. Neural networks use this same principle. Each "neuron" is it's own mini



classification model that takes inputs from other neurons and according to it's internal classification model will either provide an output or not (Goodfellow).

Another advantage of Deep Learning is hierarchical feature learning (features again being specific inputs to a model; x is a feature of the model y = mx+b). Neural networks are set up structurally as layers of nodes, each layer a higher level feature, hence the description "hierarchical feature learning." These layers of of nodes are also why neural networks are referred to as *deep learning*. Depth refers to how many layers the network contains (LeCun). Yoshua Bengio commented on neural networks that, "Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features."

For a concrete example, consider a neural network trained to identity cars. On a low level the nodes might be trained to identify edges and dots. These low level features combine to mid level features in the middle layers, then by the end, the highest layer essentially determines whether it is a car or not. We do not provide the features for the network to search for - the computer finds the features that most effectively allow it to determine whether or not an object is a car.



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Applications

Computer Vision. Being able to perceive objects in the computer's environment and reason about them. The problem I suggested at the beginning - designing a program to tell a car from a dog - is a basic computer vision problem. Self driving cars rely on advanced computer vision technology to understand their surroundings and react appropriately to changing conditions.

Healthcare. There is so much research on diseases and drugs that it can quickly overwhelm doctors. Computers, with their ability to quickly parse data, assist doctors with AI tech by using machine learning models in combination with the newest research to provide the best treatment for each patient.

Natural Language Processing. This is one of the top goals of AI researchers. Moravec's paradox states that high level reasoning requires very little computation while low level skills require much more. What is paradoxical about this is that the things that are hard for us to do (like algebra and playing chess) are achieved easily by computers, while the reverse is true for basic human skills (like perception, language, and motion) (Goodfellow). Human language is incredibly complex, relying on the context of convoluted societal norms and neuroscience to understand things like sarcasm, jokes, double meanings, slang, and . Natural language processing is an incredibly difficult engineering problem. It is so difficult in fact that many regard it as an AI-complete problem - if we develop a program capable of fluid interaction through human language, that would imply an intelligence equal to our own (Bostrom). This argument is the basis for the famous Turing Test.

Looking to the Future

Much of what I spoke of above - linear regression, neural nets, convolutional neural nets - these techniques are all used for *supervised learning*. Supervised learning refers to labeled data - we give a computer a bunch of pictures cats and eventually (with the right algorithms) the computer learns to identify cats. However, labeled data is expensive (it takes a lot of time/effort to create and label the millions of data points necessary to train a machine learning model) and it is not necessarily how we understand the world (Goodfellow). As children we do not know a cat is a cat, but we learn to differentiate cats from dogs. Most of the learning we do is in fact unsupervised learning.

Being able to do unsupervised learning effectively will be the game changer for AI. It would essentially be a general learning algorithm. The computer would be able to create its own models for different situations and learn how to interpret the world on its own. Computers are limited by the fact that they can only learn what we tell them to learn. An unsupervised learning agent would be have the capacity to take raw input from the environment around it and start to make sense of its surroundings. Hopefully this will be the final step to the creation of an AI which will radically change our world (Bostrom).

Works Cited

Bostrom, Nick. *Superintelligence: Paths, Dangers, Strategies*. Oxford: Oxford UP, 2016. Print.

Brownlee, Jason. "What Is Deep Learning?" *Machine Learning Mastery*. Machine Learning Mastery, 22 Sept. 2016. Web. 09 Dec. 2016.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge, MA: MIT, 2016. Print.

Yann LeCun. "Artificial Intelligence, Revealed." *Facebook Code*. Facebook, n.d. Web. 09 Dec. 2016.