

Recommended by Sean Everett, Jiaconda, and 403 others



Per Harald Borgen [Follow](#)

Developer @Xeneta_AS. Follow me to read about how to learn new stuff.

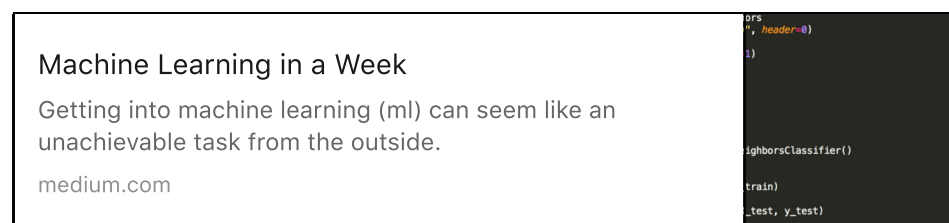
17 hrs ago · 7 min read

```
def gradient(y, y_hat):  
    return y_hat - y  
  
def softmax(x):  
    x = np.exp(x)  
    row_sum = np.sum(x, axis = 1)  
    x /= row_sum.reshape((x.shape[0], 1))  
    return x  
  
def feedForward(X, W):  
    prediction = softmax(np.dot(X, W))  
    return prediction
```

Machine Learning in a Year

From being a total ml noob to start using it at work

This is a follow up to an article I wrote last year, *Machine Learning in a Week*, on how I kickstarted my way into machine learning (ml) by devoting five days to the subject.



After this highly effective introduction, I continued learning on my spare time and almost exactly one year later I did my first ml project at work, which involved using various ml and natural language processing (nlp) techniques to qualify sales leads at Xeneta.

This felt like a blessing: getting paid to do something I normally did for fun!

It also ripped me out of the delusion that only people with masters degrees or Ph.D's work with ml professionally.

The truth is you don't need much maths to get started with machine learning, and you don't need a degree to use it professionally.

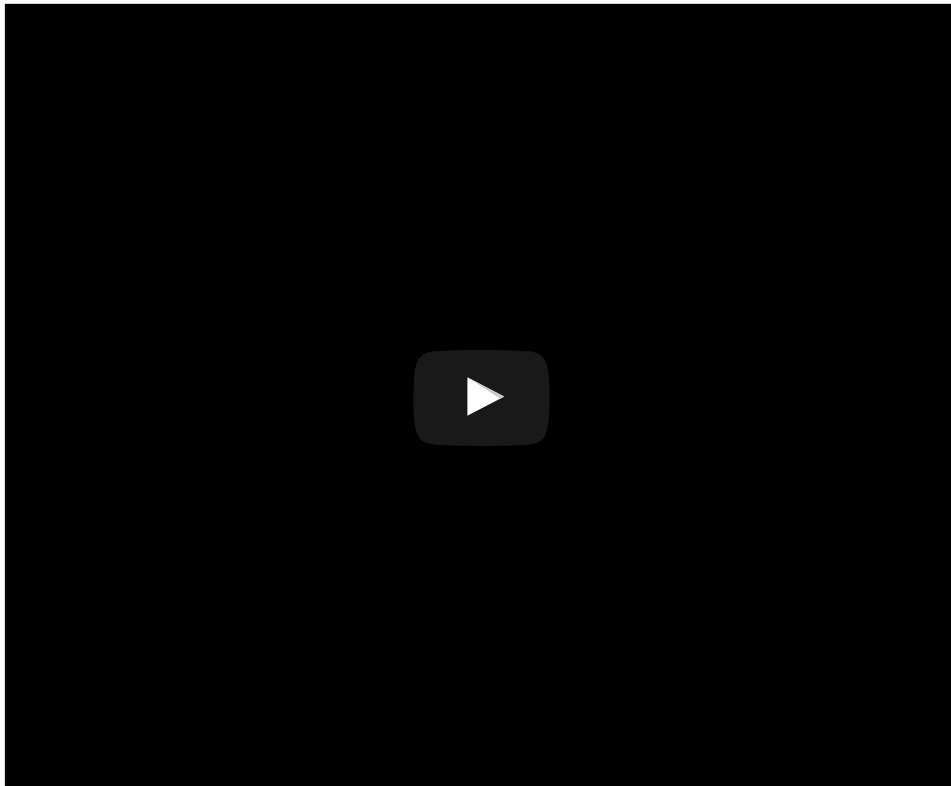
In this post I want to share my journey, as it might inspire others to do the same.

. . .

First intro: Hacker News and Udacity

My interest in ml stems back to 2014 when I started reading articles about it on Hacker News. I simply found the idea of teaching machines stuff by looking at data appealing. At the time I wasn't even a professional developer, but a hobby coder who'd done a couple of small projects.

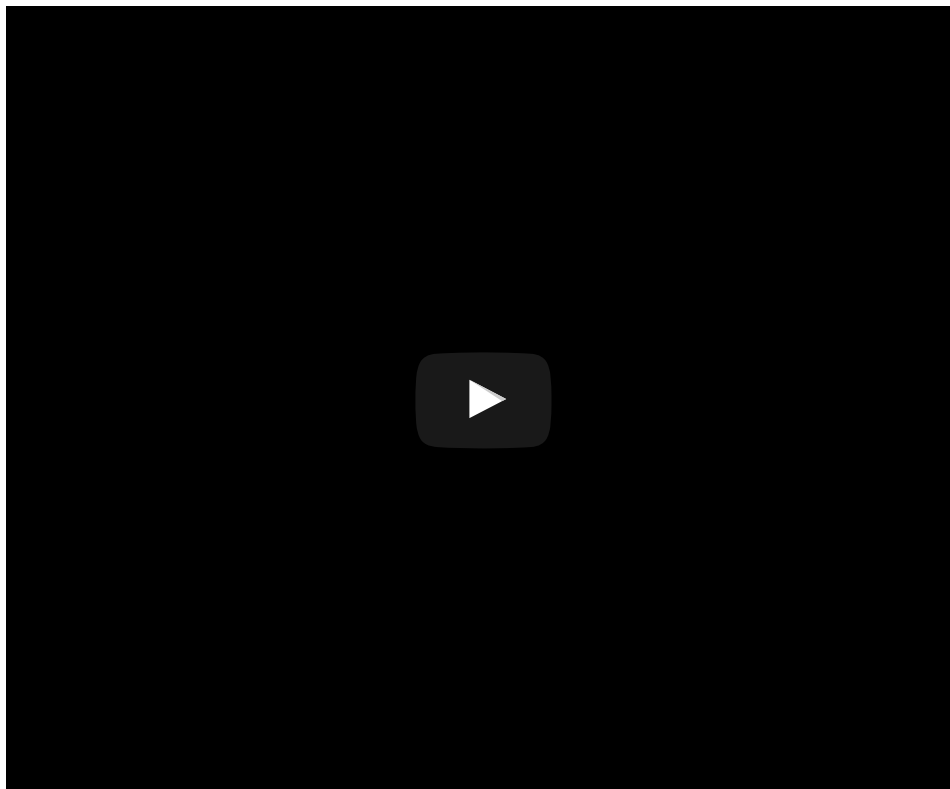
So I began watching the first few chapters of Udacity's Supervised Learning course, while also reading all articles I came across on the subject.



This gave me a little bit of conceptual understanding, though no practical skills. I also didn't finish it, as I rarely do with MOOC's.

Failing Coursera's ML Course

In January 2015 I joined the Founders and Coders (FAC) bootcamp in London in order to become a developer. A few weeks in, I wanted to learn how to actually code machine learning algorithms, so I started a study group with a few of my peers. Every Tuesday evening, we'd watch lectures from Coursera's Machine Learning course.



It's a fantastic course, and I learned a hell of a lot. But it's tough for a beginner. I had to watch the lectures over and over again before grasping the concepts. The Octave coding tasks are challenging as well, especially if you don't know Octave. As a result of the difficulty, one by one fell off the study group as the weeks passed. Eventually, I fell off it myself as well.

In hindsight, I should have started with a course that either used ml libraries for the coding tasks—as opposed to building the algorithms from scratch—or at least used a programming language I knew.

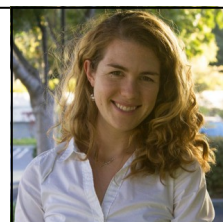
Learning a new language while also trying to code ml algorithms is too hard for a newbie.

If I could go back in time, I'd choose Udacity's Intro to Machine Learning, as it's easier and uses Python and Scikit Learn. This way, we would have gotten our hands dirty as soon as possible, gained confidence, and had more fun.

Intro to Machine Learning Course | Udacity

You'll learn how to start with a question and/or a dataset, and use machine learning to turn them into insights. Naive...

www.udacity.com



Lesson learned: Start with something easy and practical rather than difficult and theoretical.

Machine Learning in a Week

One of the last things I did at FAC was the ml week stunt. My goal was to be able to apply machine learning to actual problems at the end of the week, which I managed to do.

Throughout the week I did the following:

- got to know **Scikit Learn**
- tried ml on a **real world dataset**
- coded a **linear regression** algorithm from scratch (in Python)
- did a tiny bit of **nlp**

It's by far the steepest ml learning curve I've ever experienced. Go ahead and read the article if you want a more detailed overview.

Lesson learned: Setting off a week solely to immerse yourself into a new subject is extremely effective.

Failing neural networks

After I finished FAC in London and moved back to Norway, I tried to repeat the success from the ml week, but for neural networks instead.

This failed.

There were simply too many distractions to spend 10 hours of coding and learning every day. I had underestimated how important it was to be surrounded by peers at FAC.

Lesson learned: Find an thriving environment to surround yourself with when doing these kinds of learning stunts.

However, I got started with neural nets at least, and slowly started to grasp the concept. By July I managed to code my first net. It's probably the crappiest implementation ever created, and I actually find it embarrassing to show off. But it did the trick; I proved to myself that I understood concepts like **backpropagation** and **gradient descent**.


```

33
34     def SumOfSqErrors(self,X,Y):
35         error = 0
36         for i in xrange(len(X)):
37             target = Y[i]
38             inputs = X[i]
39             output = self.Forward(target,inputs)
40             error += self.SqErrors(target,output)
41         return error / len(X)
42

```




In the second half of the year, my progression slowed down, as I started a new job. The most important takeaway from this period was the leap from **non-vectorized** to **vectorized** implementations of neural networks, which involved repeating linear algebra from university.

By the end of the year I wrote an article as a summary of how I learned this:

<p>Learning How To Code Neural Networks</p> <p>This is the second post in a series of me trying to learn something new over a short period of time. The first time...</p> <p>medium.com</p>	
---	--

Testing out Kaggle Contests

During the christmas vacation of 2015, I got a motivational boost again and decided try out Kaggle. So I spent quite some time experimenting with various algorithms for their Homesite Quote Conversion, Otto Group Product Classification and Bike Sharing Demand contests.

	Predicting Red Hat Business Value Classify customer potential 10 days to go • Featured	2,034 teams 1,723 kernels \$50,000
	Bosch Production Line Performance Reduce manufacturing failures 2 months to go • Featured	299 teams 31 kernels \$30,000
	Melbourne University AES/MathWorks/NIH Seizure Prediction Predict seizures in long-term human intracranial EEG recordings 2 months to go • Research	151 teams 72 kernels \$20,000

Kaggle is a fun way to experiment with datasets and get feedback on your performance.

The main takeaway from this was the experience of iteratively improving the results by experimenting with the algorithms and the data.

I learned to **trust my logic** when doing machine learning.

If tweaking a parameter or engineering a new feature seems like a good idea logically, it's quite likely that it actually will help.

Setting up a learning routine at work

Back at work in January 2016 I wanted to continue in the flow I'd gotten into during Christmas. So I asked my manager if I could spend some time learning stuff during my work hours as well, which he happily approved.

How To Setup A Learning Routine At Work

4 tips to grow your skills while being hired

medium.com



Having gotten a basic understanding of neural networks at this point, I wanted to move on to deep learning.

Udacity's Deep Learning

My first attempt was Udacity's Deep Learning course, which ended up as a big disappointment. The contents of the video lectures are good, but they are too short and shallow to me.



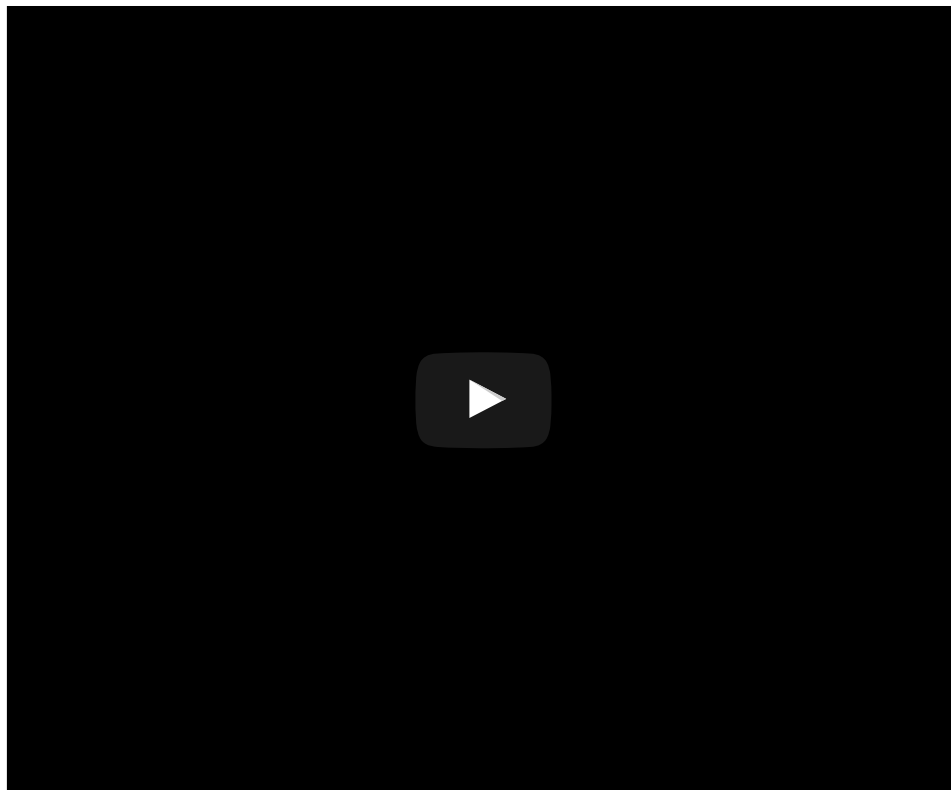
And the IPython Notebook assignments ended up being too frustrating, as I spent most of my time debugging code errors, which is the most effective way to kill motivation. So after doing that for a couple of sessions at work, I simply gave up.

To their defense, I'm a total noob when it comes to IPython Notebooks, so it might not be as bad for you as it was for me. So it might be that I simply wasn't ready for the course.

Stanford—Deep Learning for NLP

Luckily, I then discovered [Stanford's CS224D](#) and decided to give it a shot. It is a fantastic course. And though it's difficult, I never end up debugging when doing the problem sets.

Secondly, they actually give you the solution code as well, which I often look at when I'm stuck, so that I can work my way backwards to understand the steps needed to reach a solution.



Though I've haven't finished it yet, it has significantly boosted my knowledge in nlp and neural networks so far.

However it's been tough. Really tough. At one point, I realized I needed help from someone better than me, so I came in touch with a Ph.D student who was willing to help me out for 40 USD per hour, both with the problem sets


as well as the overall understanding. This has been critical for me in order to move on, as he has uncovered a lot of black holes in my knowledge.

Lesson learned: *It's possible to get a good machine learning teacher for around 50 USD per hour. If you can afford it, it's **definitely** worth it.*

Boosting Sales at Xeneta

After doing all this, I finally felt ready to do a ml project at work. It basically involved training an algorithm to qualify sales leads by reading company descriptions, and has actually proven to be a big time saver for the sales guys using the tool.

Check out out article I wrote about it below or head over to GitHub to dive straight into the [code](#).

<p>Boosting Sales With Machine Learning</p> <p>How we use natural language processing to qualify leads</p> <p>medium.com</p>	
--	--

Getting to this point has surely been a long journey. But also a fast one; when I started my machine learning in a week project, I certainly didn't have any hopes of actually using it professionally within a year.

But it's 100 percent possible. And if I can do it, so can anybody else.

. . .

Thanks for reading! My name is Per Harald Borgen, I write articles about learning new stuff, once per month on average.

If you'd like to get notified of about my next article, [just enter your email](#).

