Machine Learning for Mathematicians

Daniel Mckenzie

Tuesday March 6th, 2018

Why should we care about Machine Learning

1. Necessary for non-academic jobs.
2. Can be useful in your research.
3. Your (future) students will need to know about it.

An outline of this talk

1. Disambiguation of buzzwords.
2. Simple (yet effective) approaches.
3. Deep approaches.
4. Survey of applications.
5. Current research trends.

What do we mean by Data?

1. Could be images, audio signals, stock prices, results of surveys etc.
2. Can always **vectorize**

Example (Greyscale Images)

Suppose each I_a *is a* 28×28 *array of pixel values. Each pixel value is a number between* 0 *and* 256 *with* $0 =$*black and* $256 =$*white. Can think of each* I_a *as a* 28×28 **matrix** A^a_{ij}. *Can make this a vector by stacking:* $\mathbf{x}^a = [A_{11}, \ldots, A_{1,28}, A_{2,1}, \ldots, A_{2,28}, \ldots, A_{28,28}]$

More sophisticated approaches uses Fourier Transform or Wavelets (Applied Harmonic Analysis). Each entry of vector is called a **feature**.

3. Three V's of Big Data: **V**ariety, **V**olume and **V**elocity.

Data Science, Machine Learning, and Artificial Intelligence[1]

1. **Data Science:** Produce insights from data for humans.
2. **Machine Learning:** Find a function f that *predicts* y from input x. Eg $f(\text{Image}) = \text{cat}$. How f is doing this is often unclear.
3. **Artificial Intelligence:** Produce or recommend an action from data. Eg AlphaGo, self-driving cars.

Caution: Distinction between 'general' AI (long way off/impossible) and 'single purpose' AI (AlphaGo, self-driving cars).

[1] Robinson 2018.

Data Science

Data Scientists use

1. Statistical know how to 'wrangle' complex data in a variety of formats into a clean, usable (vectorized) data set X.
2. Algorithms (Regression, Data Clustering, Neural Networks etc) to extract insights from X. E.g.: identify a trend/ correlation, find outliers (fraud prevention), compute quantities of interest (likelihood of certain type of consumer to renew cable contract).
3. Domain-specific knowledge to evaluate appropriateness of the above.

to produce easily interpretable summaries (pie charts, reports, visualizations) to inform decision-making of other parties (management, sales team, R& D, government).

(Supervised) Machine Learning

1. **Model Problem:** Identify people from pictures.
2. **Key assumption:** Let $\mathbf{D}$ be domain of interest (e.g. all possible 28×28 pictures). Let $\mathbf{C}$ be codomain of interest (e.g. the names of people we wish to identify). We assume there exists a continuous function $f^* : \mathbf{D} \to \mathbf{C}$ mapping all photos of Dan to 'Dan' $\in \mathbf{C}$.
3. **Goal of Machine Learning:** function approximation. Find an approximation $f^\#$ to f^*.
4. **Learning $f^\#$:** Given training set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbf{D}$ and known labels $Y = \{y_1, \ldots, y_n\} \subset \mathbf{C}$. Find function $f^\#$ such that $f^\#(\mathbf{x}_i) \approx y_i$ for all i.

Caution: Generalizability very important. Need to be confident that given $\mathbf{x} \notin X$ $f^\#(\mathbf{x}) \approx f^*(\mathbf{x})$

Artificial Intelligence

This slide intentionally left blank

Simple Approaches to Machine Learning[2]

1. Let $\mathcal{P}$ be a class of 'easy functions' (e.g. piecewise polynomial). Find $f^{\#}$ as:

$$f^{\#} = \text{argmin}\{L(f, X) : \; f \in \mathcal{P}\}$$

   Think $L(f, X) = \sum_{i=1}^{n} \|f(\mathbf{x}_i) - y_i\|_2$. Regression, Splines, Finite Elements.

2. K-Nearest Neighbours. Let $\mathcal{N}^K(\mathbf{x}) = \{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_K}$ nearest to $\mathbf{x}\}$. Compute $f^{\#}(\mathbf{x}) = \frac{1}{K}\sum_{j=1}^{K} y_{i_j}$.
3. Support Vector Machine.
4. Decision trees.

[2] Goodfellow et al. 2016.

Simple Approaches: Logistic Regression

Suppose $|\mathbf{C}| = 2$ e.g. $\mathbf{C} = \{\text{'Dan'}, \text{'Not Dan'}\}$. Define *sigmoid/ logistic function* $g(u) = 1/(1 + e^{-u})$. Look for $f^{\#}$ of the form:

$$f_{\mathbf{w}}(\mathbf{x}) = \begin{cases} \text{'Dan' if } g(\mathbf{w}^\top \mathbf{x}) \approx 1 \\ \text{'not Dan' if } g(\mathbf{w}^\top \mathbf{x}) \approx 0 \end{cases}$$

That is, $f^{\#} = \operatorname{argmin}\{L(f_{\mathbf{w}}, X) : \mathbf{w} \in \mathbb{R}^n\}$. Can think of $z = g(\mathbf{w}^\top \mathbf{x})$ as probability that the image contains Dan.

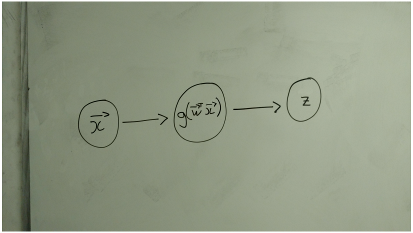


Figure: Schematic depiction of Perceptron

(Shallow) Neural Networks

Essentially iterated Logistic Regression:

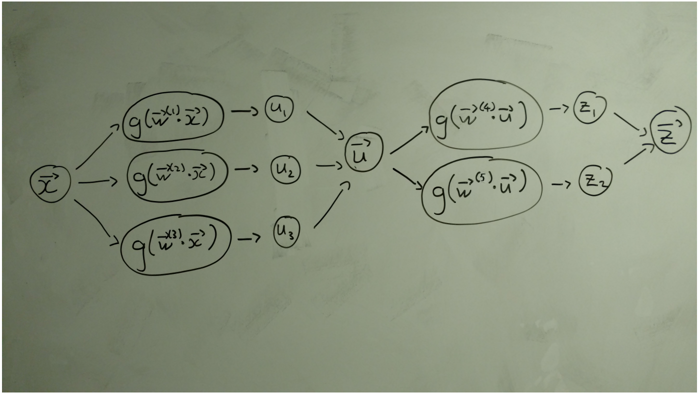


Figure: Schematic depiction of 2-layer Neural Network

(Shallow) Neural Networks cont.

Notation: $f_{\mathbf{W}}$ denotes Neural Network with weights $\mathbf{W} = \{\mathbf{w}^1, \ldots, \mathbf{w}^5\}$. $f_{\mathbf{W}}(\mathbf{x}) = \mathbf{z}$.
Typically, $z_1 =$ probability $\mathbf{x}$ in class 1, $z_2 =$ probability $\mathbf{x}$ in class 2.
Architecture: Choice of number of layers and neurons per layer.
Activation function: g. Many other choices, but *must be non-linear*!.
These layers are fully connected.
Need to find good $\mathbf{W}$. will vectorize: $\mathbf{w} = [\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^5]$.
Need to solve $f^{\#} = \text{argmin}\{L(f_{\mathbf{w}}, X) : \ \mathbf{w} \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \times \ldots \mathbb{R}^{n_5}\}$

Gradient Descent

1. **The problem:** Find minimum of $F : \mathbb{R}^m \to \mathbb{R}$. Can assume that F is differentiable.
2. Know that $-\nabla F(\mathbf{w}) \subset \mathbb{R}^m$ points in direction of steepest decrease of F at $\mathbf{x}$.
3. **Gradient Descent Algorithm:** $\mathbf{w}^{k+1} = \mathbf{w}^k - \epsilon \nabla F(\mathbf{w}^k)$.
4. **For Neural Networks:** $F(\mathbf{w}) = L(f_{\mathbf{w}}, X)$ (Think: $L(f_{\mathbf{w}}, X) = \sum_{i=1}^{n} \|f_{\mathbf{w}}(\mathbf{x}_i) - y_i\|_2$). Randomly initialize $\mathbf{w}^0$. Compute $\mathbf{w}^{k+1}$ using gradient descent until 'good enough'.
5. **Issue 1:** Computing ∇L can be costly (typically use Stochastic Gradient Descent).
6. **Issue 2:** L is usually (highly) non-convex. No guarantee that Gradient Descent will converge.

Skills necessary for ML

For Undergrads

1. Coursework: Multivariable calculus, Linear Algebra, Numerical Analysis, Probability.
2. Online resources: `http://cs229.stanford.edu/`, `https://www.coursera.org/learn/machine-learning`

Additional resources for Grads

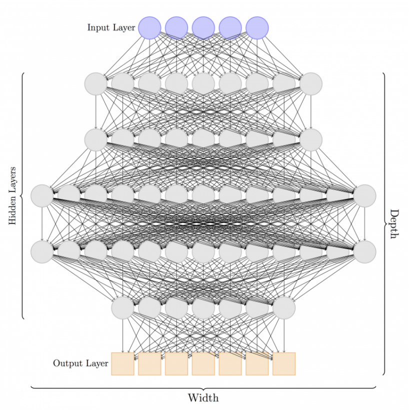
1. Coursework: Harmonic Analysis, Image Processing, Statistics.
2. Some programming.
3. Deep learning book: `http://www.deeplearningbook.org/`
4. 18.657: Graduate Course on Mathematics of Machine Learning taught at MIT (*all materials/lecture notes available online*)
5. Blogs: `http://nuit-blanche.blogspot.com/`

Deep Neural Networks

1. **Key Insight:** Vectorizing/ feature extraction is the most important step.
2. Many techniques from Applied Harmonic Analysis (*e.g.* Wavelets, Curvelets,. . .) could be used.
3. **Deep Learning:** Use many convolutional layers to extract good, problem specific features. Then use a few, fully connected layers to classify.
4. Hinton, Osindero, and Teh 2006 [3] was first to show this was feasible.
5. Krizhevsky, Sutskever, and Hinton 2012 presented a Deep NN halving previous error rate for image classification
6. **Key Drivers of DL:** Increased processing power (GPU's). Large training sets (sourced from the internet).

[3] Geoff Hinton is the great-great-grandson of George Boole, inventor of Boolean logic.

Deep Neural Networks [4]



[4] Figure from:
https://developingideas.me/deepneuralnetworkoverview/

Prototypical Applications of Machine Learning

1. **Handwritten Digit Classification** State-of-the-art algorithms are $> 99.75\%$ accurate.
2. **Automated Captioning:** Given an image, algorithm should output brief sentence describing what is going on .
3. **Natural Language Processing:** Alexa, Siri *et. al.* Sentiment Analysis.

"baseball player is throwing ball in game."

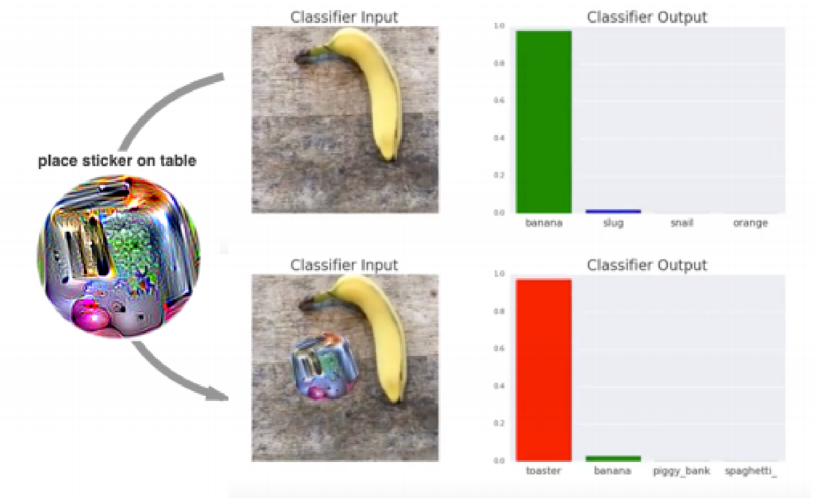
"a horse is standing in the middle of a road."

Figure: First two pictures from Karpathy and Fei-Fei 2015

Current Research Trends

1. Dealing with Data scarcity.
2. Regularization and priors.
3. Transfer Learning: Getting a neural network trained to do one thing (e.g. play 'Pong') to learn to do another thing quickly (e.g. play 'Seaquest') (see Fernando et al. 2017).
4. What the hell is actually going on here? Still not clear how deep neural networks do what they do. This leaves them susceptible to manipulation (*adversarial attacks*).

An Adversarial Patch[5]



[5] Brown et al. 2017.

Applications to Mathematics: Data Driven Dynamical Systems[6]

1. For many physical/ biological systems of interest: $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t))$.
2. Can usually collect historical data via observation: $X = \{\mathbf{x}(t_1), \mathbf{x}(t_2), \ldots, \mathbf{x}(t_n)\}$ and $Y = \{\dot{\mathbf{x}}(t_1), \dot{\mathbf{x}}(t_2), \ldots, \dot{\mathbf{x}}(t_n)\}$
3. **Model:** Assume that $f(\mathbf{x}(t))$ is a sparse linear combination of elementary functions $\varphi_1, \ldots, \varphi_N$ (*e.g.* polynomials, trig. functions etc)
4. Use Machine Learning to find an optimal $f^{\#} = \sum_{i=1}^{N} a_i \varphi_i$. (Strong connections with Compressive Sensing).

[6] Brunton, Proctor, and Kutz 2016.

Application to Mathematics: Predicting Hodge Numbers

1. Let $\mathbb{WP}^4$ be weighted projective space.
2. Large finite number of 3-dim Calabi Yau $M^a \subset \mathbb{WP}^4$. Each cut out by a degree $w = \sum_{i=0}^4 w_i$ homogeneous polynomial.
3. Of interest to string theorists to compute Hodge numbers $h^{i,j}$
4. To each M^a associate the data vector $\mathbf{x}^a$ of coefficients of defining polynomial[7].
5. Training set: $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^m\}$ and $Y = \{h^{2,1}(M^1), \ldots, h^{2,1}(M^m)\}$.
6. In (He 2017), Neural Network was trained in above dataset to predict whether $h^{2,1}(M)$ large (> 50) or not large (≤ 50) given M.
7. They report 94.4% accuracy on unseen data.

[7] Plus possibly some side info like χ

Thanks! Any questions?

Figure: Neural Style Transfer from Johnson, Alahi, and Fei-Fei 2016

References I

- Brown, Tom B et al. (2017). "Adversarial patch". In: *arXiv preprint arXiv:1712.09665.*
- Brunton, Steven L, Joshua L Proctor, and J Nathan Kutz (2016). "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences* 113.15, pp. 3932–3937.
- Fernando, Chrisantha et al. (2017). "Pathnet: Evolution channels gradient descent in super neural networks". In: *arXiv preprint arXiv:1701.08734.*
- Goodfellow, Ian et al. (2016). *Deep learning*. Vol. 1. MIT press Cambridge.
- He, Yang-Hui (2017). "Deep-learning the landscape". In: *arXiv preprint arXiv:1706.02714.*

References II

- Hinton, Geoffrey E, Simon Osindero, and Yee-Whye Teh (2006). "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7, pp. 1527–1554.
- Johnson, Justin, Alexandre Alahi, and Li Fei-Fei (2016). "Perceptual losses for real-time style transfer and super-resolution". In: *European Conference on Computer Vision*. Springer, pp. 694–711.
- Karpathy, Andrej and Li Fei-Fei (2015). "Deep visual-semantic alignments for generating image descriptions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137.

References III

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Robinson, David (2018). *What's the difference between data science, machine learning, and artificial intelligence?* http://varianceexplained.org. Blog.