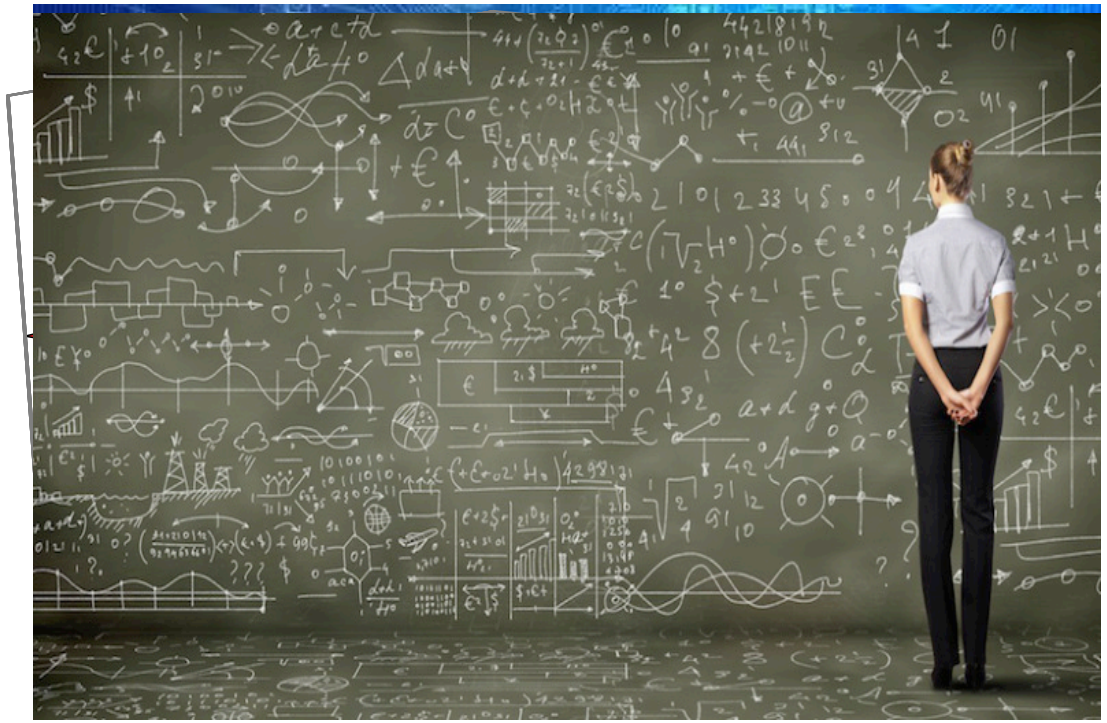
Introduction To Machine Learning
Selected Topics



David Meyer
Brocade Chief Scientist and Fellow
dmm@{brocade.com,uoregon.edu,1-4-5.net,..}
http://www.1-4-5.net/~dmm/ml/talks/2016/ml_intro.20160921.{pptx,pdf}
ML Talk with Ruckus
21 Sep 2016

You might be surprised but what is going to drive innovation in the enterprise and in the public cloud is machine learning.

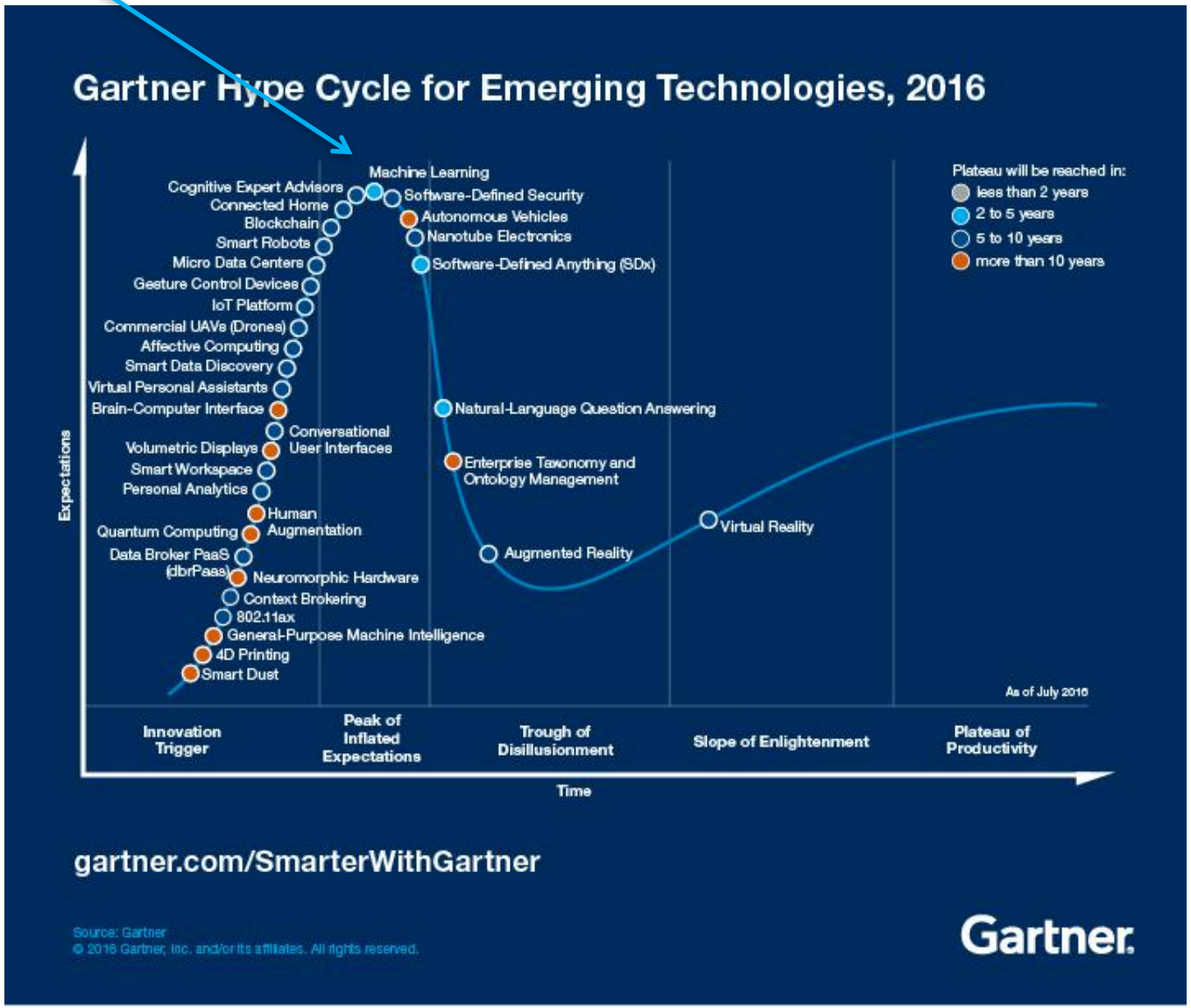
Bill Coughran, Sequoia Capital #ONUGSpring16

Machine Learning is the way we are going to automate your automation.

Chris Wright, RedHat CTO #RHSummit

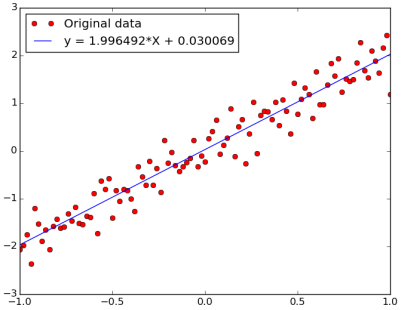
You are here (again)

However…



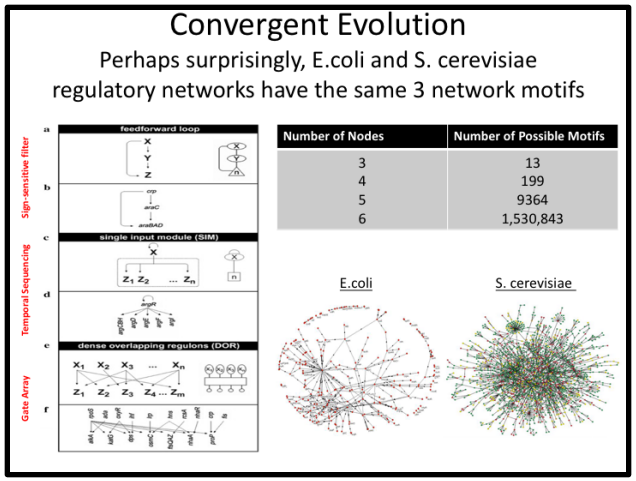
Agenda

- Who Am I?
- Probability Theory Review?
- What Is Machine Learning?
- What is all the (Machine Learning) Excitement About?
- Integrated Approaches
- Histograms, LDA, and Neural Networks
- Machine Learning Excitement Redux: Beyond Static Learning
- Technical explanations/code
  - https://github.com/davidmeyer/ml
  - http://www.1-4-5.net/~dmm/ml



Who Am I?

- Chief Scientist and Fellow at Brocade
- Adjunct Faculty Computer Science University of Oregon
  - http://www.routeviews.org/
- 15 years at Cisco
- 5 years at Sprint
- 4 years at Brocade
- IETF, NANOG, RIPE, OpenDaylight, ...

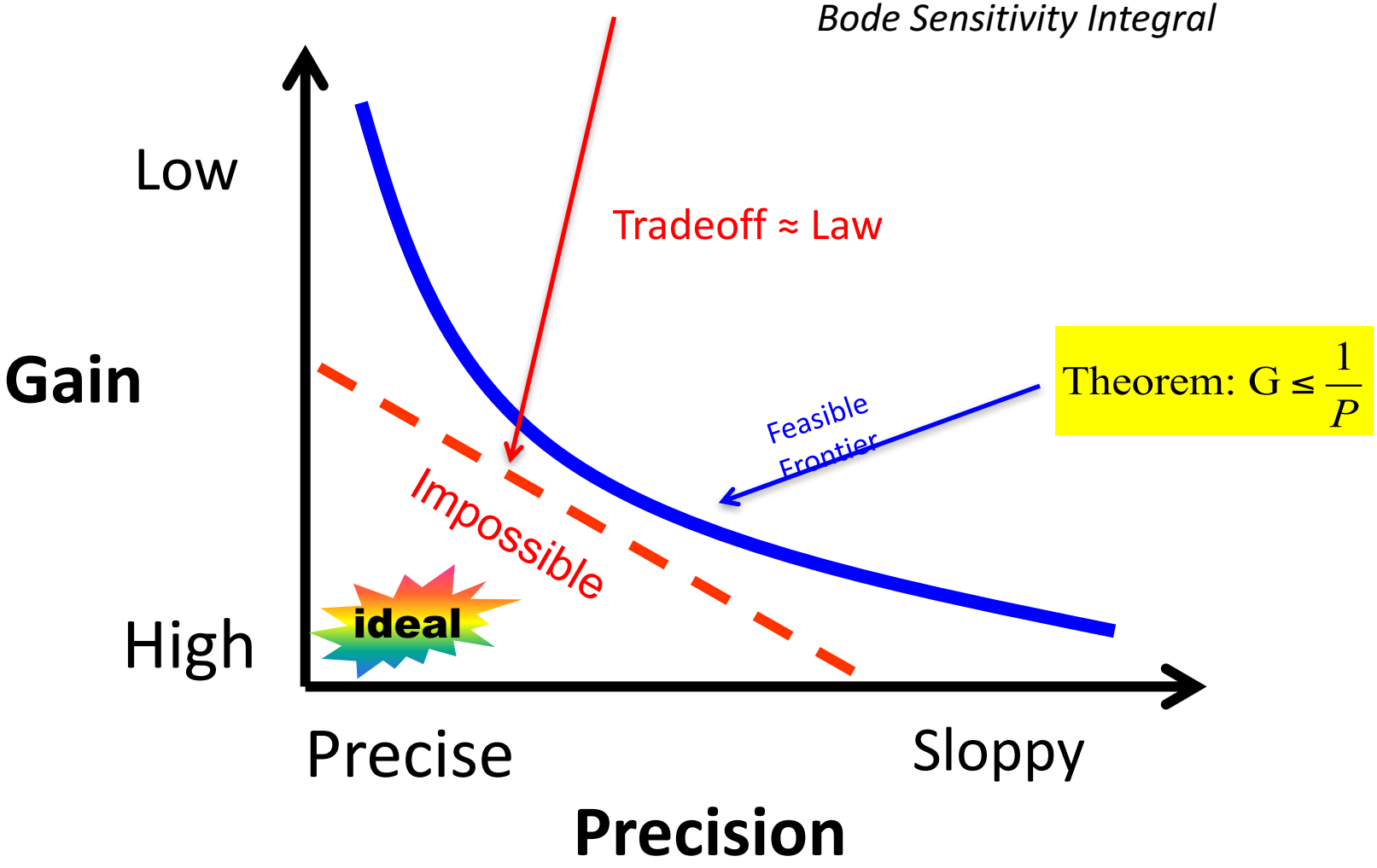


- Other areas I've been active in: Biology, Math, Control Theory, Law, ...
- Focus of last several years: Machine Learning theory and practice for networking
- See http://www.1-4-5.net/~dmm/vita.html for more detail

Gain/Sensitivity Tradeoff In Feedback Control

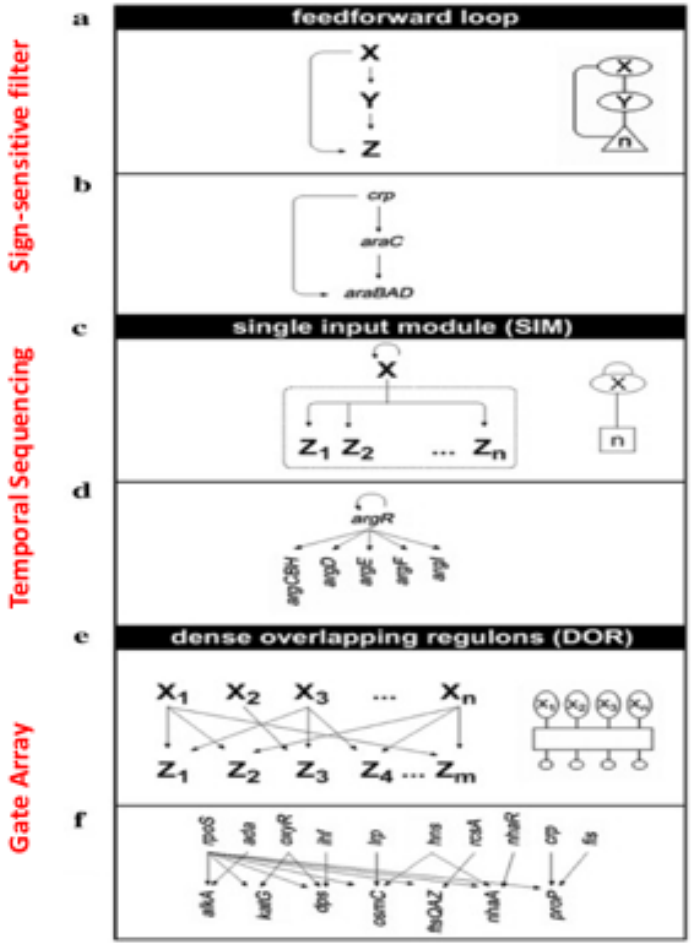
$$\int_0^\infty \ln|S(i\omega)|d\omega = \int_0^\infty \ln\left|\frac{1}{1+L(i\omega)}\right| d\omega = \pi \sum Re(p_k) - \frac{\pi}{2}\lim_{s\to\infty} sL(s)$$

Bode Sensitivity Integral



Convergent Evolution

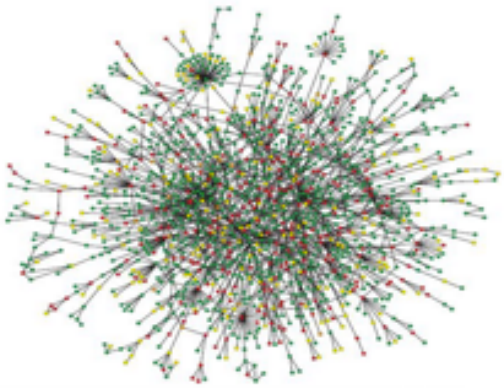
Perhaps surprisingly, E.coli and S. cerevisiae regulatory networks have the same 3 network motifs



Number of Nodes	Number of Possible Motifs
3	13
4	199
5	9364
6	1,530,843

E.coli

S. cerevisiae



Aside: A Couple of Aspects Of Our Challenge

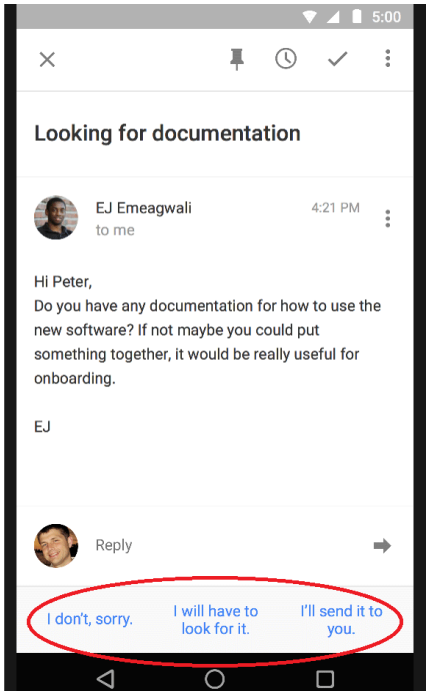
- The current *distance* between theory and practice in Machine Learning is effectively zero

- What does this mean?

- Consider "sequence to sequence learning"
  - https://arxiv.org/pdf/1409.3215v3.pdf
  - Plus "thought vectors"

- Elapsed time from NIPS paper to Google Smart Reply[1]?
  - A little over one year

- This means that the latest theory is being rapidly deployed in product
  - The best ML theory folks are also the ML best coders
  - And the field is moving at an astounding rate
  - e.g., https://github.com/LeavesBreathe/tensorflow_with_latest_papers



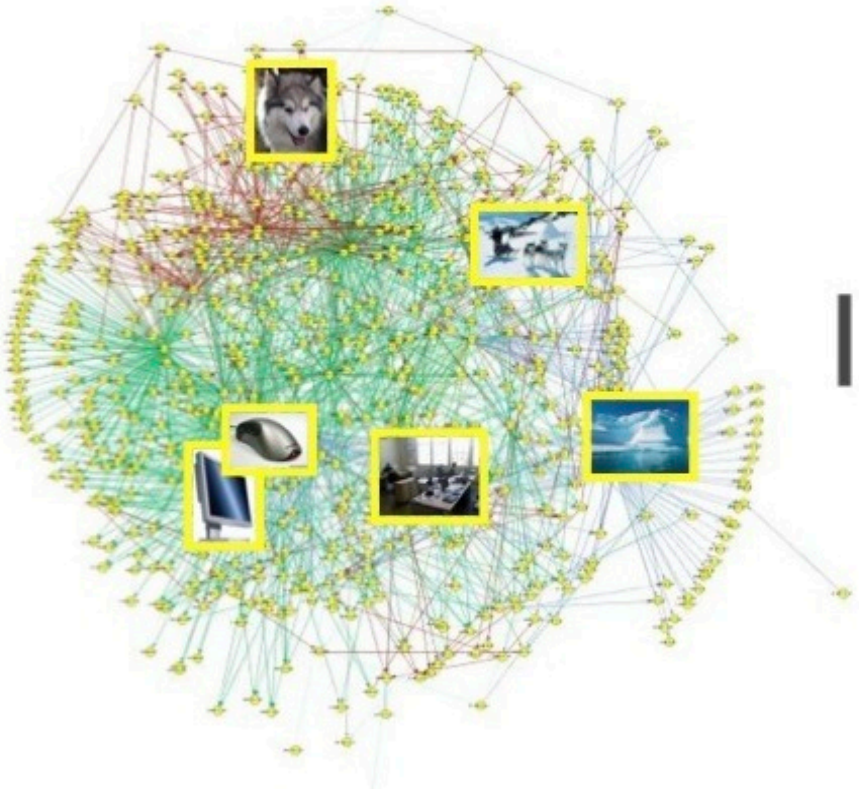
- Which not surprisingly means that achieving SOA ML results requires deep understanding of both theory and practice

- This presents a challenge (skills mismatch) for the networking field

1 https://gmail.googleblog.com/2016/03/smart-reply-comes-to-inbox-by-gmail-on-the-web.html

Another Aspect Of Our Challenge

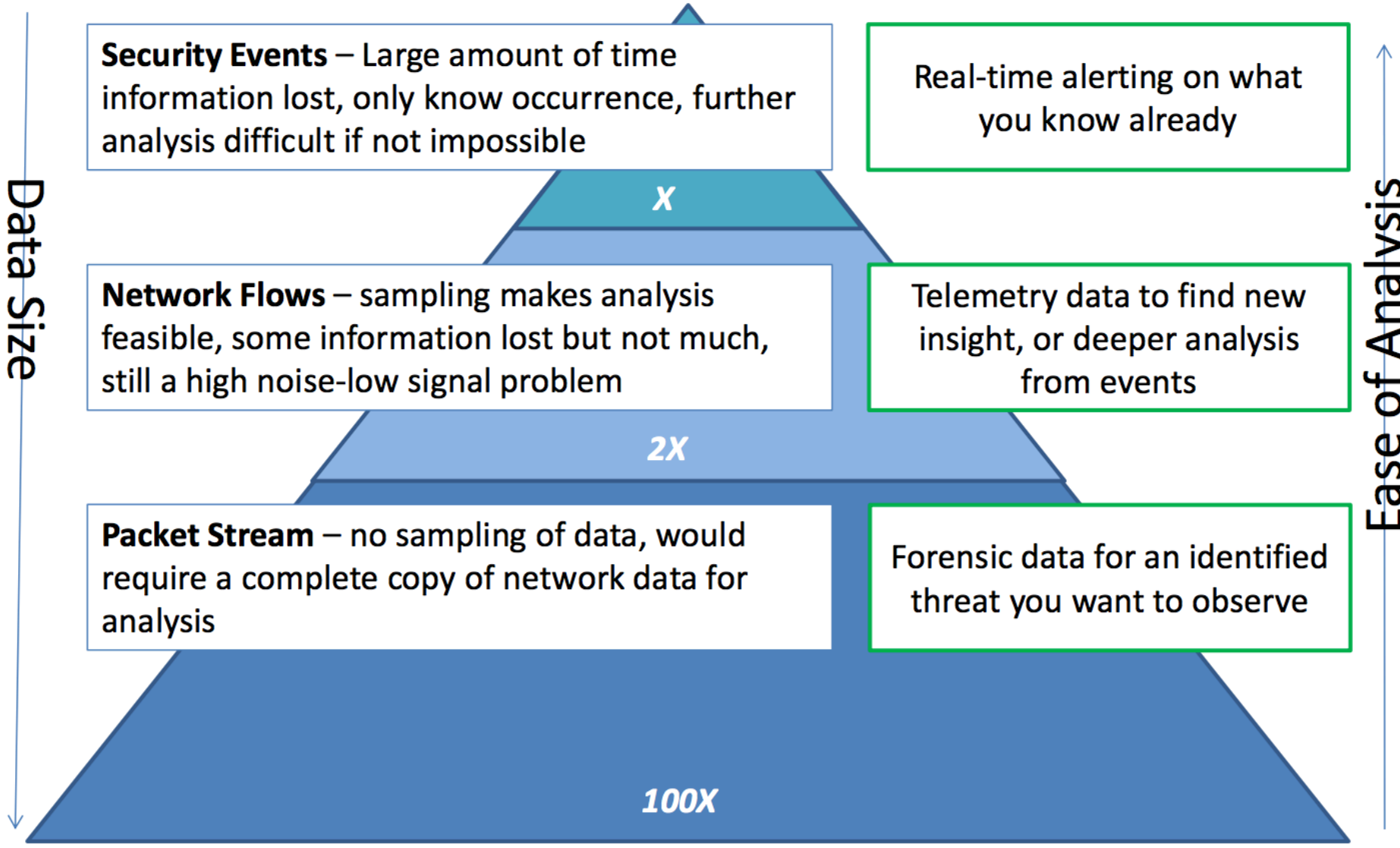
What is our ”ImageNet”?



IMAGENET

And is there a theory of “network”, or is every network a one-off?

Bridging the Gap



Agenda

- Who Am I?
- Level Set: What Is Machine Learning?
- What is all the (Machine Learning) Excitement About?
- Integrated Approaches
- Histograms, LDA, and Neural Networks
- Machine Learning Excitement Redux: Beyond Static Learning
- Technical explanations/code
  - https://github.com/davidmeyer/ml
  - http://www.1-4-5.net/~dmm/ml

First, What is Machine Learning?

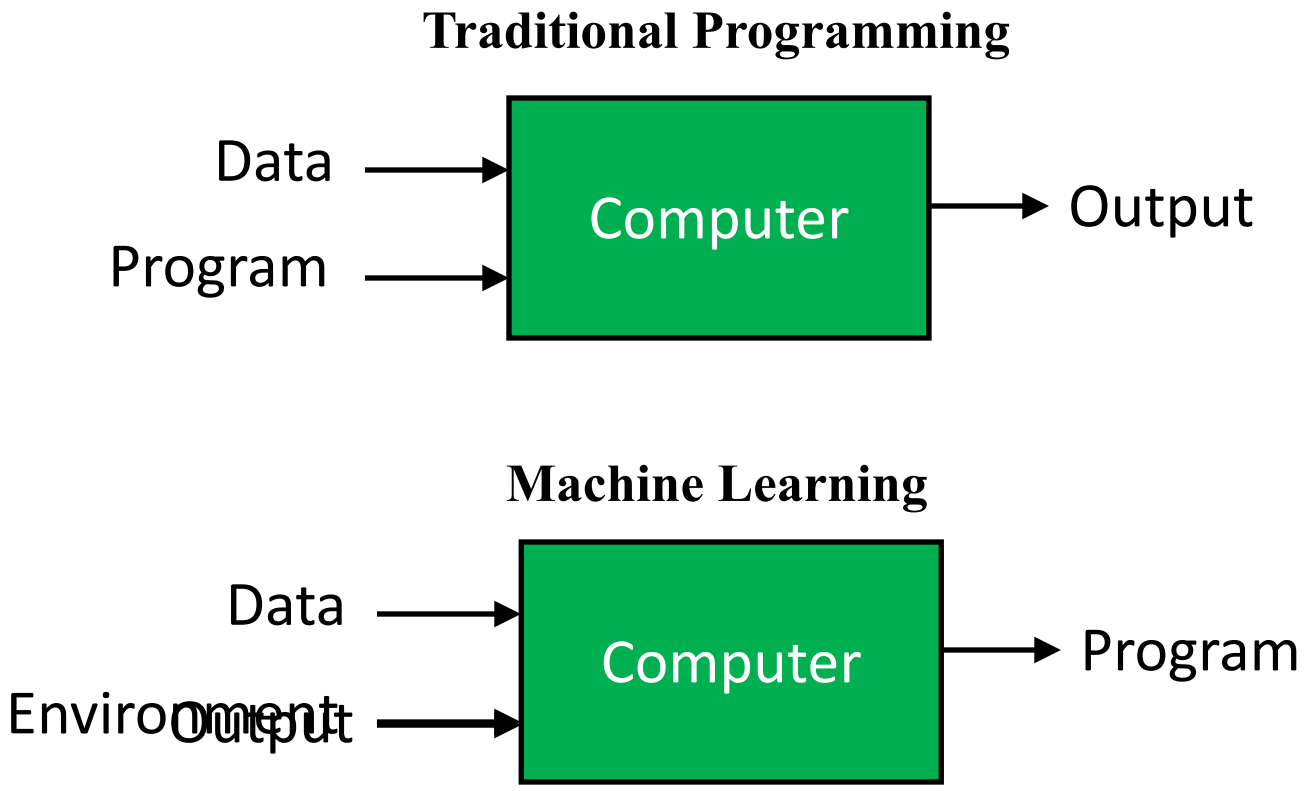
The complexity in traditional computer programming is in the code (programs that people write). In machine learning, learning algorithms are in principle simple and the complexity (structure) is in the data. Is there a way that we can automatically learn that structure? That is what is at the heart of machine learning.

-- Andrew Ng

- Said another way, we want to discover the *Data Generating Distribution* that underlies the data that we observe. This is the function that we want to learn.

- Moreover, we care about primarily about the generalization accuracy of our model (function)
  - *Accuracy on examples we have not yet seen (BTW, how is this possible?)*
  - as opposed the accuracy on the training set (note: overfitting)

Same Thing Said In Cartoon Form



Reinforcement Learning: Learn from interaction with environment

So When Would We Use Machine Learning?

- When patterns exists in our data
  - Even if we don't know what they are
    - Or perhaps especially when we don't know what they are
- We can not pin down the functional relationships mathematically
  - Else we would just code up the algorithm
- When we have lots of (unlabeled) data
  - Labeled training sets harder to come by
  - Data is of high-dimension
    - High dimension "features"
    - For example, sensor data
  - Want to "discover" lower-dimension representations
    - Dimension reduction
- Aside: Machine Learning is heavily focused on implementability
  - Frequently using well know numerical optimization techniques
  - Lots of open source code available
    - All kinds of open source frame works, e.g., https://www.tensorflow.org/ or http://torch.ch/
    - Well-worn libraries http://scikit-learn.org/stable/ (many others)
    - Languages (e.g., octave: https://www.gnu.org/software/octave/)

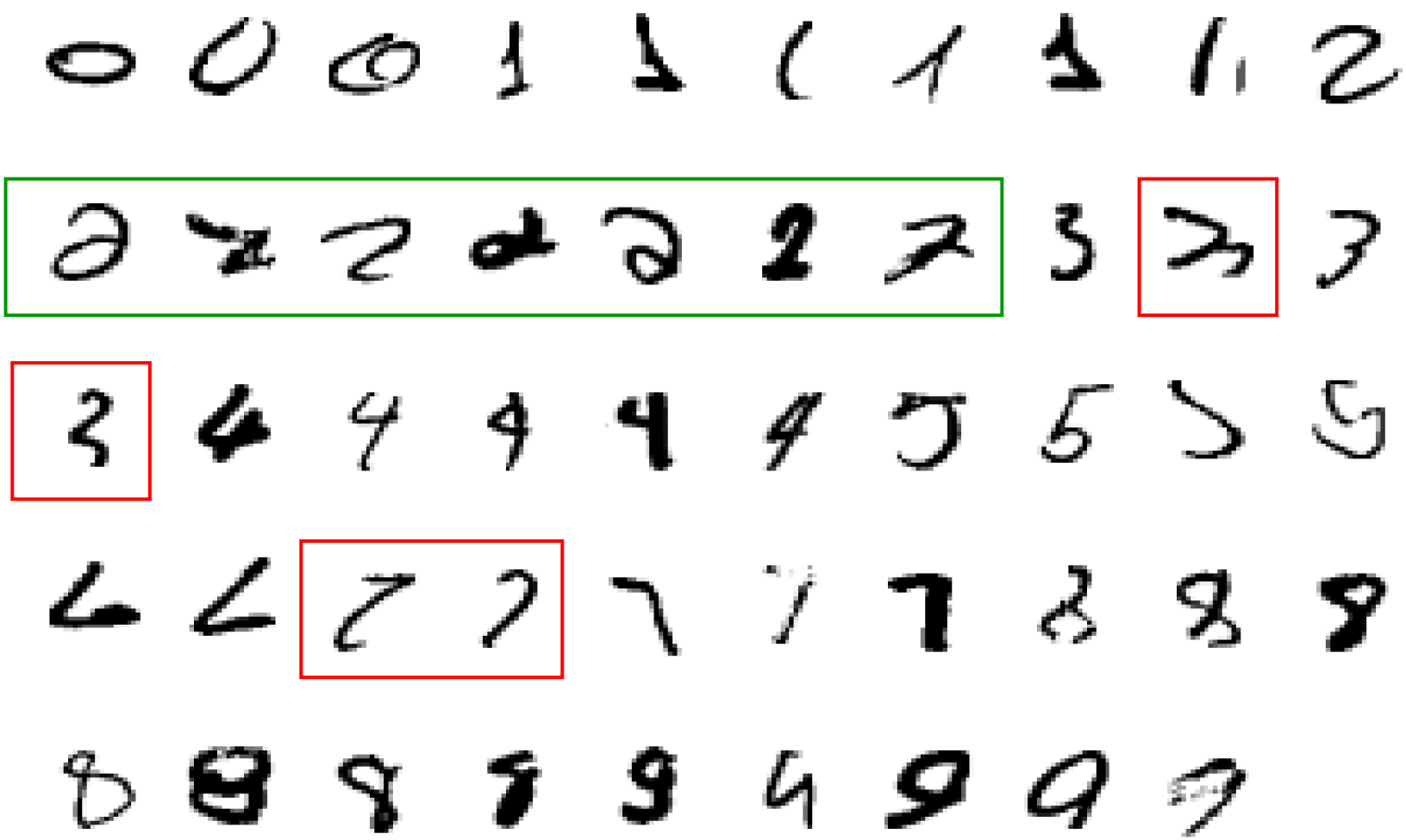
Why Machine Learning is Hard?

You See

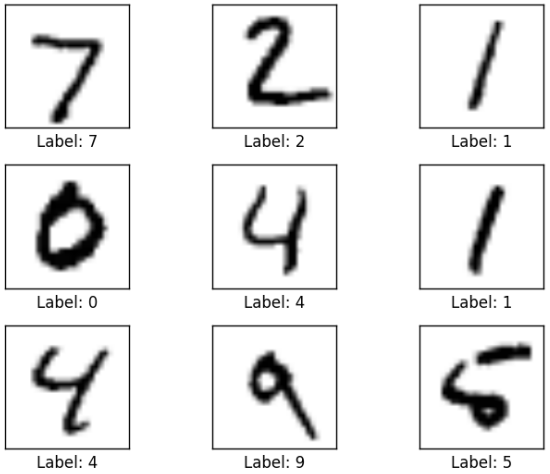
Your ML Algorithm Sees
A Bunch of Bits

Why Machine Learning Is Hard, Redux

What is a “2”?



BTW, How Hard Is It *These Days* To Write Code That Recognizes Handwritten Digits?



MNIST: 28x28 Grayscale Handwritten Digit Dataset

- Training set: 55000 images
- Test set: 10000 images
- Validation set: 5000 images

Features here are pixels (28x28 = 784)

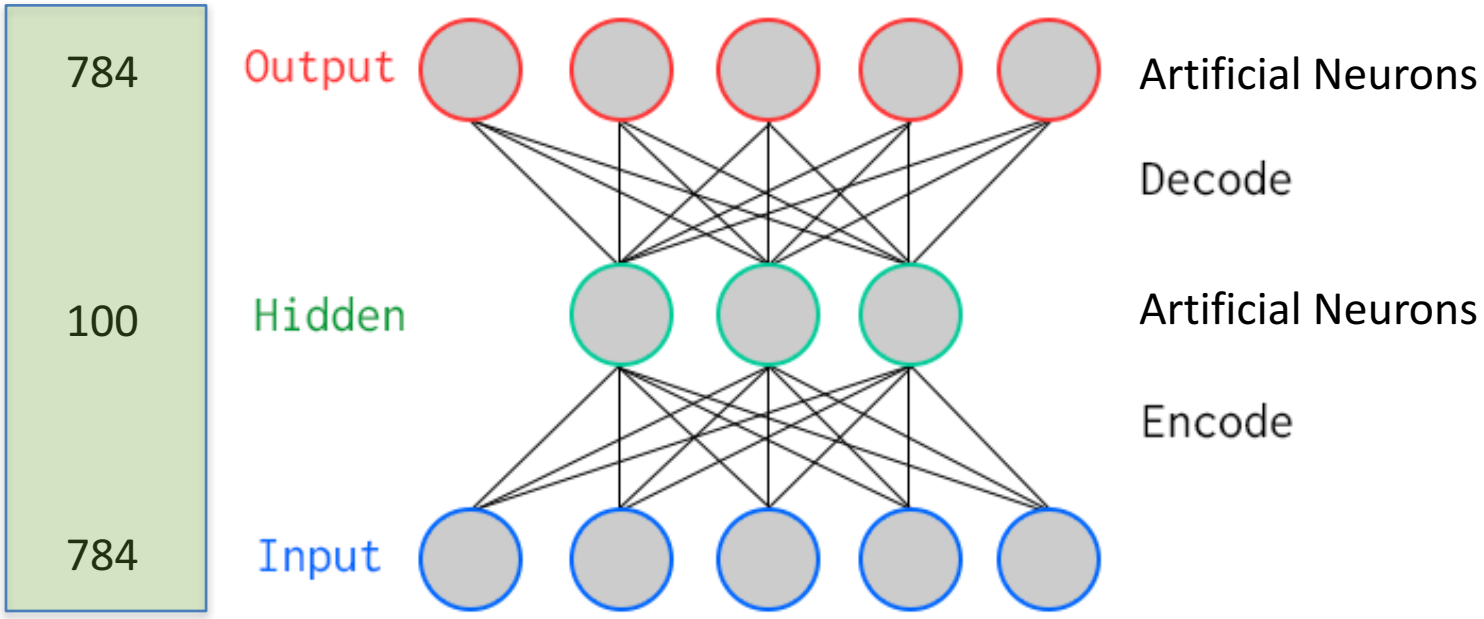
Can We Analyze Network Flow (or other) Data In A Similar Fashion?

Src IP	Dst IP	Appln	Src Port	Dst Port	Protocol	DSCP	TCP FLAGS	Flow Rate	Traffic	Packets	NextHop	FNF NbarApp
192.168.10.	192.168.13.1	compressnet	2	169	TCP	AF12	UAP SF	0 Kbps	1.0 KB	2	15.80.39.28	-
192.168.10.	192.168.13.1	compressnet	2	564	TCP	AF12	UAP SF	0 Kbps	1.0 KB	2	190.23.69.213	-
192.168.10.	192.168.13.1	compressnet	2	741	TCP	AF12	UAP SF	0 Kbps	1.0 KB	2	95.55.61.159	-
192.168.10.	192.168.13.1	compressnet	281	2	TCP	AF12	UAP SF	1.0 Kbps	1.0 KB	2	223.191.78.79	-
192.168.10.	192.168.13.1	compressnet	165	3	TCP	AF12	UAP SF	0 Kbps	1.0 KB	2	64.15.70.93	-
192.168.10.	192.168.13.1	compressnet	424	3	TCP	AF12	UAP SF	0 Kbps	1.0 KB	2	149.191.44.148	-
192.168.10.	192.168.13.1	compressnet	822	3	TCP	AF12	UAP SF	0 Kbps	1.0 KB	2	73.7.175.22	-
192.168.10.	192.168.13.1	rje	800	5	TCP	AF12	UAP SF	0 Kbps	1.0 KB	2	74.157.168.220	-
192.168.10.	192.168.13.1	discard	9	714	TCP	AF12	UAP SF	0 Kbps	1.0 KB	2	1.209.56.6	-
192.168.10.	192.168.13.1	daytime	13	252	TCP	AF12	UAP SF	0 Kbps	1.0 KB	2	232.237.195.100	-
192.168.10.	192.168.13.1	daytime	960	13	TCP	AF12	UAP SF	0 Kbps	1.0 KB	2	22.88.95.248	-
192.168.10.	192.168.13.1	msp	18	116	TCP	AF12	UAP SF	1.0 Kbps	1.0 KB	2	81.203.252.131	-
192.168.10.	192.168.13.1	msp	18	735	TCP	AF12	UAP SF	0 Kbps	1.0 KB	2	18.50.17.126	-
192.168.10.	192.168.13.1	ftp-data	20	513	TCP	AF12	UAP SF	0 Kbps	1.0 KB	2	240.7.195.4	-

Features here are various counters

One Way To Learn To Recognize MNIST: Use An Autoencoder

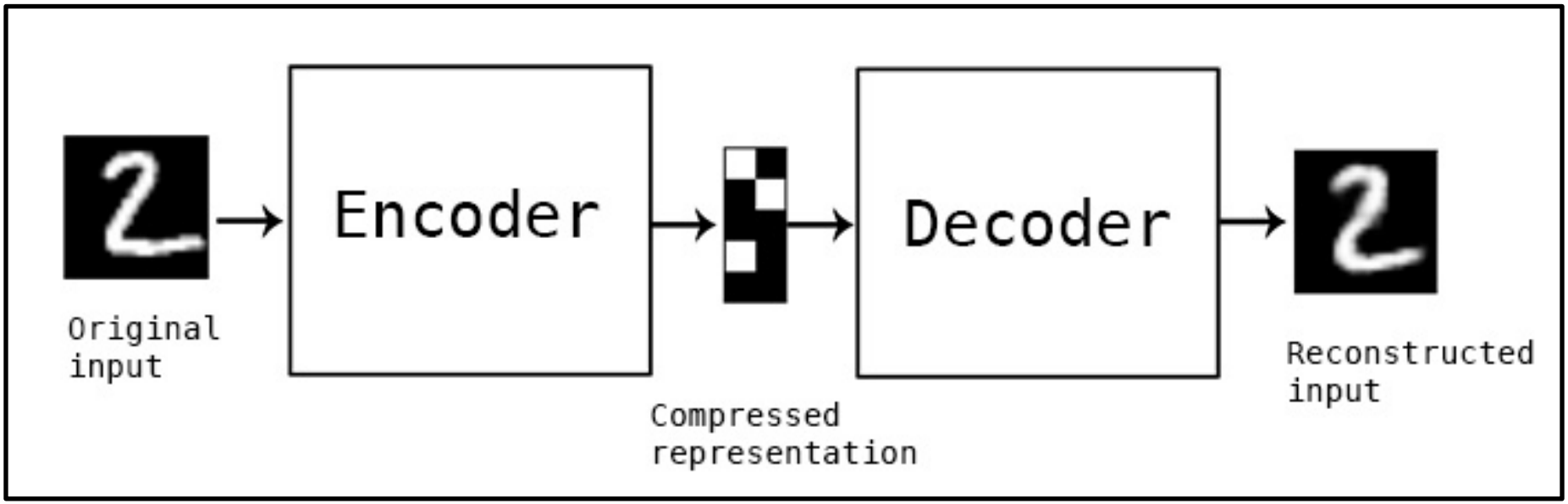
Special Kind of Neural Network



- Key Characteristic: Hidden layer has fewer units than input/output → Compression
- Goal: Minimize reconstruction (decode) error
  - How to define error (loss, cost)?
  - Binary classification: Threshold reconstruction error → normal/abnormal
- Unsupervised learning

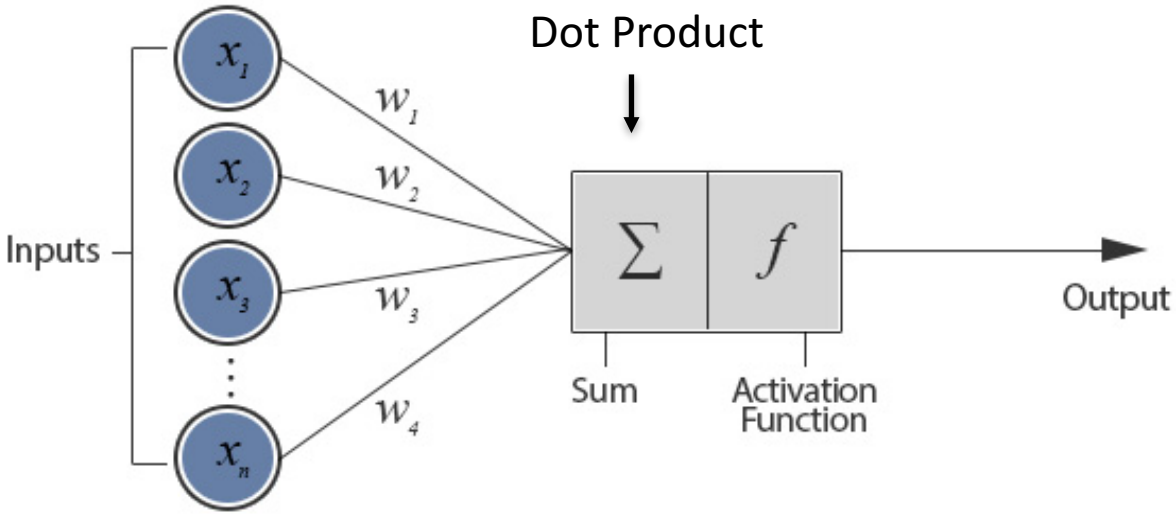
Another Way To Look At This

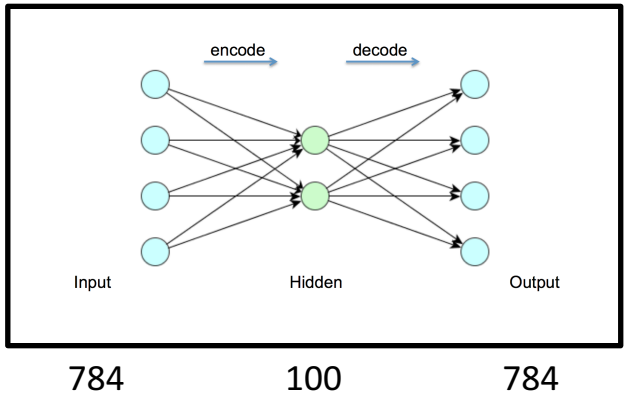
Minimizing Reconstruction Error During Training



We can use autoencoders do anomaly detection by bounding reconstruction error. That is, if an new example has reconstruction error greater than some hyper-parameter, we declare a potential anomaly. Importantly AEs can capture the covariance between features, but have many of the classic AD challenges such as needing baselines, etc.

But First: What Does A Single Neuron Do?



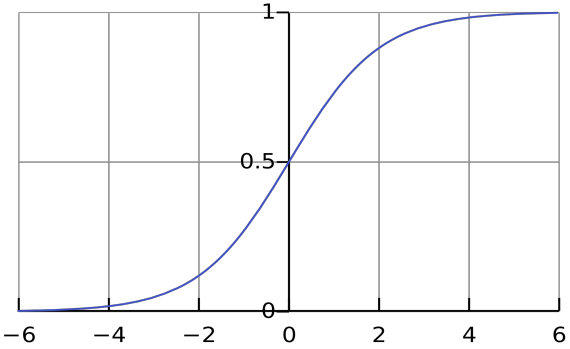


BTW, how many parameters?

784x100 + 100x784 = 156800

$$\boldsymbol{W}^T\boldsymbol{x} = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \sum_{i=1}^{n} w_i x_i = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$f(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}) = \sigma(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}) = \frac{1}{1 + e^{-(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b})}}$$



All Cool, But How Does The Autoencoder Actually Work?

- Encoder $h^{(t)} = f_\theta(x^{(t)}), \{x^{(1)}, \ldots, x^{(T)}\}$

Where h is feature vector or representation or code computed from x

- Decoder

maps from feature space back into input space, producing a reconstruction

$$r = g_\theta(h)$$

attempting to incur the lowest possible reconstruction error $L(x, r)$.

Good generalization means low reconstruction error at test examples, while having high reconstruction error for most other x configurations

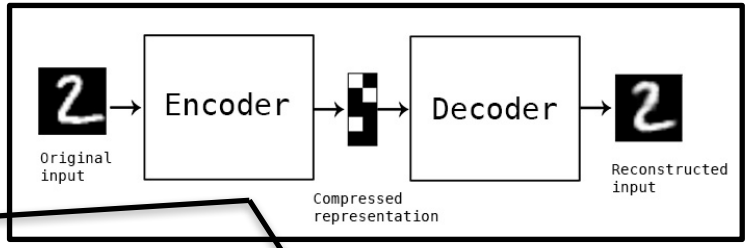
$$h^{(i)} = f_{\theta_e}(x^{(i)}) = \sigma(\theta_e^{\mathrm{T}} x^{(i)} + b_e^{(i)})$$
$$r^{(i)} = g_{\theta_r}(h^{(i)}) = \sigma(\theta_r^{\mathrm{T}} h^{(i)} + b_r^{(i)})$$

$$L(x, r) = \frac{1}{T} \sum_{i=1}^{T} (x^{(i)} - r^{(i)})^2$$

How Hard To Code This Up In Tensorflow?

```
#
#       encoder/decoder
#
#       try tf.nn.sigmoid, tf.nn.relu, etc for nonlinearity
#       nonlinearity=False means transfer function (aka activation funtion)
#       g(x) = x
#
def encoder(x, nonlinearity=False):
    code = tf.add(tf.matmul(x, weights['encoder']), biases['encoder'])
    if nonlinearity:
        code = nonlinearity(code)
    return code

def decoder(code, nonlinearity=False):
    reconstruction = tf.add(tf.matmul(code, weights['decoder']), biases['decoder'])
    if nonlinearity:
        reconstruction = nonlinearity(reconstruction)
    return reconstruction
#
#       get the encoding and decoding operations
#
#       relu seems less efficient here
#
#
#       first encode
#
encoder_op = encoder(X,tf.nn.sigmoid)
#
#       then decode
#
decoder_op = decoder(encoder_op,tf.nn.sigmoid)
#
#       decoder_op is our predicted value (y_pred)
#
y_pred = decoder_op
#
#       y_true is the input X
#
y_true = X
#
reg_losses   = tf.get_collection(tf.GraphKeys.REGULARIZATION_LOSSES)
reg_constant = 0.01
#
if (USE_REGULARIZER):
        error = tf.add(tf.reduce_mean(tf.square(tf.sub(y_true,y_pred))),
                       tf.mul(reg_constant,tf.reduce_sum(reg_losses)))
else:
        error = tf.reduce_mean(tf.square(tf.sub(y_true,y_pred)))
```



$$h^{(i)} = f_{\theta_e}(x^{(i)}) = \sigma(\theta_e^{\mathrm{T}} x^{(i)} + b_e^{(i)})$$

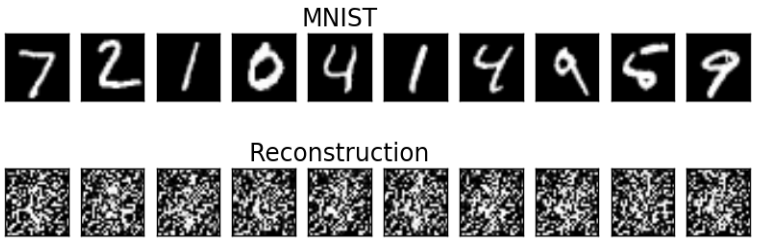
$$r^{(i)} = g_{\theta_r}(h^{(i)}) = \sigma(\theta_r^{\mathrm{T}} h^{(i)} + b_r^{(i)})$$

$$L(x, r) = \frac{1}{T} \sum_{i=1}^{T} (x^{(i)} - r^{(i)})^2$$

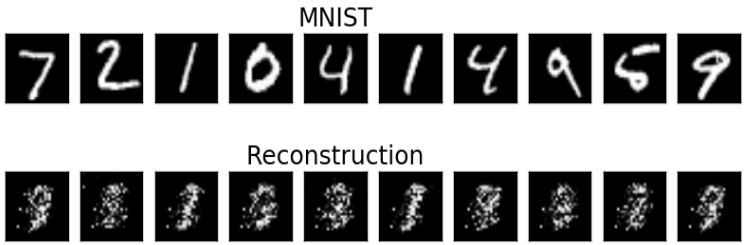
https://github.com/davidmeyer/ml/blob/master/tensorflow/autoencoder.py

Autoencoder Output

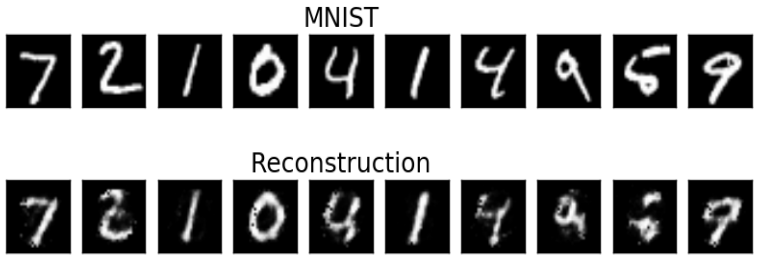
1 example



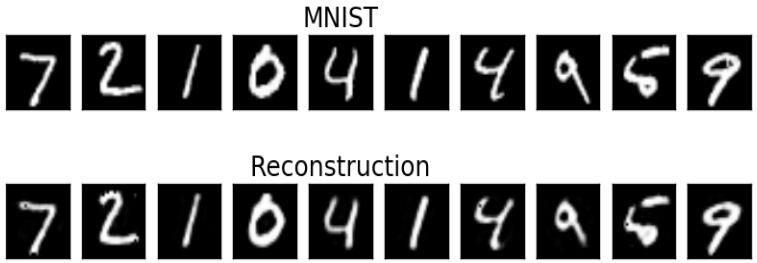
10 examples



100 examples



1000 examples

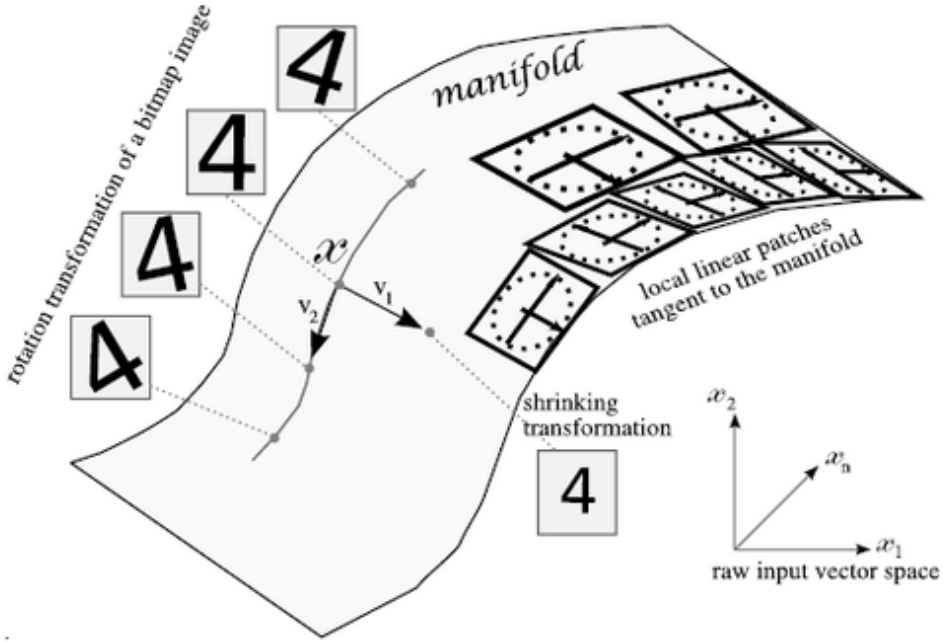


After training, the AE gets low *reconstruction error* on digits from MNIST and high reconstruction error on everything else: It has learned to recognize MNIST

$$L(x, r) = \frac{1}{T} \sum_{i=1}^{T} (x^{(i)} - r^{(i)})^2$$

What Is Really Going On Here?
(Manifold Hypothesis)

- X is a high dimensional vector ($28^2 = 784$)
- Data concentrated around a lower dimensional manifold



- Hope to find representation *R* of that manifold

Graphic courtesy http://www.deeplearningbook.org

Many Beautiful Mysteries Remain

- Back-propagation
  - Gradient-based optimization with gradients computed by back-prop
  - Why is such a simple algorithm so powerful?
  - Optimization an active area of research
  - Many new techniques , e.g., Layer Normalization
  - https://arxiv.org/pdf/1607.06450v1.pdf
- Neural Nets
  - What are the units (artificial neurons) actually doing?
  - https://arxiv.org/pdf/1509.06321.pdf
- Adversarial Images
  - Generative Adversarial Nets (GANs)
  - https://arxiv.org/abs/1412.6572
- Why does deep and cheap learning work so well?
  - Physics perspective
  - http://arxiv.org/pdf/1608.08225v1.pdf
- …

- BTW, how many 28x28 grayscale images are there?
- Say there are 8 bits of grayscale → 256^{784} possible images
- How can such a simple autoencoder recognize them?

So What Kind of Problems Can We Solve With Machine Learning?

- Pattern Recognition
  - Facial identities or facial expressions
  - Handwritten or spoken words (e.g., Siri)
  - Medical images
  - Sensor Data/IoT
  - Network traffic and performance
- Optimization
  - Many parameters have “hidden” relationships that can be the basis of optimization
- Pattern Generation
  - Generating images or motion sequences
- Anomaly Detection
  - Unusual patterns in the telemetry from physical and/or virtual plants (e.g., data centers)
  - Unusual sequences of credit card transactions
  - Unusual patterns of sensor data from a nuclear power plant
    - or unusual sound in your car engine or …
- Prediction
  - Future stock prices or currency exchange rates
  - Network events
  - …

Note that ML is a form of Induction
(Supervised Learning Setting)

- Given examples of a function *(x, f(x))*
  - *Don't explicitly know f*
    - Rather, trying to learn *f* from the data
  - *Labeled* training data set (i.e., the *f(x)*'s)
  - Training set will be noisy, e.g., *(x, (f(x) + ε))*
  - Notation: $(x_i, f(x_i))$ denoted $(x^{(i)}, y^{(i)})$
  - $y^{(i)}$ sometimes called t_i (t for "target")

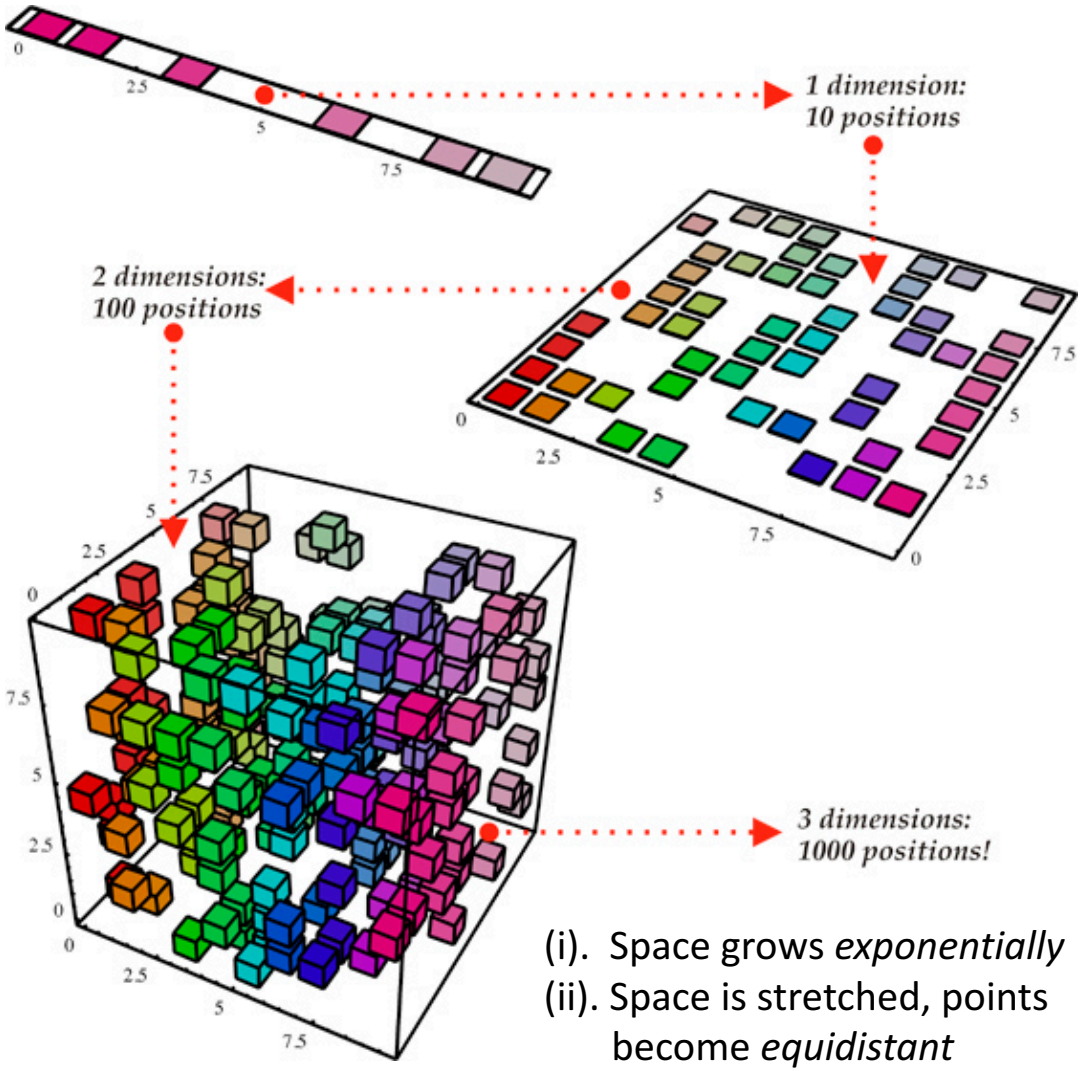
- Predict function ***f(x)*** for new examples *x*
  - Discrimination/Prediction (Regression): *f(x)* continuous
  - Classification*: f(x)* discrete
  - Estimation*: f(x)* = P(Y = c|*x*) for some class c

BTW, How Can Machine Learning Possibly Work?

- We want to build statistical models that **generalize to unseen cases**
- What assumptions do we need to do this (essentially predict the future)?
- 4 main "prior" assumptions are (at least) required
  - Smoothness
  - Manifold Hypothesis
  - Distributed Representation/Compositionality
    - Compositionality is useful to describe the world around us efficiently → distributed representations (features) are meaningful by themselves.
    - Non-distributed → # of distinguishable regions linear in # of parameters
    - Distributed → # of distinguishable regions grows almost exponentially in # of parameters
      - Each parameter influences many regions, not just local neighbors
    - Want to generalize non-locally to never-seen regions → essentially ***exponential gain***
  - Shared Underlying Explanatory Factors
    - The assumption here is that there are shared underlying explanatory factors, in particular between p(x) (prior distribution) and p(Y|x) (posterior distribution). ***Disentangling*** these factors is in part what machine learning is about.
- Before this, however: What is the problem in the first place?

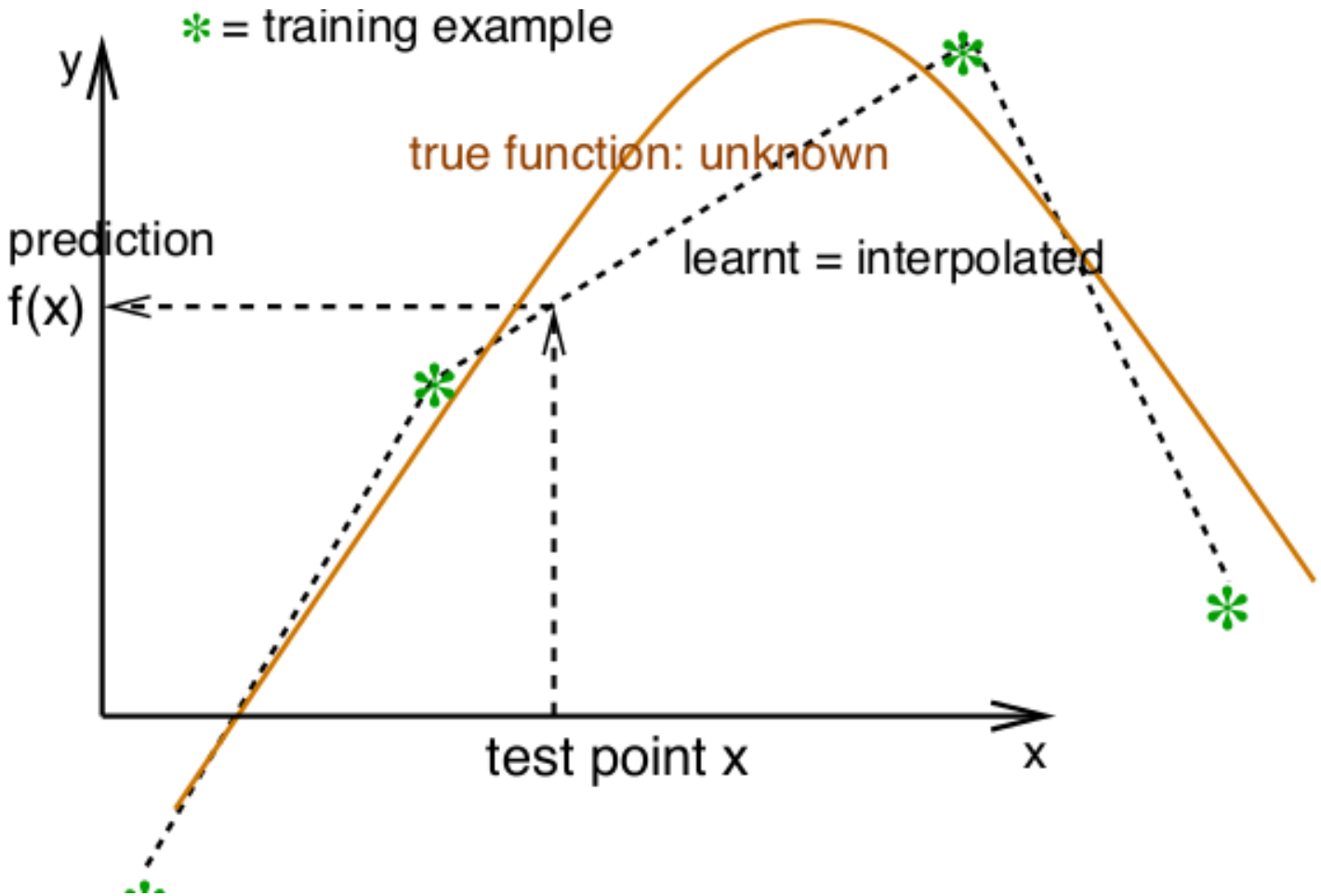
Why ML Is Hard
The Curse Of Dimensionality

- To generalize locally, you need representative examples from all relevant variations (and there are an *exponential* number of them)!
- Classical Solution: Hope for a smooth enough target function, or make it smooth by handcrafting good features or kernels
- *Smooth?*



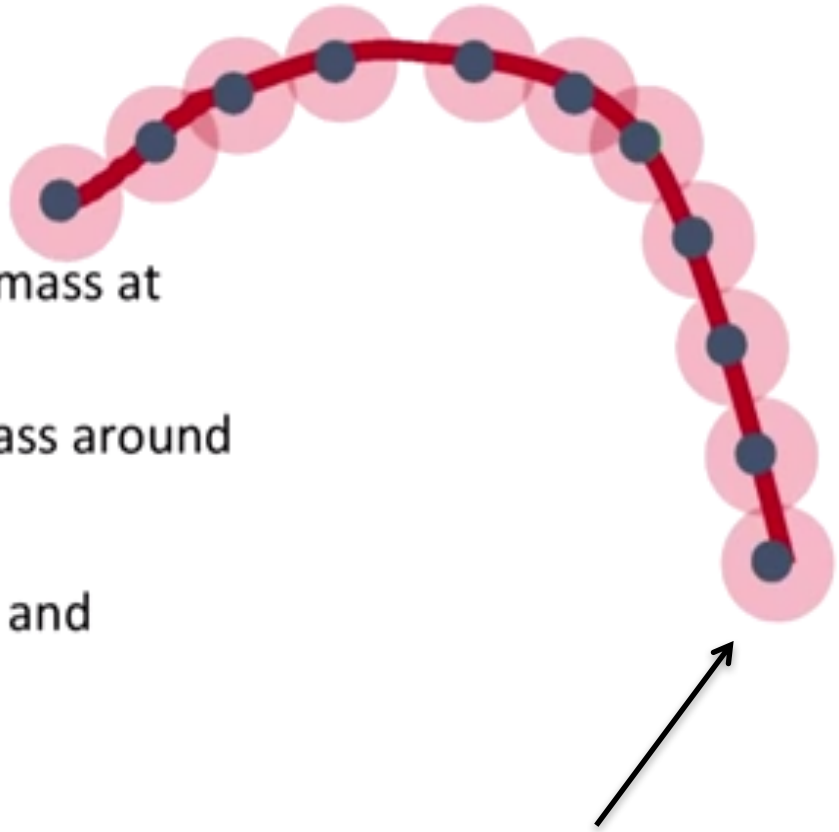
(i). Space grows *exponentially*
(ii). Space is stretched, points become *equidistant*

So What Is Smoothness?



Smoothness → If **x** is geometrically close to **x'** then **f(x) ≈ f(x')**

Smoothness, basically…

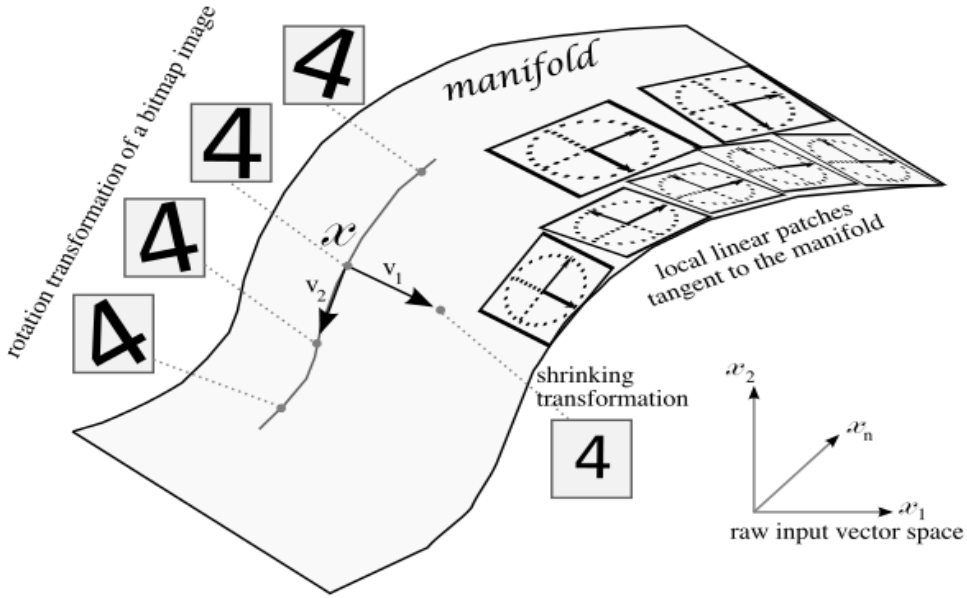


- Empirical distribution: mass at training examples
- Smoothness: spread mass around
- Insufficient
- Guess some 'structure' and generalize accordingly

Probability mass **P(Y=c|X;θ*)***

This is where the *Manifold Hypothesis* comes in…

Manifold Hypothesis

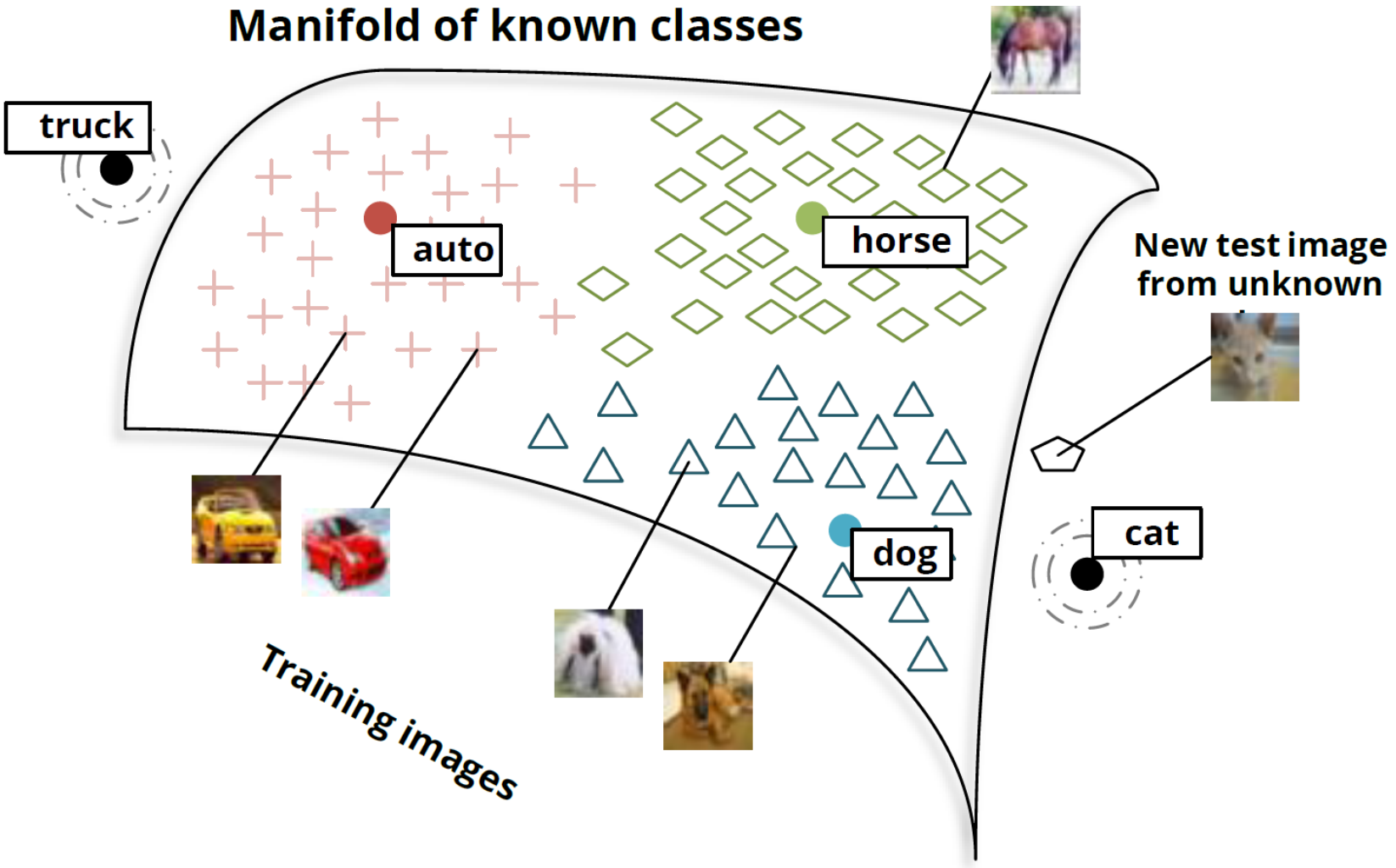


The Manifold Hypothesis states that ***natural data*** forms lower dimensional manifolds in its embedding space. Why should this be? Well, it seems that there are both theoretical and experimental reasons to suspect that the Manifold Hypothesis is true.

So if you believe that the MH is true, then the task of a machine learning classification algorithm is fundamentally to separate a bunch of tangled up manifolds.

However, consider Adversarial Images/Generative Adversarial Networks

Another View: Manifolds and Classes



Ok, Great. What Then Is Learning?

- Learning is a procedure that consists of estimating the model parameters so that the learned model (algorithm) can perform a specific task
  - In Artificial Neural Networks, these parameters are the *weight matrix* ($w_{i,j}$'s)

- 3+ types of learning considered here
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning
  - Semi-supervised Learning

- Supervised Learning
  - Present the algorithm with a set of inputs and their corresponding outputs
  - See how closely the actual outputs match the desired ones
    - Note generalization error (bias, variance)
  - Iteratively modify the parameters to better approximate the desired outputs (gradient descent)

- Unsupervised Learning
  - Algorithm learns internal representations and important features

- Reinforcement Learning
  - Learn from interacting with the environment ("trial and error")
  - More on this later

- So let's take a closer look at these learning types

Supervised Learning

- The desired response (function) of given inputs is well known
  - You are given the "answer" (label) in the training set
  - Training data is set of $(x^{(i)},y^{(i)})$ pairs, $x^{(i)}$ is the input example, $y^{(i)}$ is the label
  - Task is typically to predict the label y for new x ($y = f(x)$)

- There are many 10s (if not 100s or 1000s) of supervised learning learning algorithms
  - These include: Artificial Neural Networks, Decision Trees, Ensembles (Bagging, Boosting, Random Forests, …), k-NN, Linear Regression, Naive Bayes, Logistic Regression (and other CRFs), Support Vector Machines (and other Large Margin Classifiers), …
  - Focus on Artificial Neural Networks (ANNs) here

- The Google Data Center PUE example we will look at later uses supervised learning on a deep neural network (if time)

Unsupervised Learning

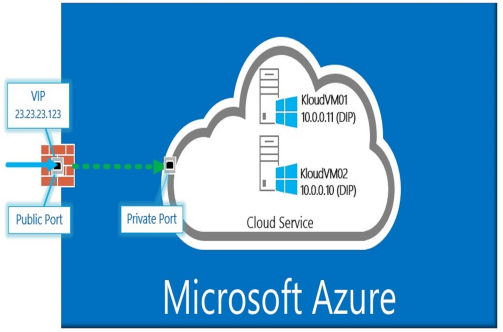
- Basic idea: Discover unknown structure in input data
- Data clustering and dimension reduction
  - More generally: find the relationships/structure in the data set
- No need for labeled data
  - The network itself finds the correlations in the data
- Learning algorithms include (again, many algorithms)
  - K-Means Clustering
  - Auto-encoders/deep neural networks
  - Restricted Boltzmann Machines
    - Hopfield Networks
  - Sparse Encoders
  - …

Agenda

- Who Am I?
- Level Set: What Is Machine Learning?
- What is all the (Machine Learning) Excitement About?
- Integrated Approaches
- Histograms, LDA, and Neural Networks
- Machine Learning Excitement Redux: Beyond Static Learning
- Technical explanations/code
  - https://github.com/davidmeyer/ml

What is all the ML Excitement About?

ML Applications You Likely Interact with Everyday



Why this is relevant: Compute, Storage, Networking, Security and Energy (CSNSE) use cases will all use similar technology (deep nets, …)

For example

techemergence

Search

TECHNOLOGY INTERVIEWS BUSINESS RESEARCH SUBSCRIBE ABOUT

AT&T Predicts Future, Saves Service with Machine Learning – A Conversation with Dr. Mazin Gilbert

DANIEL FAGGELLA × FEBRUARY 28, 2016

"Machine learning is revolutionizing the world as we know it, both in and out of the digital realms, and is predicted to expand to a $2 trillion market by 2025 (as reported by Julie Bort in Business Insider)."

5G Networks


CogNet
Home
Project
CogNet in 5GPPP
Dissemination
Events
News
Contact
Building an Intelligent System
of Insights and Action
for 5G Network Management
COGNET

Think Object Recognition is Impressive?

LOGIN | SIGN UP

EMOTIENT

Solutions Products Technology About Blog

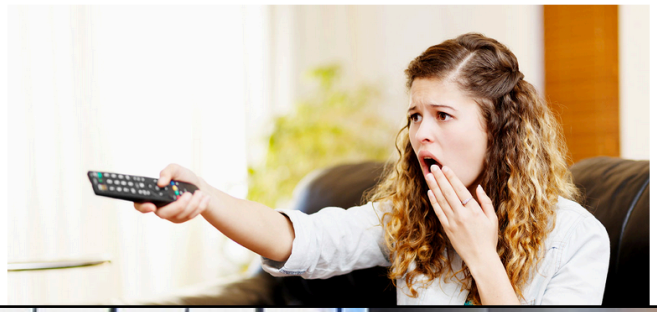
Solutions: Media Testing

Shifting Media Tests from Art to Science

Will consumers enjoy and recommend this movie? Is this program actually funny or engaging, and to whom? Will this news program segment create viewer loyalty?

Programming for digital, cinematic, broadcast or other media is renowned for being a mix of art and science, marked by the occasional hit making up for numerous flops. What if you could move the needle farther in the direction of science, reduce your risk, and increase hit your rate?

That is the promise of measuring the direct emotional respon programming before you spend millions on launch and risk u advertisers. Emotient Analytics lets you directly measure cust attention, at which points they are engaged with your progra emotions it elicits.



Solutions Products Technology About Blog

Emotient Analytics Free Trial

Lip Reading?

LIPREADING WITH LONG SHORT-TERM MEMORY

Michael Wand, Jan Koutník, Jürgen Schmidhuber

The Swiss AI Lab IDSIA, USI & SUPSI

ABSTRACT

Lipreading, i.e. speech recognition from visual-only recordings of a speaker's face, can be achieved with a processing pipeline based solely on neural networks, yielding significantly better accuracy than conventional methods. Feed-forward and recurrent neural network layers (namely Long Short-Term Memory; LSTM) are stacked to form a single structure which is trained by back-propagating error gradients through all the layers. The performance of such a stacked network was experimentally evaluated and compared to a standard Support Vector Machine classifier using conventional computer vision features (Eigenlips and Histograms of Oriented Gradients). The evaluation was performed on data from 19 speakers of the publicly available GRID corpus. With 51 different words to classify, we report a best word accuracy on held-out evaluation speakers of 79.6% using the end-to-end neural network-based solution (11.6% improvement over the best feature-based solution evaluated).

Index Terms— Lipreading, Long Short-Term Memory, Recurrent Neural Networks, Image Recognition

1. INTRODUCTION

It is well-known that humans understand speech not only by listening, but also by taking visual cues into account [1]. Hearing-impaired persons are in fact able to comprehend hu-

Local Binary Patterns [7]. Classification is frequently done with Support Vector Machines (SVMs), e.g. [7], or Hidden Markov Models (HMMs), e.g. [4, 5, 8, 9].

Our aim is to replace the complete visual speech recognition pipeline with a compact neural network architecture. Neural networks (NNs) have become increasingly popular in conventional speech recognition, first as feature extractors in an HMM-based architecture [10–12], more recently replacing the entire processing chain [13]. For the latter, the *Long Short Term Memory* (LSTM; [14]) architecture is typically used. Consequently, our approach to the lipreading problem uses a NN that chains feed-forward layers and LSTM layers, described in detail in subsection 4.2. Manual feature extraction is no longer required. The NN inputs are now the raw mouth images, as is common in modern computer vision tasks, but stands in stark contrast e.g. to [5, 7].

2. RELATED WORK

Lipreading has been used as a complementary modality for speech recognition from noisy audio data [2, 15], as well as for purely visual speech recognition [3, 16, 17]. The latter gives rise to a *Silent Speech interface*, which is defined as a system "enabling speech communication to take place when an audible acoustic signal is unavailable" [18]. Silent Speech technology has a large number of applications: It allows per-

Real-Time Language Translation

One More

Venture Capital Dispatch

An inside look from VentureWire at high-tech startups and their investors.

INVESTORS/RAISING CAPITAL
Norwest Venture Partners Closes Its Third $1.2 Billion Fund

COMPANY FUNDING
Internet-of-Things Security Company ForeScout Valued at $1 Billion

9:00 am ET
Jan 28, 2016 COMPANY FUNDING

Bay Labs Launches to Bring Artificial Intelligence to Ultrasounds

PREVIOUS
Why Southeast Asia's GrabTaxi Is Removing 'Taxi' From Its Name

NEXT
Insurify Raises $2 Million for Virtual Insurance Agent

ARTICLE | COMMENTS

BAY LABS | ELEVEN TWO CAPITAL | KEVIN GOODWIN | KHOSLA VENTURES

Email | Print | 169

By CAT ZAKRZEWSKI CONNECT

SEARCH VENTURE CAPITAL DISPATCH GO

About Venture Capital Dispatch

Produced by the editors of **Dow Jones VentureWire**, Venture Capital Dispatch tracks the fast-moving developments at the intersection of high-tech innovation and venture capital finance. Featuring the VentureWire reporting team in the Silicon Valley, New York, Boston and Shanghai tech centers, Venture Capital Dispatch provides insight into the newest start-ups and latest trends in venture capital investing. Write us at **VCdispatch@dowjones.com**. For more information on Dow Jones products covering venture capital and other financial markets, go to **http://pevc.dowjones.com**.

Follow Venture Capital Dispatch on Twitter

Like Venture Capital Dispatch on Facebook

Venture Capital Dispatch Bureau

Mike Billings

Scott Martin

Why is this all happening now?

- **Before 2006 people thought deep neural networks couldn't be trained**
  - So why now?

- **Theoretical breakthroughs in 2006**
  - Learned how to train deep neural networks
    - Technically: RBMs, Stacked Auto-encoders, sparse coding
  - Nice overview of LBH DL journey: http://chronicle.com/article/The-Believers/190147/

- **Compute**
  - CPUs were 2^20s of times too slow
  - Parallel processing/algorithms
  - GPUs + OpenCL/CUDA
  - FPGAs, custom hardware

- **Datasets**
  - Massive data sets: Google, FB, Baidu, …
  - *Standardized* data sets

2006: The Deep Breakthrough

- Hinton, Osindero & Teh « A Fast Learning Algorithm for Deep Belief Nets », *Neural Computation*, 2006
- Bengio, Lamblin, Popovici, Larochelle « Greedy Layer-Wise Training of Deep Networks », *NIPS'2006*
- Ranzato, Poultney, Chopra, LeCun « Efficient Learning of Sparse Representations with an Energy-Based Model », *NIPS'2006*

- **Alternate view of history?**
  - LBH Nature DL review: http://www.nature.com/nature/journal/v521/n7553/full/nature14539.html
  - Jürgen Schmidhuber's critique : http://people.idsia.ch/~juergen/deep-learning-conspiracy.html
  - LBH rebuttal: http://recode.net/2015/07/15/ai-conspiracy-the-scientists-behind-deep-learning/

Image courtesy Yoshua Bengio

BTW, What Kinds of Custom H/W?

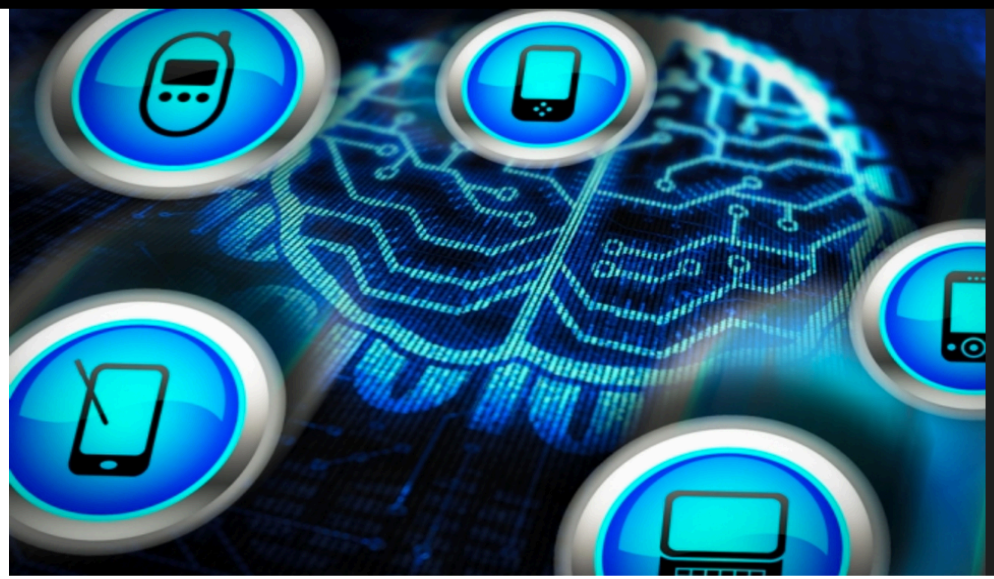
Massachusetts Institute of

NEWS | VIDEO | SOCIAL | FOLLOW MIT

MIT News

ON CAMPUS AND AROUND THE WORLD

Browse or Search



FULL SCREEN

MIT researchers have designed a new chip to implement neural networks. It is 10 times as efficient as a mobile GPU, so it could enable mobile devices to run powerful artificial-intelligence algorithms locally, rather than uploading data to the Internet for processing.

Image: MIT News

Energy-friendly chip can perform powerful artificial-intelligence tasks

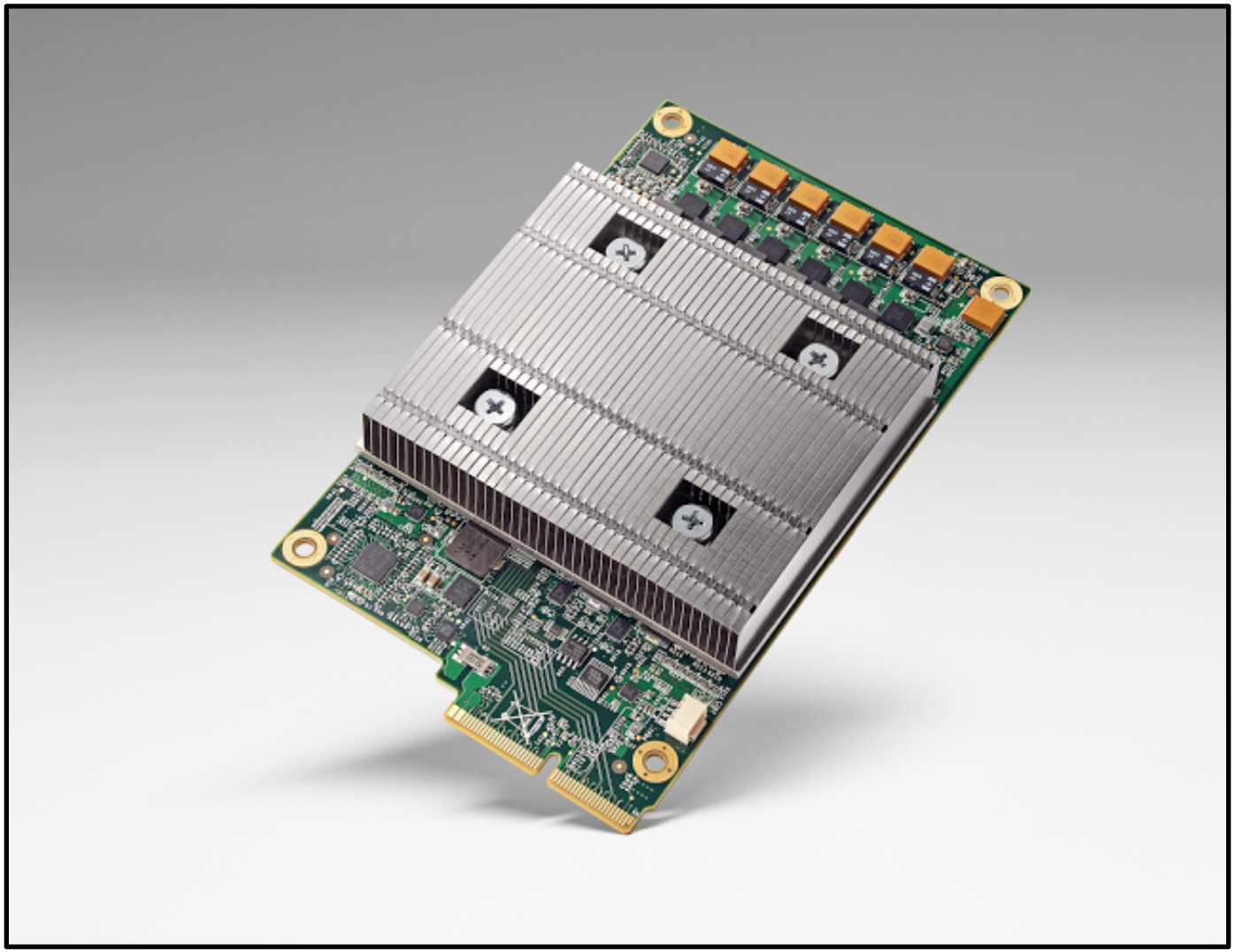
Advance could enable mobile devices to implement “neural networks” modeled on the human brain.

Larry Hardesty | MIT News Office
February 3, 2016

Press Inquiries

RELATED

What Kinds of Custom H/W, cont
Google TPU



What Kinds of Custom H/W, cont

recode TRENDING TOPICS WRITERS PODCASTS EVENTS

BIG DATA INTEL ARTIFICIAL INTELLIGENCE

Intel is paying more than $400 million to buy deep-learning startup Nervana Systems

The chip giant is betting that machine learning is going to be a big deal in the data center.

BY INA FRIED · AUG 9, 2016, 4:00P

TWEET SHARE LINKEDIN

Intel data center chief Diane Bryant, with Nervana co-founders Naveen Rao, Arjun Bansal, Amir Khosrowshahi and Intel vice president Jason Waxman (left to right) Intel

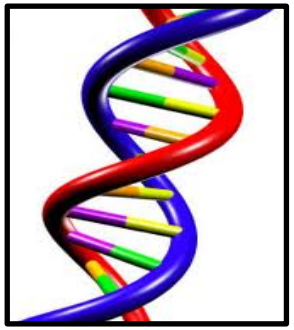
TRENDING

The Google X moonshot factory is struggling to get products out the door

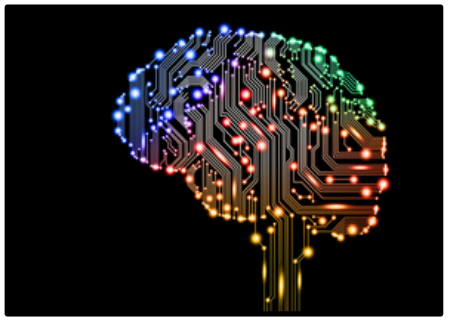
Bruno Mars and Mark Zuckerberg showed up at Spotify CEO Daniel Ek's

Ok, We Know That Machines Are Getting Smarter, But Where Does Knowledge Come From?

Evolution



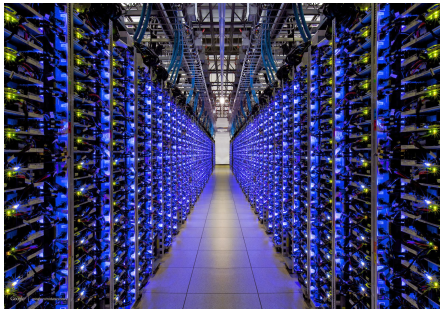
Experience



Many orders of magnitude faster and larger

Culture

Machines



So how can machines discover new knowledge?

How Can Machines Discover New Knowledge?

- **Fill the gaps in existing knowledge**
  - Symbolists
  - Technology: Induction/Inverse Deduction
- **Emulate the brain**
  - Connectionists
  - Technology: Deep neural nets
- **Emulate evolution**
  - Evolutionaries
  - Technology: Genetic Algorithms
- **Systematically reduce uncertainty**
  - Bayesians
  - Technology: Bayesian Inference
- **- Notice similarities between old and new**
  - Analogizers
  - Technology: Kernel machines/Support Vector Machines

These correspond to the 5 major schools of thought in machine learning

https://en.wikipedia.org/wiki/The_Master_Algorithm

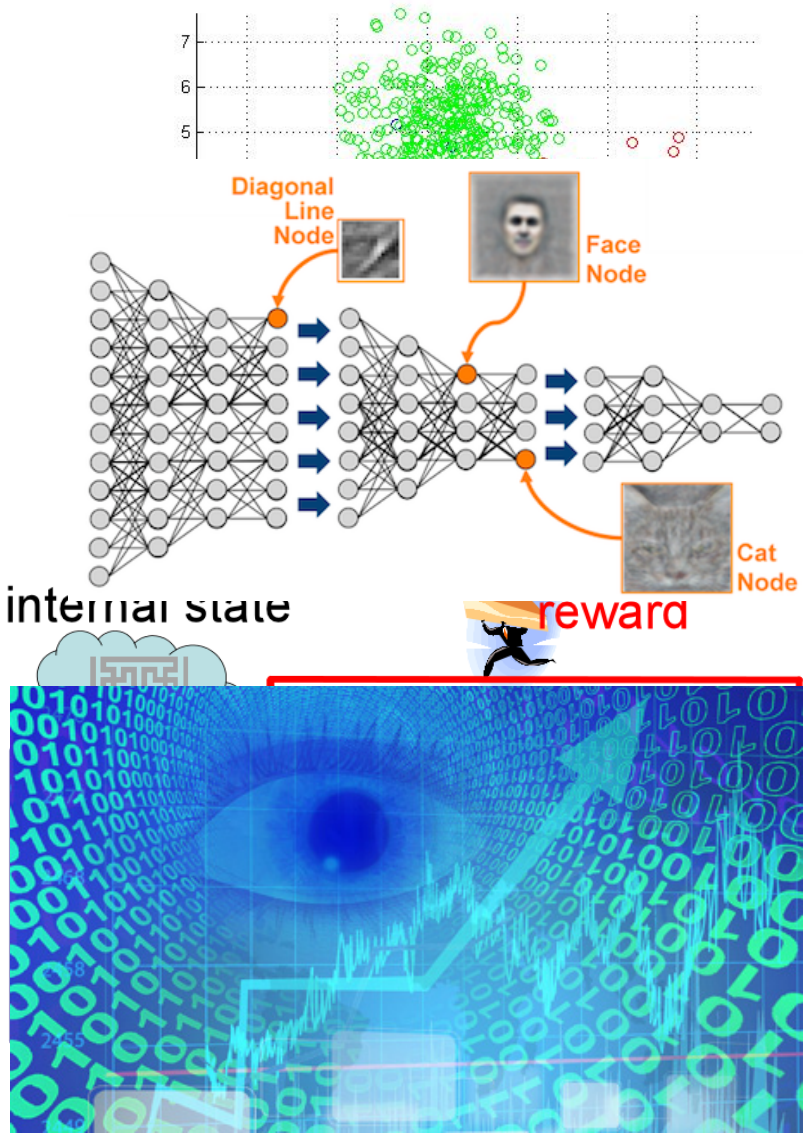
Agenda

- Who Am I?
- Level Set: What Is Machine Learning?
- What is all the (Machine Learning) Excitement About?
- Integrated Approaches
- Histograms, LDA, and Neural Networks
- Machine Learning Excitement Redux: Beyond Static Learning
- Technical explanations/code
  - https://github.com/davidmeyer/ml

Integrated Approaches

Bringing Rigor to ML for Networking

- Clustering
  - Categorical and continuous
    - e.g., Histograms, LDA, K-means, …
  - Most “Machine Learning” systems you see today
  - Anomaly detection
- Deep Neural Networks
  - FF/Recurrent/memory nets (LSTMs, NTMs, ...)
  - Time series/long range dependencies
  - Understand sequences such as network flows
  - Can use autoencoders for anomaly detection too
- Reinforcement Learning
  - Give Machine Learning ***agency***
    - learn feedback control of actions
  - Non-stationary distributions
  - Deep neural networks (value/policy networks)
  - Understand/react in adversarial environments
- Standardized (and public) data sets
  - Required to evaluate techniques
  - Move the field forward
    - e.g. MNIST , ImageNet, …



So What Kinds of Use Might We Work on With PSNC?

- Security/Anomaly Detection
- Site Reliability Engineering
- NFV orchestration and optimization
- New automation tools for DevOps
- Predicting and remediating problems mobile networks
- Network control plane optimization
- Network Gamification
- …

Importantly, we can use deep neural networks to capture intuition from experts in these problem domains

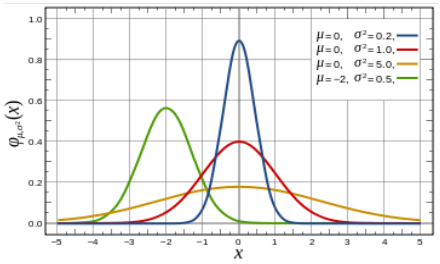
BTW, Anomaly Detection: What and Why

- **It is clear that one of the major challenges we face as a civilization is dealing with deluge of data that are being collected from our networks at global (and beyond) scale**
  - **While at the same time we are "knowledge starved"**
  - **Can't find the needles in an exponentially growing haystack**
  - **Anomaly Detection is one piece of the puzzle**
  - **Machine Learning is a fundamental part of the answer**

- **Key Assumption for Anomaly Detection**
  - **Anomalous events occur relatively infrequently (alternatively: most events normal)**
  - **Second order assumption: Common events follow a Gaussian distribution (likely to be wrong)**

$$f(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

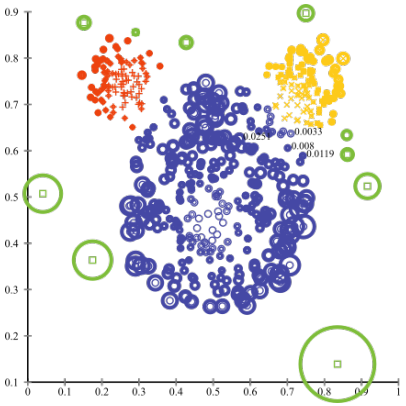
$$Z = \frac{X - \mu}{\sigma}$$

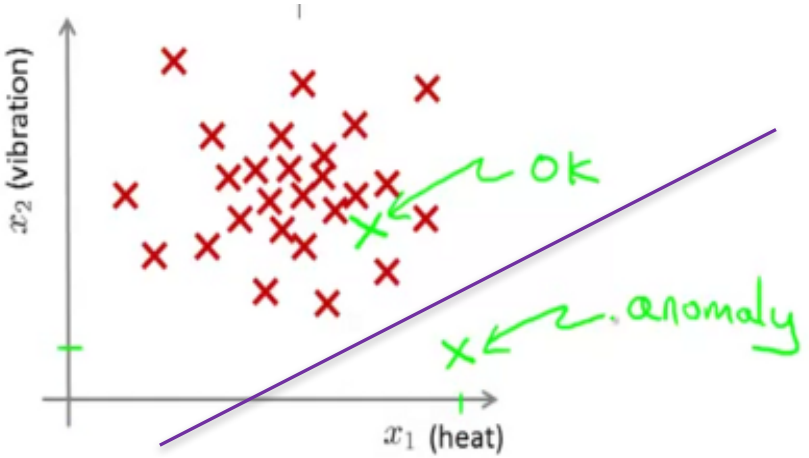


- **What is obvious: When anomalous events do occur, their consequences can be quite serious and often have substantial negative impact on our businesses, security, …**

So What are Anomalies?

- An anomaly is a pattern that does not conform to the expected behaviour
  - How to define expected behaviour?
  - How to find the “outliers”?



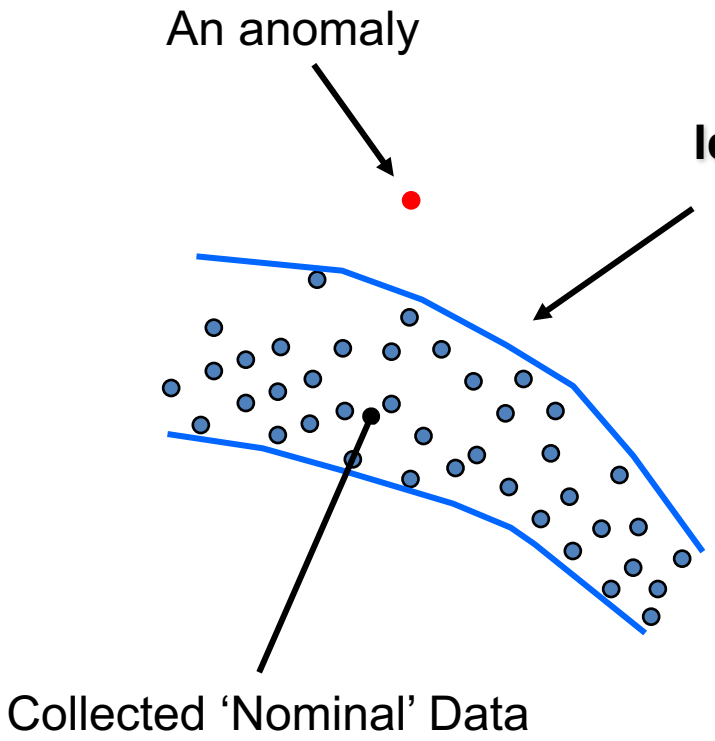


Linear Decision Boundary

- Anomalies translate to significant real life events
  - Cyber intrusions
  - Cyber crime
  - Manufacturing/product defects
  - …

Graphic courtesy Andrew Ng, others

Basic Idea Behind Anomaly Detection



Idea: Assume that a boundary exists and that
- Nominal data is inside the boundary
- Anomalous data is outside the boundary

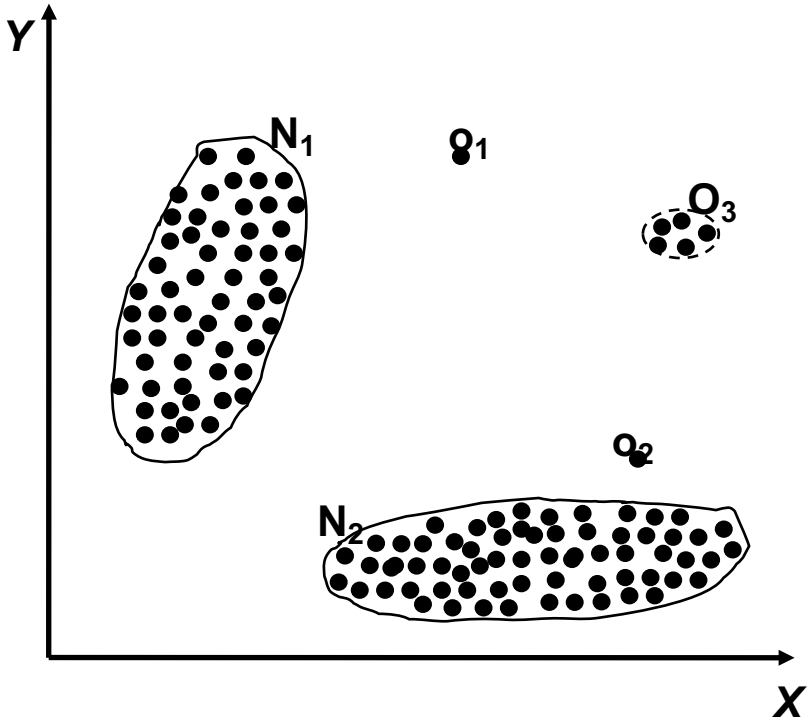
Problem: How to estimate/approximate the boundary?

Problem: What measurement(s) caused the anomaly?

Problem: How far off-nominal is the anomaly/feature?

Simple Example

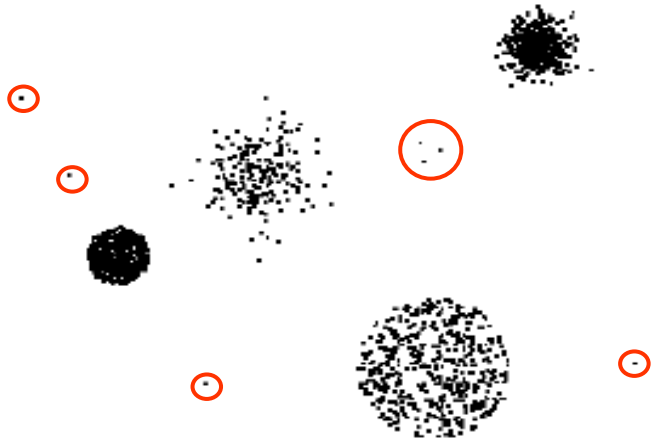
- N_1 and N_2 are regions of normal behaviour
  - Say, normal flows in a network
- Points o_1 and o_2 are anomalies
- Points in region O_3 are anomalies
- Challenge:
  - How to define “normal” regions?
  - How to find the outlier points?
- This is the job of machine learning



Anomaly Detection Schemes

- General Steps
  - Build a profile of the “normal” behavior
    - Profile can be patterns or summary statistics for the overall population
  - Use the “normal” profile to detect anomalies
    - Anomalies are observations whose characteristics differ significantly from the normal profile

- Types of anomaly detection schemes
  - Graphical & Statistical-based
  - Distance-based
  - Model-based
  - FP Mining, K-means, …

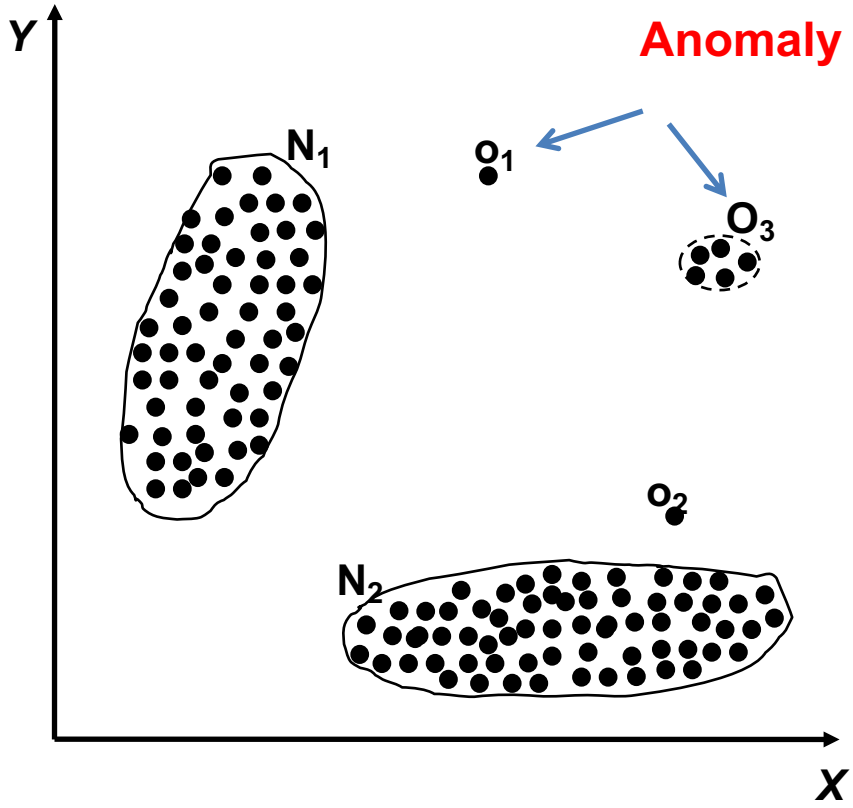


3 Main Types of Anomaly

- Point Anomalies
- Contextual Anomalies
- Collective Anomalies

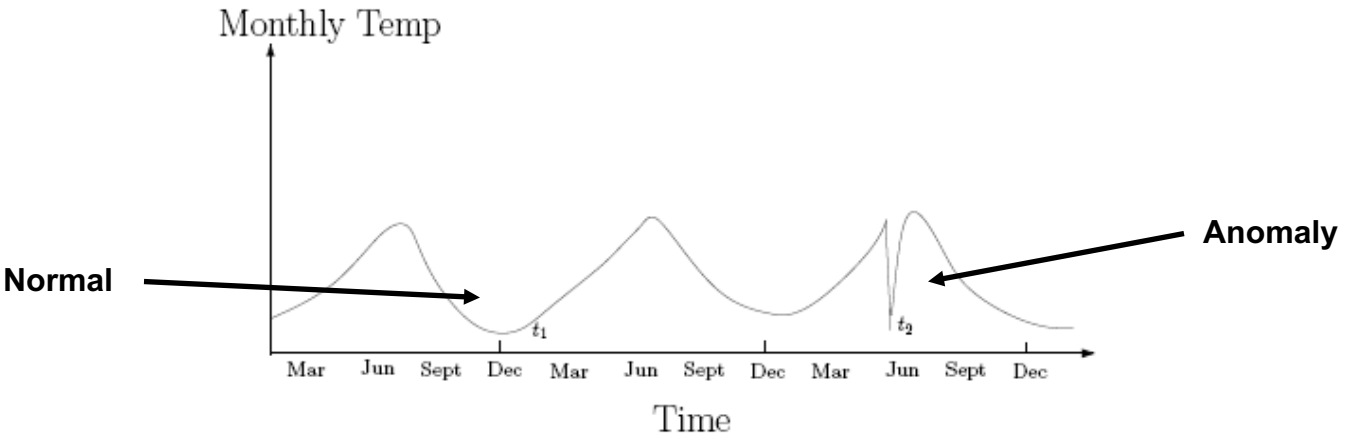
Point Anomalies

- An individual data instance is anomalous if it deviates significantly from the rest of the data set.



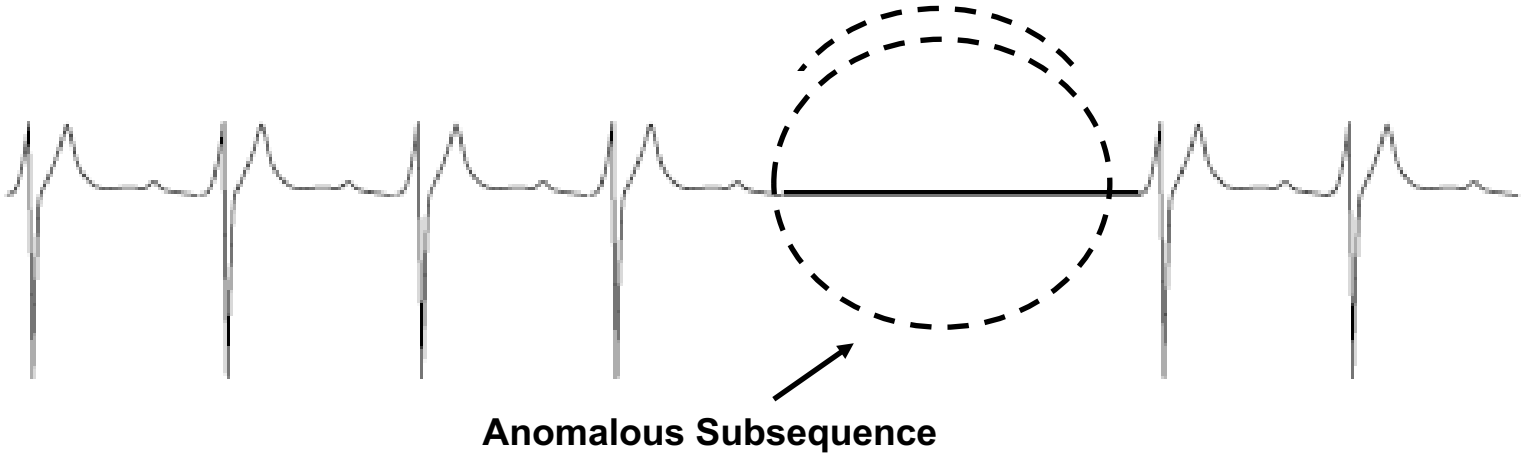
Contextual Anomalies

- Individual data instance is anomalous within a context
- Requires a notion of context
- Also referred to as conditional anomalies



Collective Anomalies

- A collection of related data instances is anomalous
- Requires a relationship among data instances
  - Sequential Data
  - Spatial Data
  - Graph Data
- The individual instances within a collective anomaly are not anomalous by themselves



Key Challenges
for Anomaly Detection Algorithms

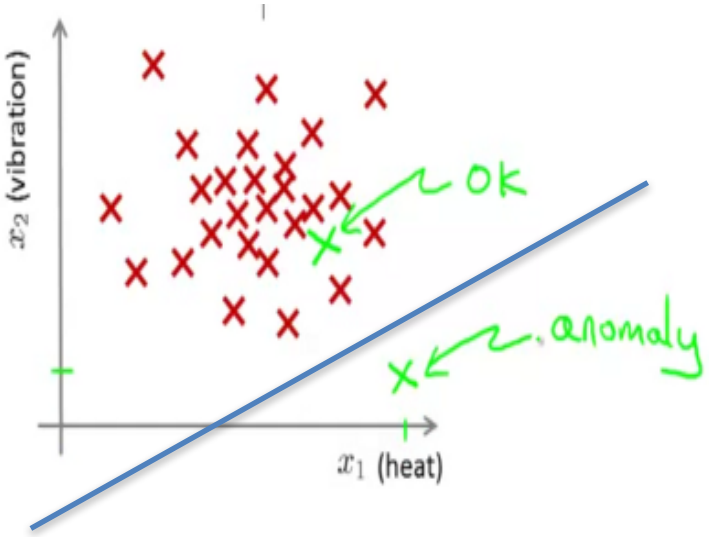
- Defining a representative normal region is challenging
- The boundary between normal and outlying behaviour is often not precise
- The exact notion of an outlier is different for different application domains
- Availability of labelled data for training/validation (unsupervised learning)
- Malicious adversaries
- Data is very noisy
- False positive/negatives
- Normal behaviour keeps evolving (non-stationary distributions)

Machine Learning Approaches

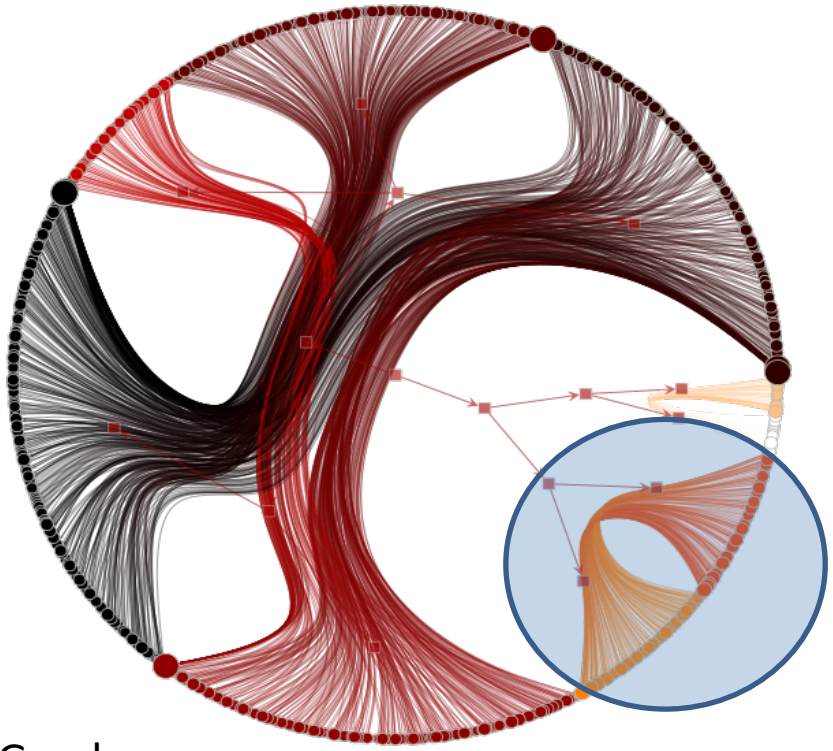
- Time-Based Inductive Methods
  - Use probability and a directed graph to predict the next event
    - Bayesian approaches
    - Can also use undirected approaches (Markov Random Fields)
- Instance Based Learning
  - Define a distance to measure the similarity between feature vectors
    - Histograms, K-Means, …
- Neural Networks
  - Autoencoders: hyper-parameter bounds reconstruction error
- …

Example: Using Flow Data for Anomaly Detection

Generalized Anomaly Detection Setting



Linear Decision Boundary

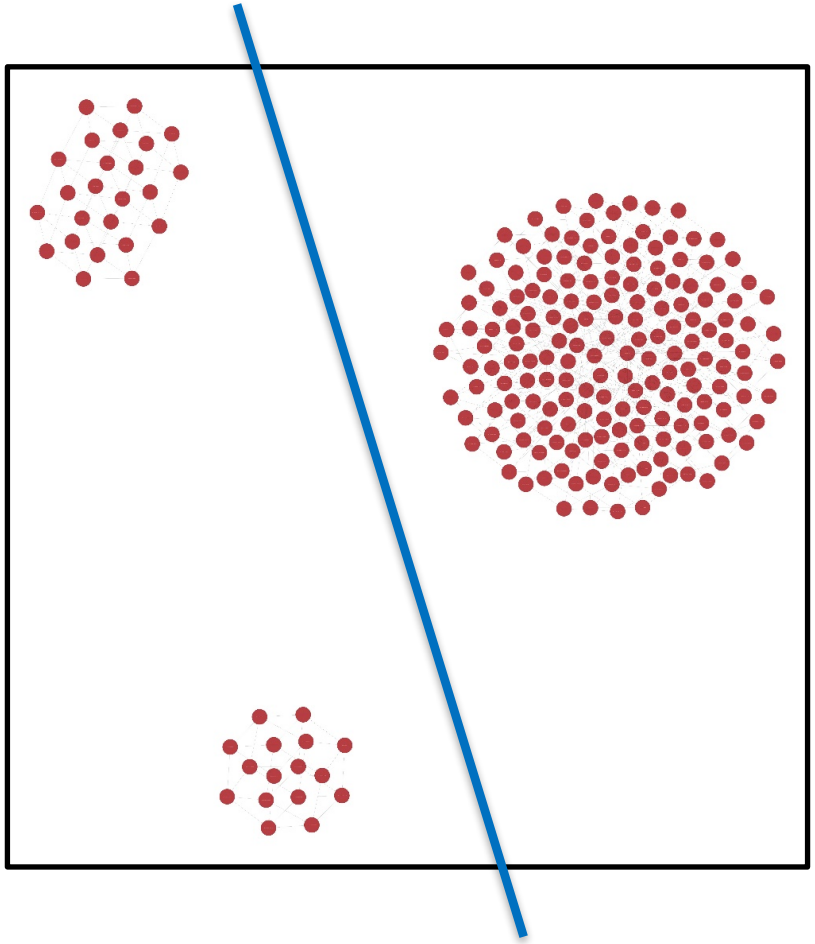


Generalization Graph

Radial Nested Block State Model with Edge Bundling

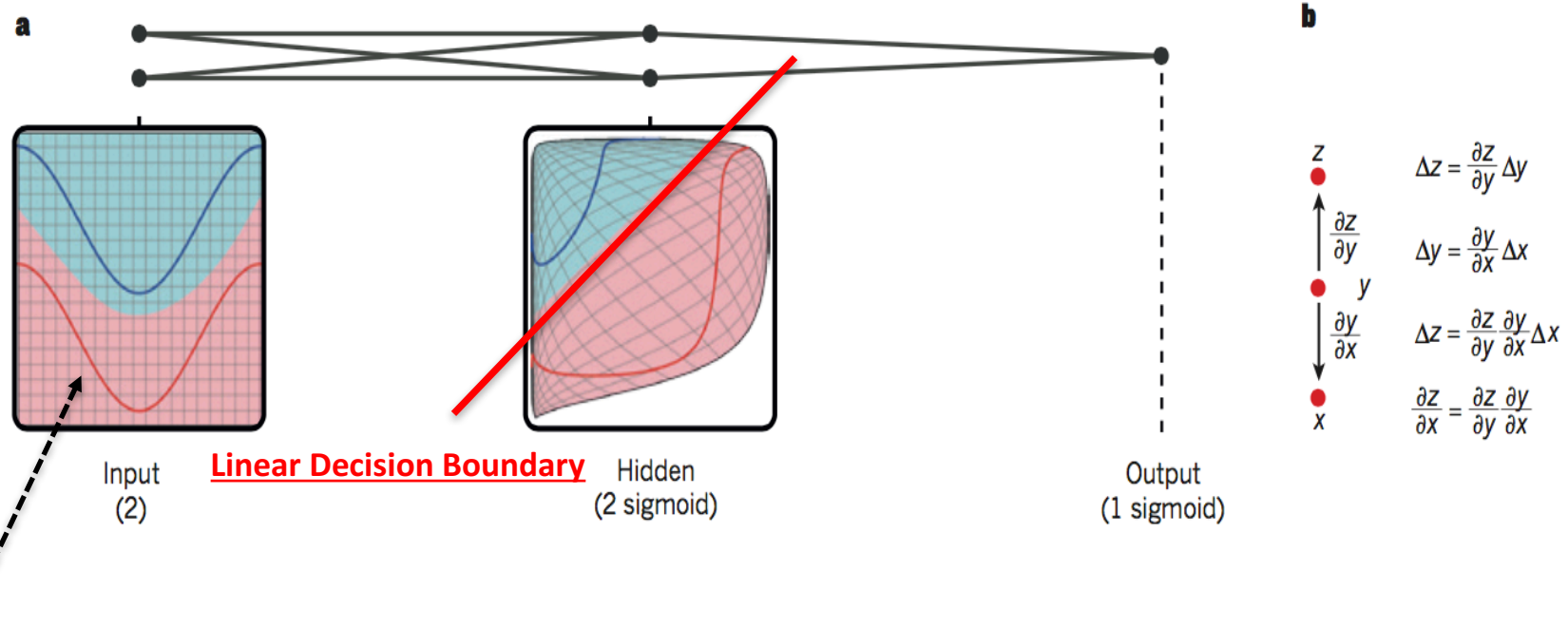
DNS tunneling

Another Visualization (same data)



Linear Decision Boundary

What is Really Happening Here?



Target function represented by input data is some twisted up manifold

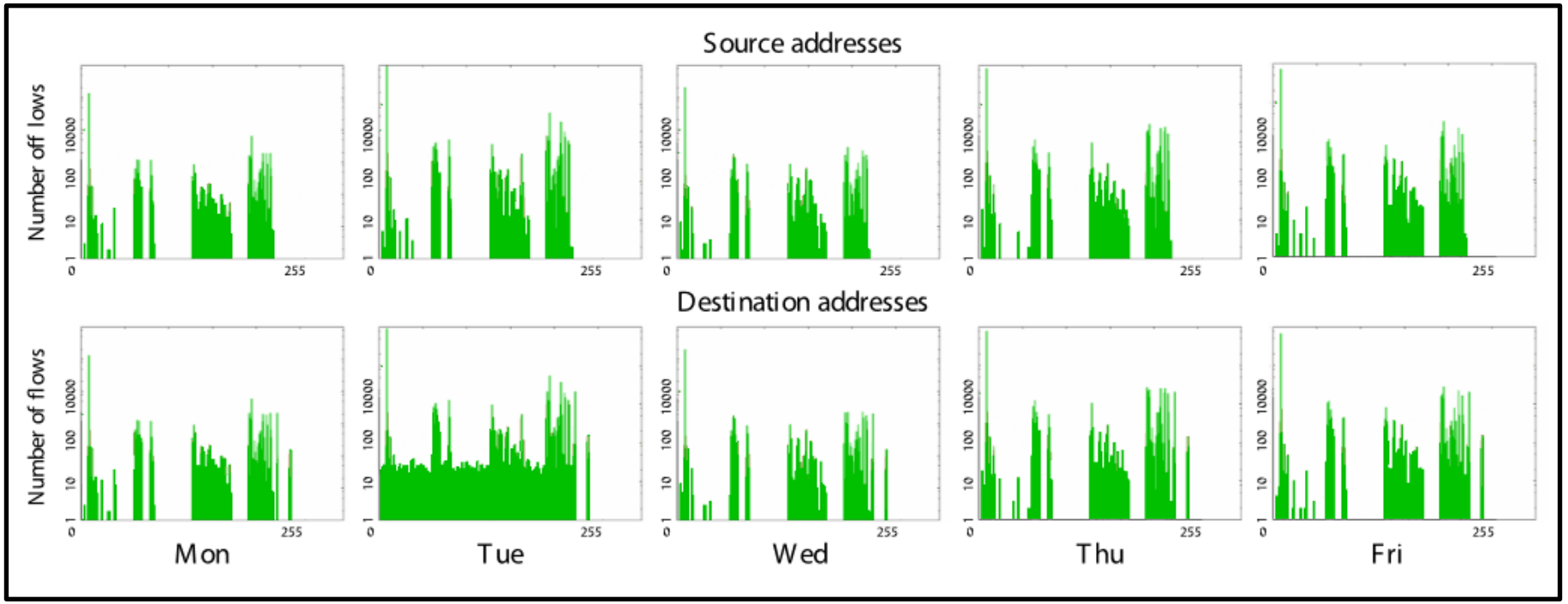
Deep Nets ***disentangle*** the underlying explanatory factors in the data so as to make them *linearly separable*

Graphic courtesy Christopher Olah

Agenda

- Who Am I?
- Level Set: What Is Machine Learning?
- What is all the (Machine Learning) Excitement About?
- Integrated Approaches
- Histograms, LDA, and Neural Networks
- Machine Learning Excitement Redux: Beyond Static Learning
- Technical explanations/code
  - https://github.com/davidmeyer/ml

Histograms



- Essentially unnormalized counts (normalizing a histogram → probability distribution)
- Works with categorical data (again, counts)
- Design metric to reflect the distance between “histograms” (vectors)
- Popular: Mahalanobis Distance (https://en.wikipedia.org/wiki/Mahalanobis_distance)
  - http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html

Overview: Latent Dirichlet Allocation (LDA)

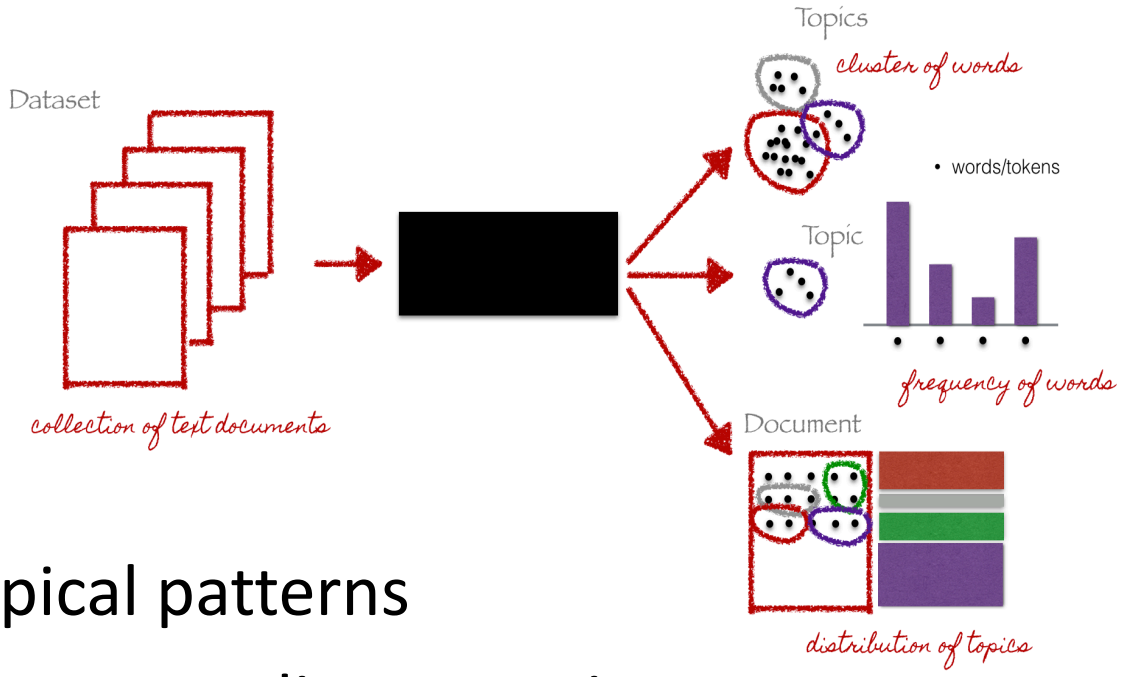
- Proposed as a form of Topic Modeling
- Canonical reference
  - http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf
- Open Network Insight
  - https://github.com/Open-Network-Insight
- Lets take a deeper dive

Sub-agenda: Topic Modeling and LDA

- What is Topic Modeling?
- Parametric vs. Non-Parametric Models
- Latent Dirichlet Allocation
- Probabilistic Graphical Models
- The Effect of the Dirichlet parameter α
- Dynamic LDA

Topic Modeling

- Methods for automatically organizing, understanding, searching and summarizing *categorical data*



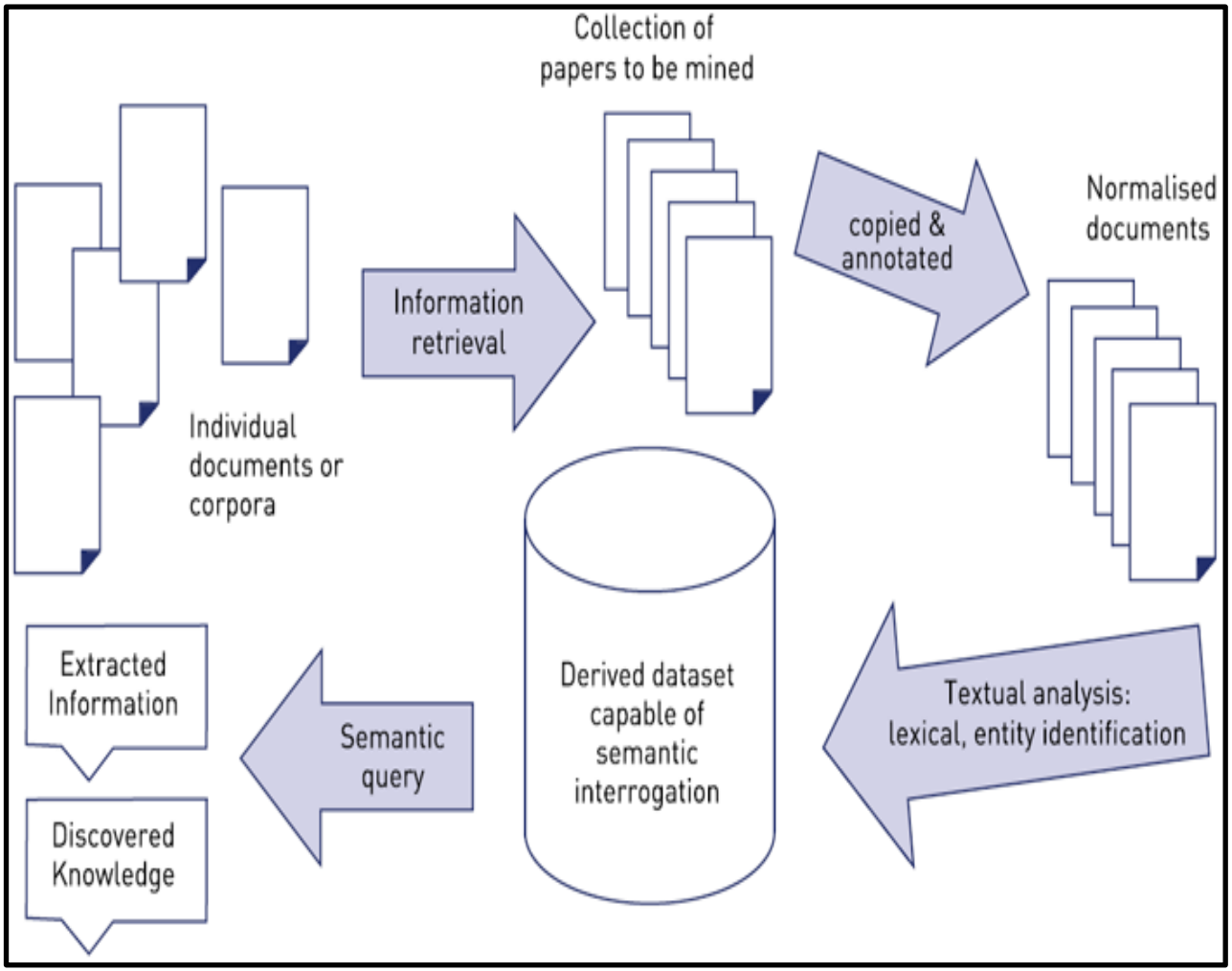
- Goals
  - Uncover hidden topical patterns
  - Annotate documents according to topics
  - Organize, Summarize, Search

A Bit of Information Retrieval (IR) Terminology

- **Corpus:** is a large and structured set of texts
- **Stop words:** words which are filtered out before or after processing of natural language data (text)
- **Unstructured text:** information that either does not have a pre-defined data model or is not organized in a pre-defined manner.
- **Tokenizing:** process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens (see also lexical analysis)
- **Natural language processing:** field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages
- **Term document (or document term) matrix:** is a mathematical matrix that describes the frequency of terms that occur in a collection of documents
- **Supervised learning:** s the machine learning task of inferring a function from labeled training data
- **Unsupervised learning:** find hidden structure in unlabeled data
- **Stemming:** the process for reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form

Graphic courtesy Wikipedia

IR Workflow



Graphic courtesy http://www.jisc.ac.uk/reports/value-and-benefits-of-text-mining

Agenda

- ~~What is Topic Modeling?~~
- Parametric vs. Non-Parametric Models
- Latent Dirichlet Allocation
- Probabilistic Graphical Models
- The Effect of the Dirichlet parameter α
- Dynamic LDA

Parametric vs. Non-parametric Models

- Parametric models assume some finite set of parameters θ. This means that given parameters θ, future predictions **x** are *independent* of the data *D.* That is:
    - $\mathrm{p}(\mathbf{x}|\theta, \mathrm{D}) = \mathrm{p}(\mathbf{x}|\theta)$
- This implies that θ captures everything there is to know about the data
- Also means that the complexity of the model is bounded even if the amount of data isn't
- Taken together these factors make parametric models inflexible
- Information theoretic view
  - Information is constrained to flow from the prior through finite θ to the posterior
  - Remembering that the $posterior = \frac{likelihood \cdot prior}{evidence}$
  - $p(\theta|x) = \frac{p(x|\theta) \cdot p(\theta)}{p(x)} = \frac{p(x,\theta)}{p(x)} = \frac{p(x,\theta)}{\int p(x,\theta) d\theta}$
  - http://www.1-4-5.net/~dmm/ml/ps.pdf
- Latent Dirichlet Allocation is a parametric model (why?)

Non-Parametric Models

- Non-parametric models assume that the data distribution cannot be defined in terms of such a finite set of parameters
- Parameters can often be described using an infinite dimensional θ
  - Think of θ as a function
- The amount of information that θ can capture about the data D can grow as the amount of data grows
- This makes non-parametric models more flexible
  - Better predictive performance
  - More "realistic"
  - Most successful ML models are non-parametric
- Examples include
  - Kernel methods (SVMs, GPs)
  - DNNs
  - K-NNs, …

Agenda

- ~~What is Topic Modeling?~~
- ~~Parametric vs. Non-Parametric Models~~
- Latent Dirichlet Allocation
- Probabilistic Graphical Models
- The Effect of the Dirichlet parameter α
- Dynamic LDA
- Q&A

Latent Dirichlet Allocation (LDA)

- Generative probabilistic model
  - ***Parametric*** Bayesian Probabilistic Graphical Model
  - Treats data as observations
  - Contains hidden variables
  - Hidden variables reflect thematic structure of the collection
  - Wealth of material here:
    - https://www.cs.princeton.edu/~blei/topicmodeling.html
- Approach: Infer hidden structure using *posterior inference*
  - Discovering topics in the collection using Bayesian inference
- Placing new data into the estimated model
  - Situating new documents into the estimated topic structure
- Other Approaches
  - Latent Semantic Indexing (LSI)
  - Probabilistic Latent Semantic Indexing (pLSI)
  - …

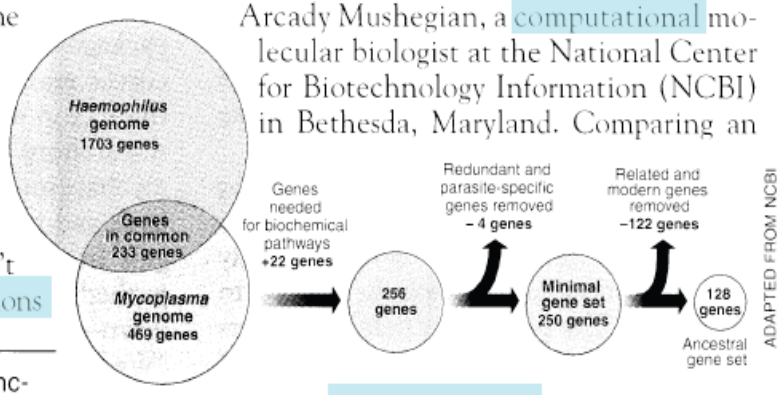
LDA Basic Intuition

http://ai.stanford.edu/~ang/papers/nips01-lda.pdf

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

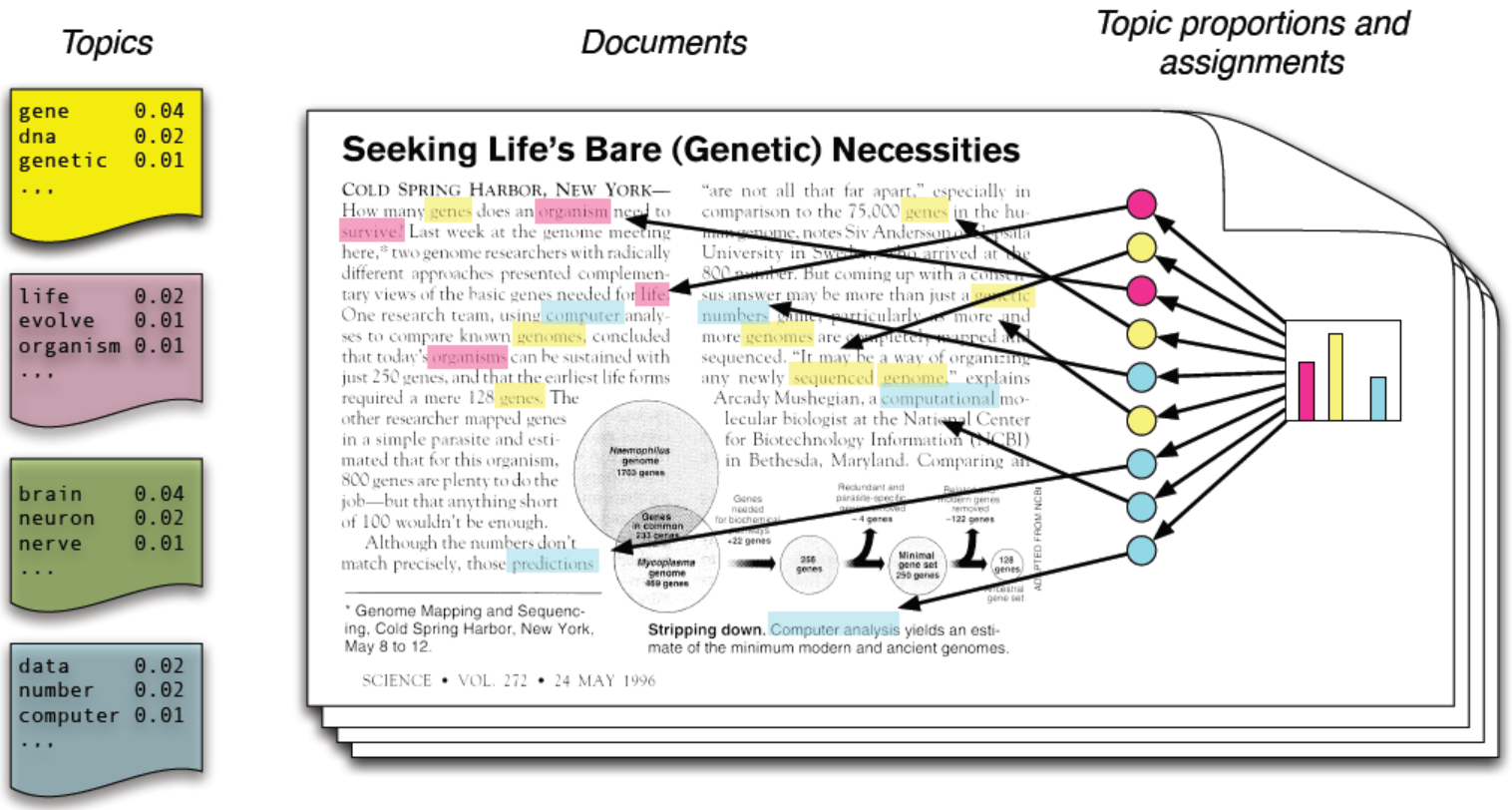
Obvious Questions

- What exactly is a topic?
- Where do topics come from?
- How many "topics"/document
- Simple Intuition:
  - Documents exhibit multiple topics
  - Contrast "mixture" models

Graphic courtesy David Blei https://www.cs.princeton.edu/~blei/kdd-tutorial.pdf

LDA Generative Model

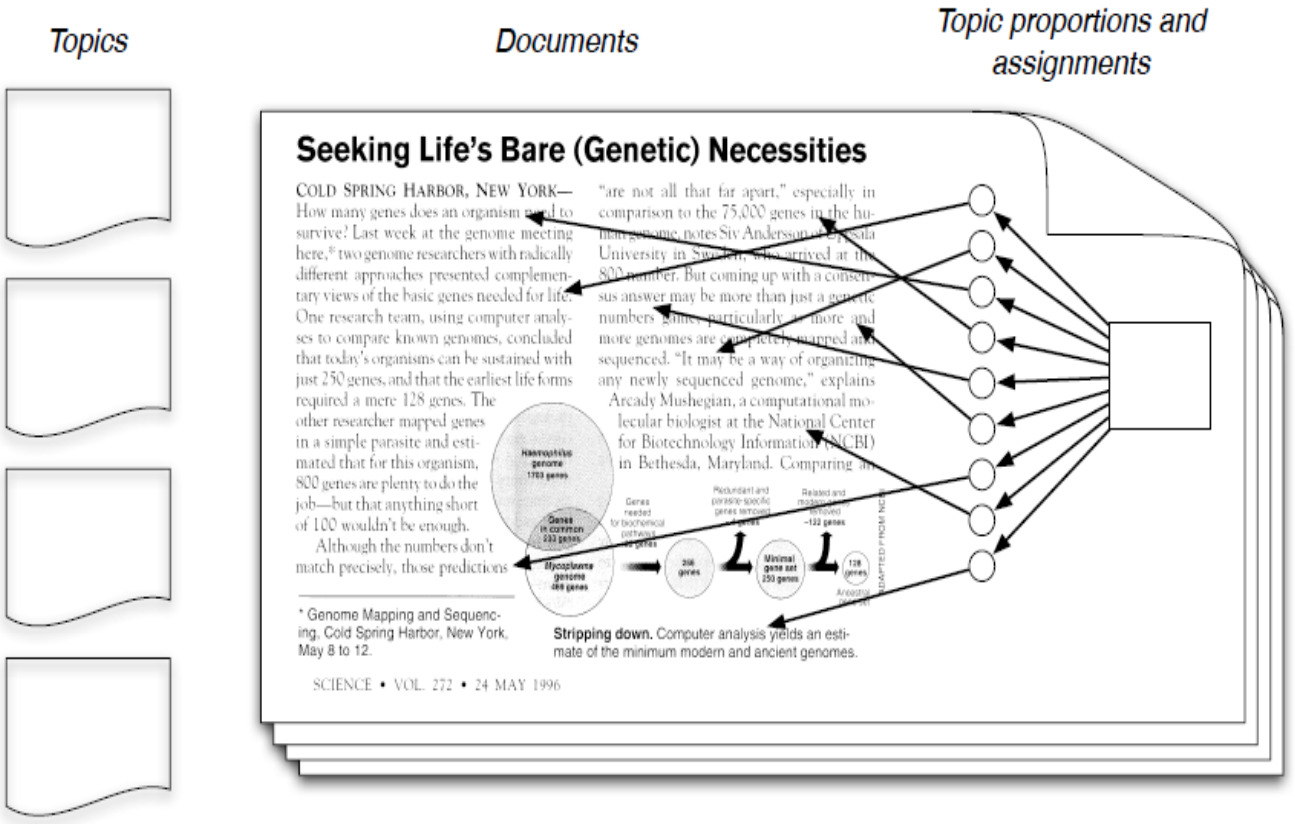
Hallucinate that the observed data were generated this way



- Assume topics exists outside of the document collection
- Each topic is a *distribution* over fixed vocabulary
  - Each word is drawn from one of those topics
- Each document is a *random mixture* of corpus-wide topics

Graphic courtesy David Blei https://www.cs.princeton.edu/~blei/kdd-tutorial.pdf

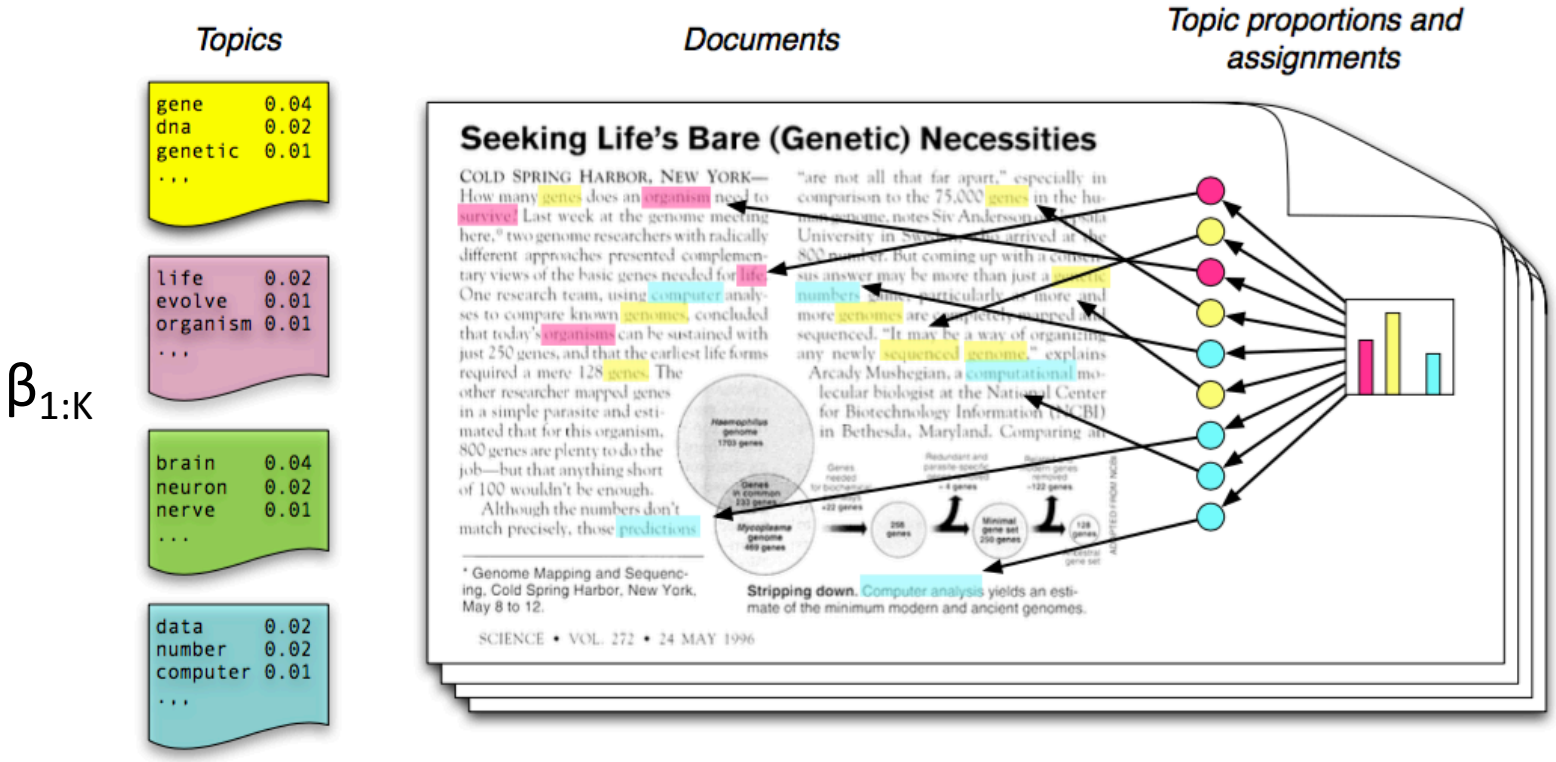
BTW, What Do We Actually Observe?



- So our goal here is to **infer** the hidden (latent) variables
- i.e., compute their distribution conditioned on the documents:

$$p(\text{topics, proportions, assignments} \mid \text{documents})$$

Graphic courtesy David Blei https://www.cs.princeton.edu/~blei/kdd-tutorial.pdf



Let K be the number of topics (distributions over words) # $\beta_{1:K} \sim \text{Dir}(\eta)$

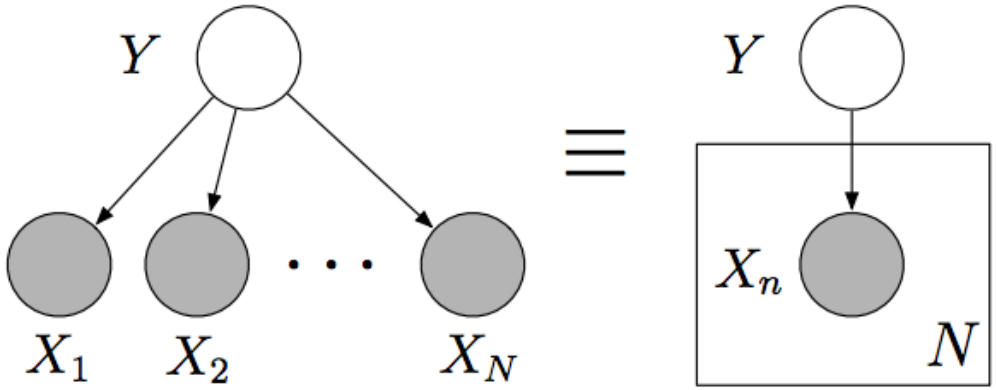
For each document d in corpus C:

- Draw the topic proportion distribution θ_d for document d # $\theta_d \sim \text{Dir}(\alpha)$
- For each n = 1..N_d: # N_d = # words in d
  - Draw topic index $z_{d,n}$ for the n^{th} word of d from θ_d # $1 \leq z_{d,n} \leq K$
  - Draw word $w_{d,n}$ from topic $\beta[z_{d,n}]$ # $z_{d,n}$ indexes $\beta_{1:K}$

Agenda

- ~~What is Topic Modeling?~~
- ~~Parametric vs. Non-Parametric Models~~
- ~~Latent Dirichlet Allocation~~
- Probabilistic Graphical Models
- The Effect of the Dirichlet parameter α
- Dynamic LDA
- Q&A

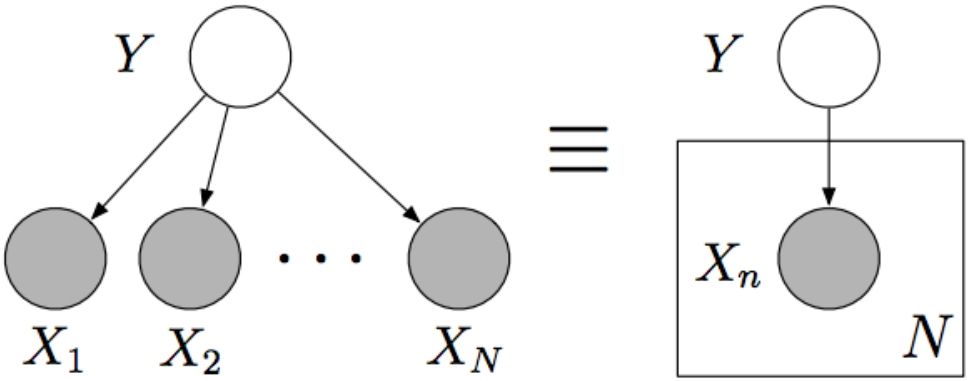
Probabilistic Graphical Models



- Nodes are random variables
- Edges denote possible dependence
- Observed variables are shaded
- Plates denote replicated structure

Graphic courtesy https://www.cs.princeton.edu/~blei/kdd-tutorial.pdf

Probabilistic Graphical Models, Cont



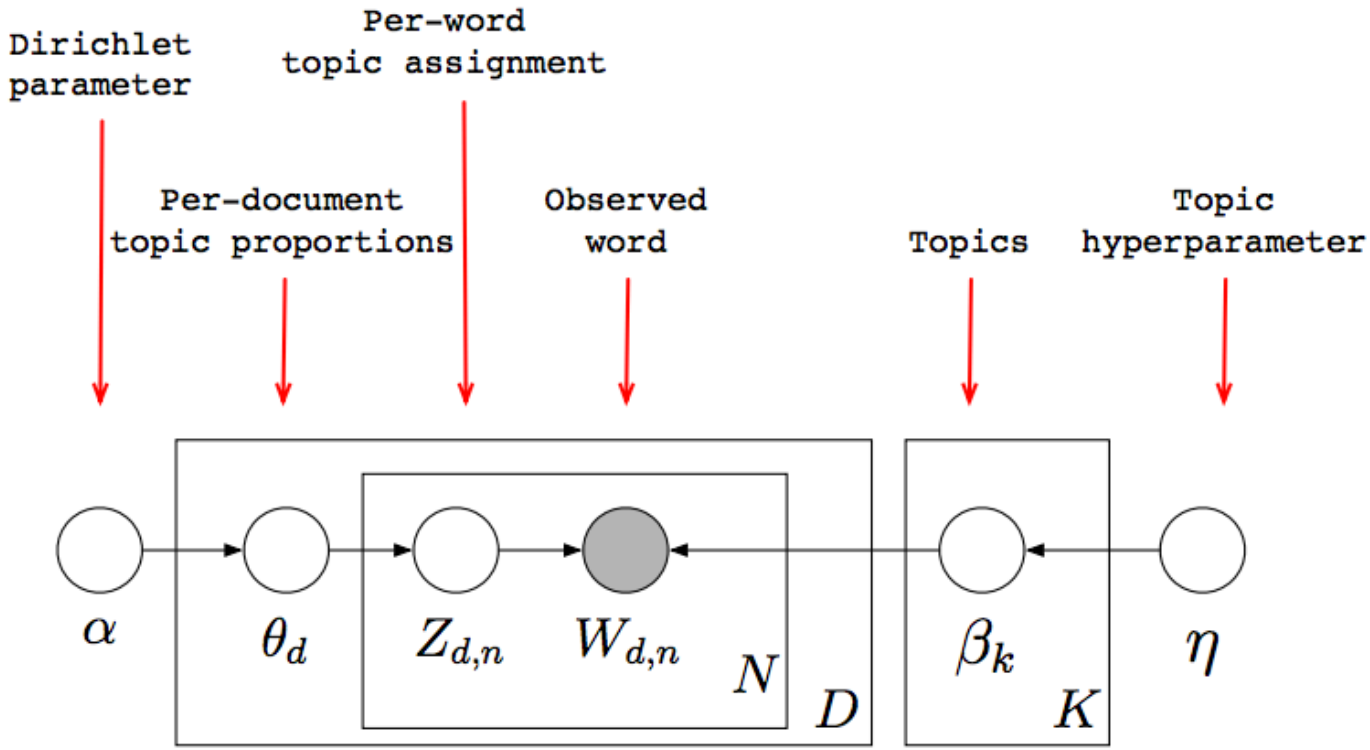
- Structure of the graph defines the pattern of conditional dependence between the ensemble of random variables
- E.g., this graph corresponds to

$$p(y, x_1, \ldots, x_N) = p(y) \prod_{n=1}^{N} p(x_n \mid y)$$

Graphic courtesy https://www.cs.princeton.edu/~blei/kdd-tutorial.pdf

LDA Graphical Model

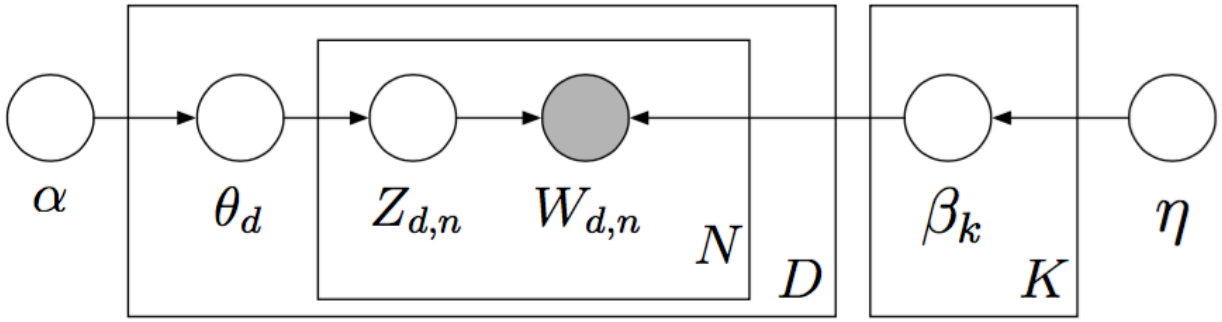
(based on the generative model)



Each piece of the structure is a random variable.

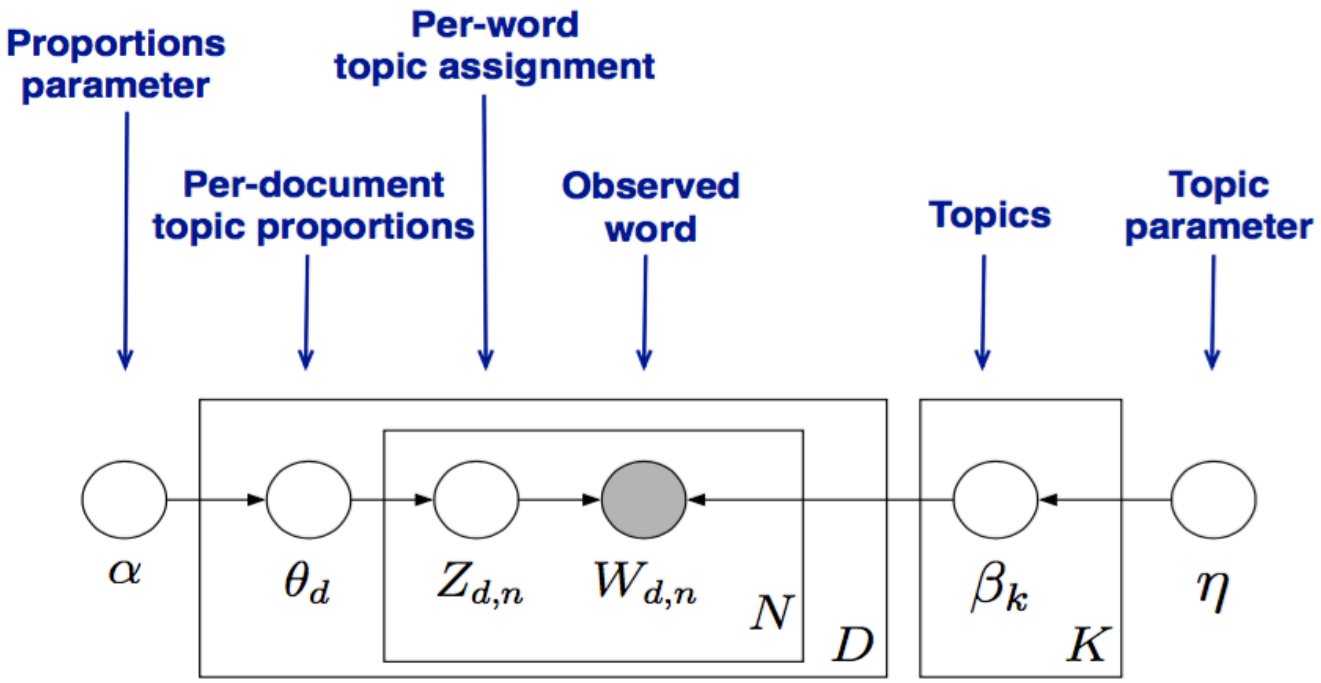
Graphic courtesy https://www.cs.princeton.edu/~blei/kdd-tutorial.pdf

LDA Graphical Model Details



- From a collection of documents, infer
  - Per-word topic assignment $z_{d,n}$
  - Per-document topic proportions θ_d
  - Per-corpus topic distributions β_k
- Use posterior expectations to perform the task at hand, e.g., information retrieval, document similarity, etc.

What is the LDA Joint Distribution?

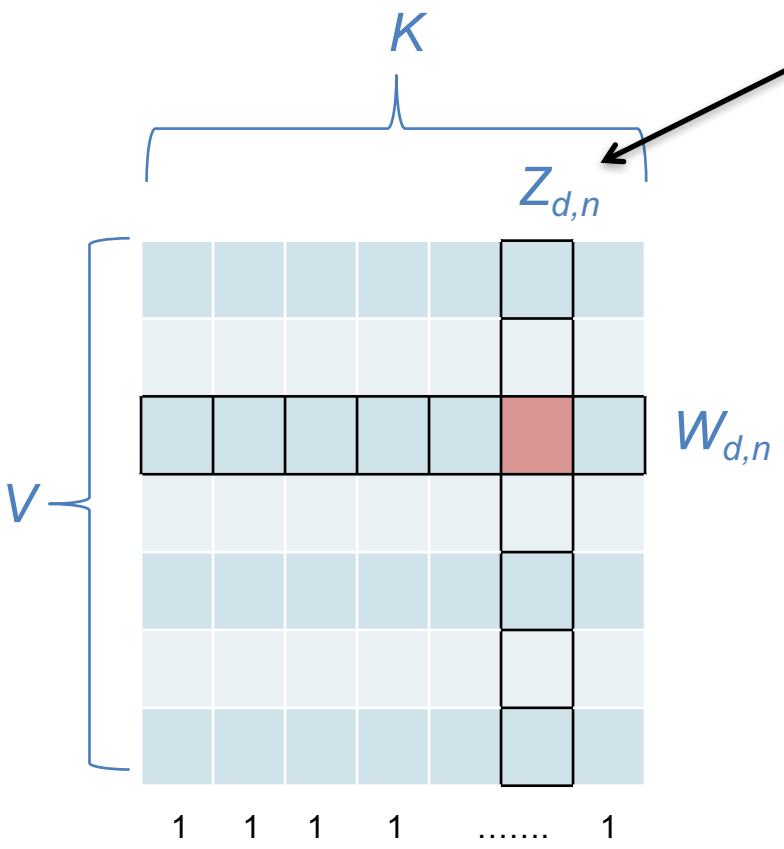


$$p(\beta, \theta, \mathbf{z}, \mathbf{w}) = \left(\prod_{i=1}^{K} p(\beta_i \mid \eta) \right) \left(\prod_{d=1}^{D} p(\theta_d \mid \alpha) \prod_{n=1}^{N} p(z_{d,n} \mid \theta_d) p(w_{d,n} \mid \beta_{1:K}, z_{d,n}) \right)$$

Why does $w_{d,n}$ depend on $z_{d,n}$ and β?

V x K Topic Matrix

Columns are the β_ks



- K -- number of topics
- V -- number of words in the vocabulary
- TM_{VxK} -- topic matrix
- $Z_{d,n}$ -- index of topic which $W_{d,n}$ comes from
- $W_{d,n}$ -- the n^{th} word in the d^{th} document
- $TM[W_{d,n},Z_{d,n}]$ -- the probability of $W_{d,n}$, $\beta_k[w_{d,n}]$ ($k = z_{d,n}$)

Now, What Exactly is the Dirichlet Distribution

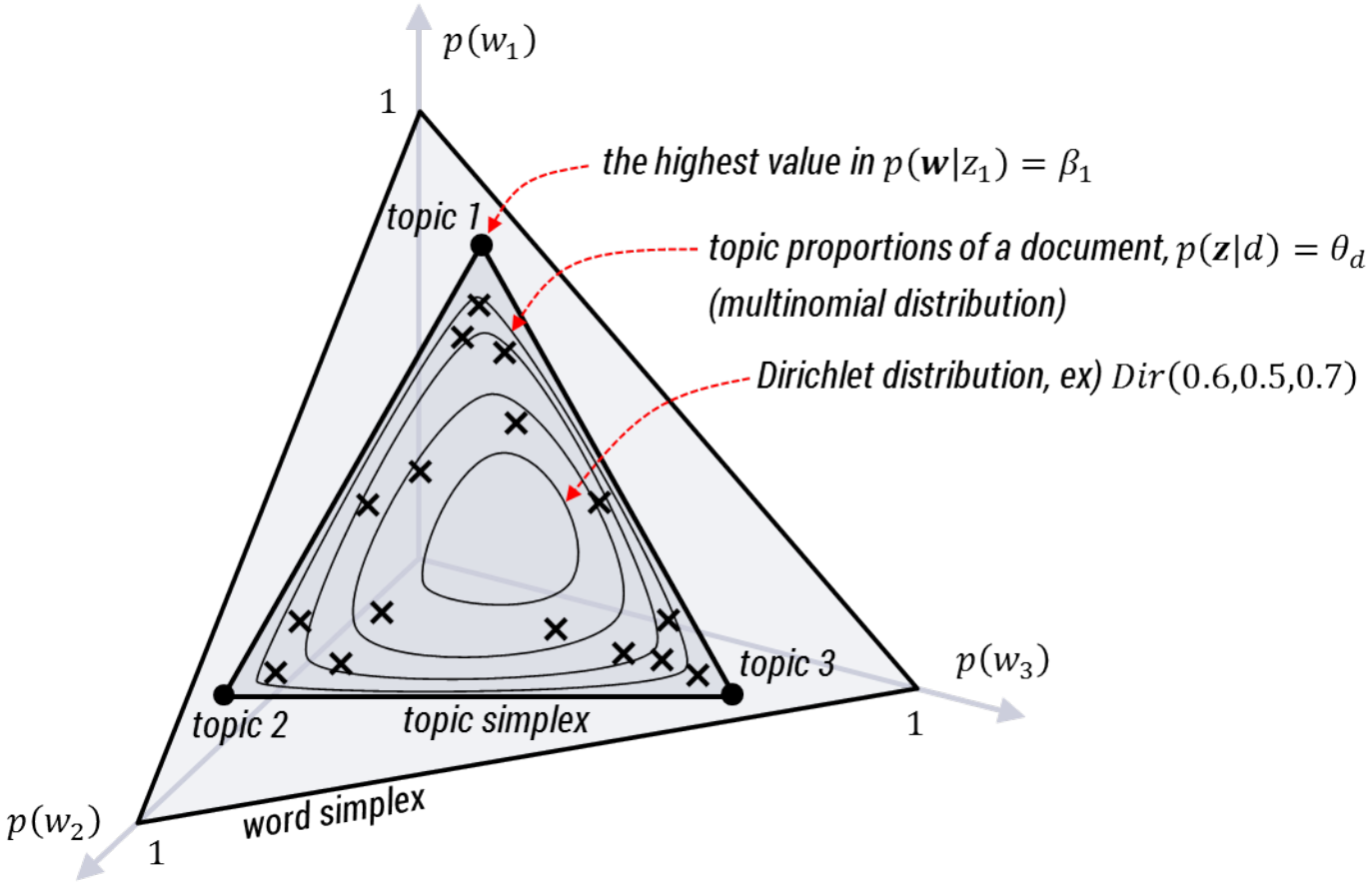
(and why are we using it?)

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1}$$

- The Dirichlet is a “dice factory”
  - Multivariate equivalent of the Beta distribution (“coin factory”)
  - Parameters α determine the form of the prior
- The Dirichlet is defined over the (k-1) simplex
  - The k non-negative arguments which sum to one
- The Dirichlet is the ***conjugate prior*** to the multinomial distribution
  - If the likelihood has conjugate prior P then the posterior has the same form as P
    - If we have a conjugate prior we know the (closed) form of the posterior
    - So in this case the posterior is also a Dirichlet
- The parameter α controls the mean shape and sparsity of θ
- In LDA the topics are a V-dimensional Dirichlet and the topic proportions are a K-dimensional Dirichlet

Simplex?

(space of non-negative vectors which sum to one)



Aside: Conjugate Priors

Likelihood $f(y\|\theta)$	**Prior** $\pi(\theta)$	**Posterior** $\pi(\theta\|y)$
Normal $\mathcal{N}(\theta, \sigma^2)$	Normal $\mathcal{N}(\mu, \tau^2)$	Normal $\mathcal{N}\left(\frac{\sigma^2\mu+\tau^2 y}{\sigma^2+\tau^2}, \frac{\sigma^2\tau^2}{\sigma^2+\tau^2}\right)$
Poisson $\text{Poisson}(\theta)$	Gamma $\Gamma(\alpha, \beta)$	Gamma $\Gamma(\alpha + y, \beta + 1)$
Gamma $\Gamma(\nu, \theta)$	Gamma $\Gamma(\alpha, \beta)$	Gamma $\Gamma(\alpha + \nu, \beta + y)$
Binomial $\text{Bin}(n, \theta)$	Beta $\text{Beta}(\alpha, \beta)$	Beta $\text{Beta}(\alpha + y, \beta + n - y)$
Multinomial $\text{M}_k(\theta_1, \ldots, \theta_k)$	Dirichlet $\text{D}(\alpha_1, \ldots, \alpha_k)$	Dirichlet $\text{D}(\alpha_1 + y_1, \ldots, \alpha_k + y_k)$
Normal $\mathcal{N}(\mu, 1/\theta)$	Gamma $\Gamma(\alpha, \beta)$	Gamma $\Gamma\left(\alpha + \frac{1}{2}, \beta + \frac{1}{2}(\mu - y)^2\right)$

Notes/Issues:

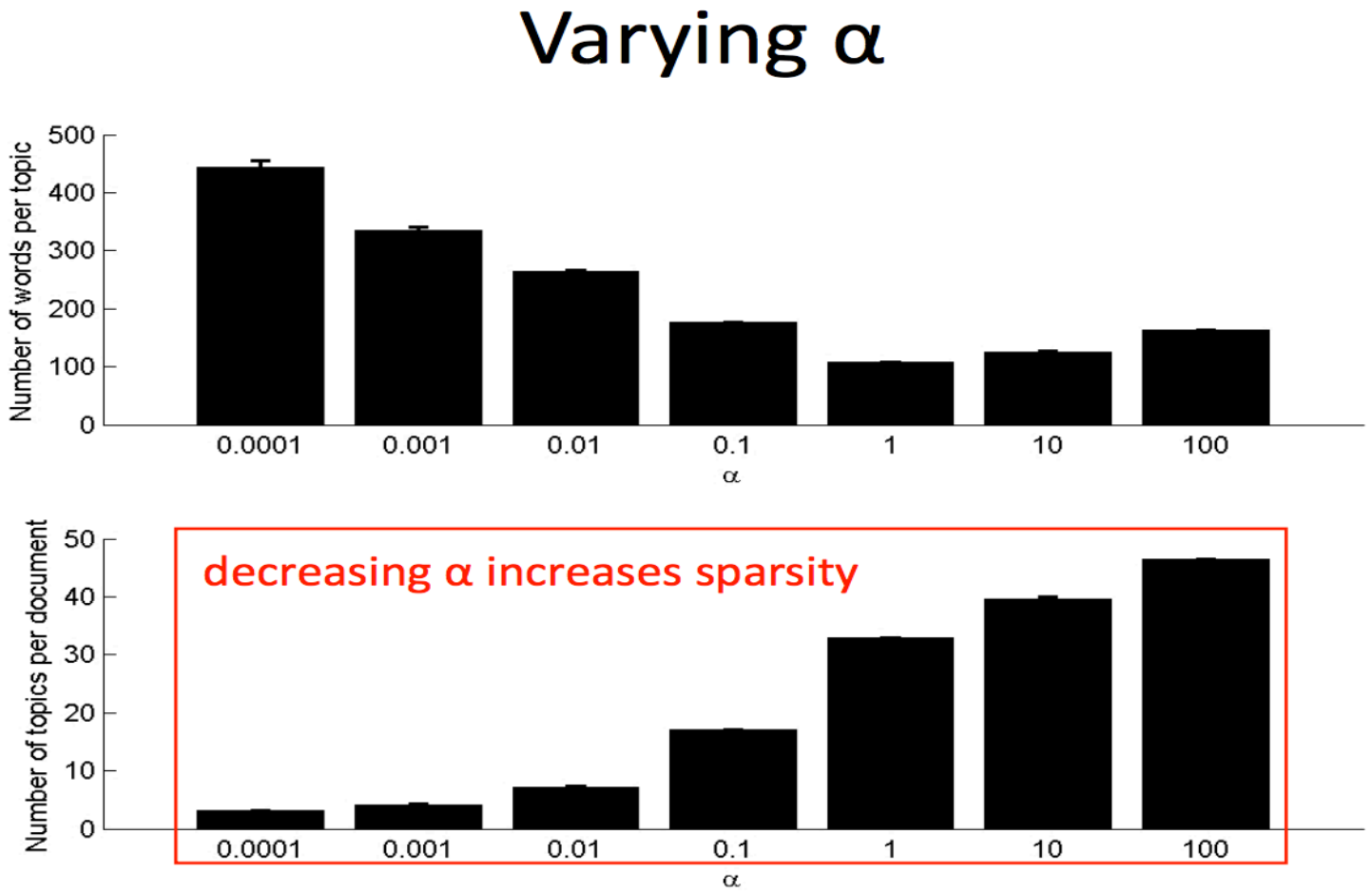
- Conjugate Prior: If the likelihood has conjugate prior P then the posterior has the same form as P
- The conjugate prior might not correctly reflect our uncertainty about θ
- Non-conjugate priors are typically not available in analytic (closed) form
- It is difficult to quantify our uncertainty about θ in the form of a particular distribution

Agenda

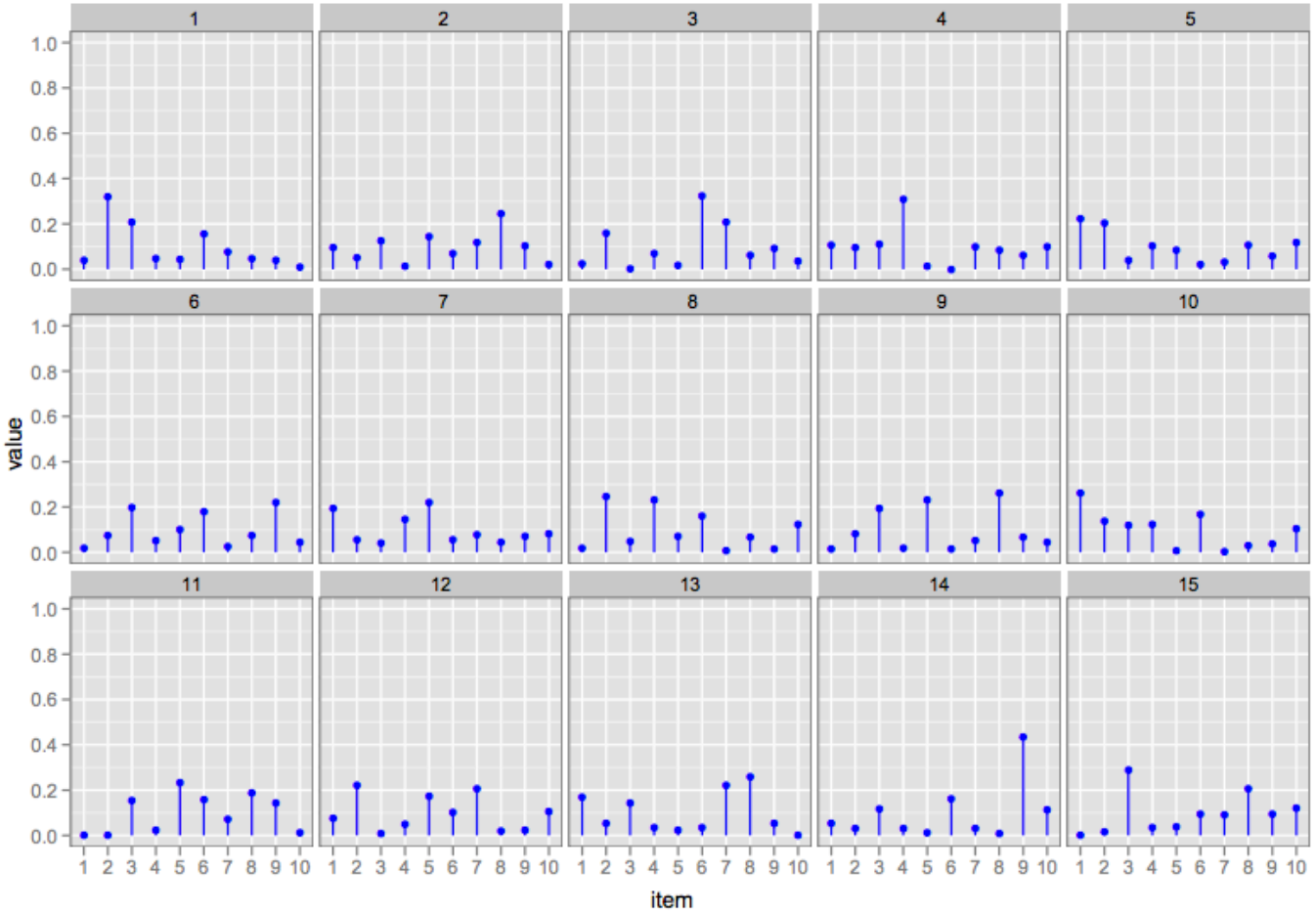
- ~~What is Topic Modeling?~~
- ~~Parametric vs. Non-Parametric Models~~
- ~~Latent Dirichlet Allocation~~
- ~~Probabilistic Graphical Models~~
- The Effect of the Dirichlet parameter α
- Dynamic LDA

The Effect of the Dirichlet parameter α

Generalizing the Idea of Co-Occurrence

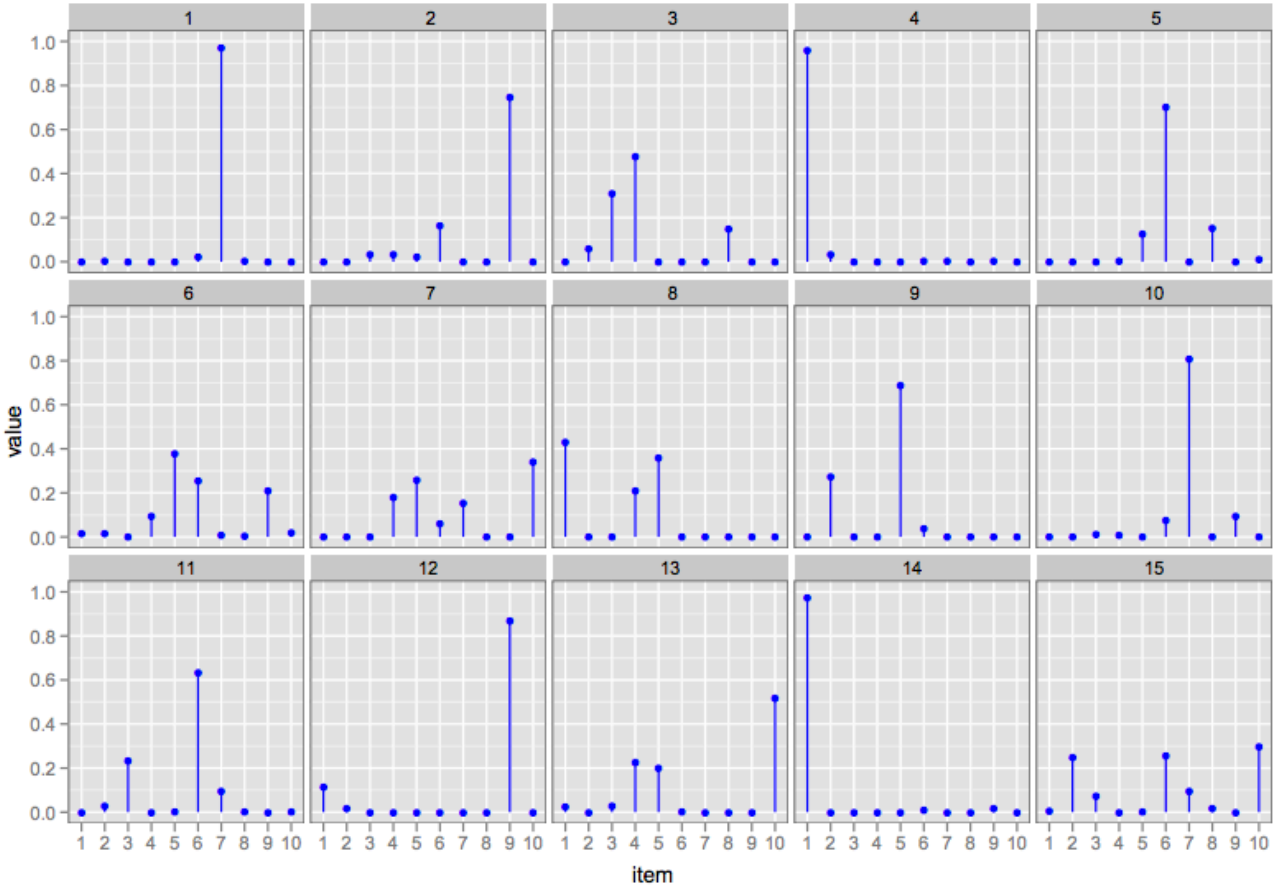


α = 1

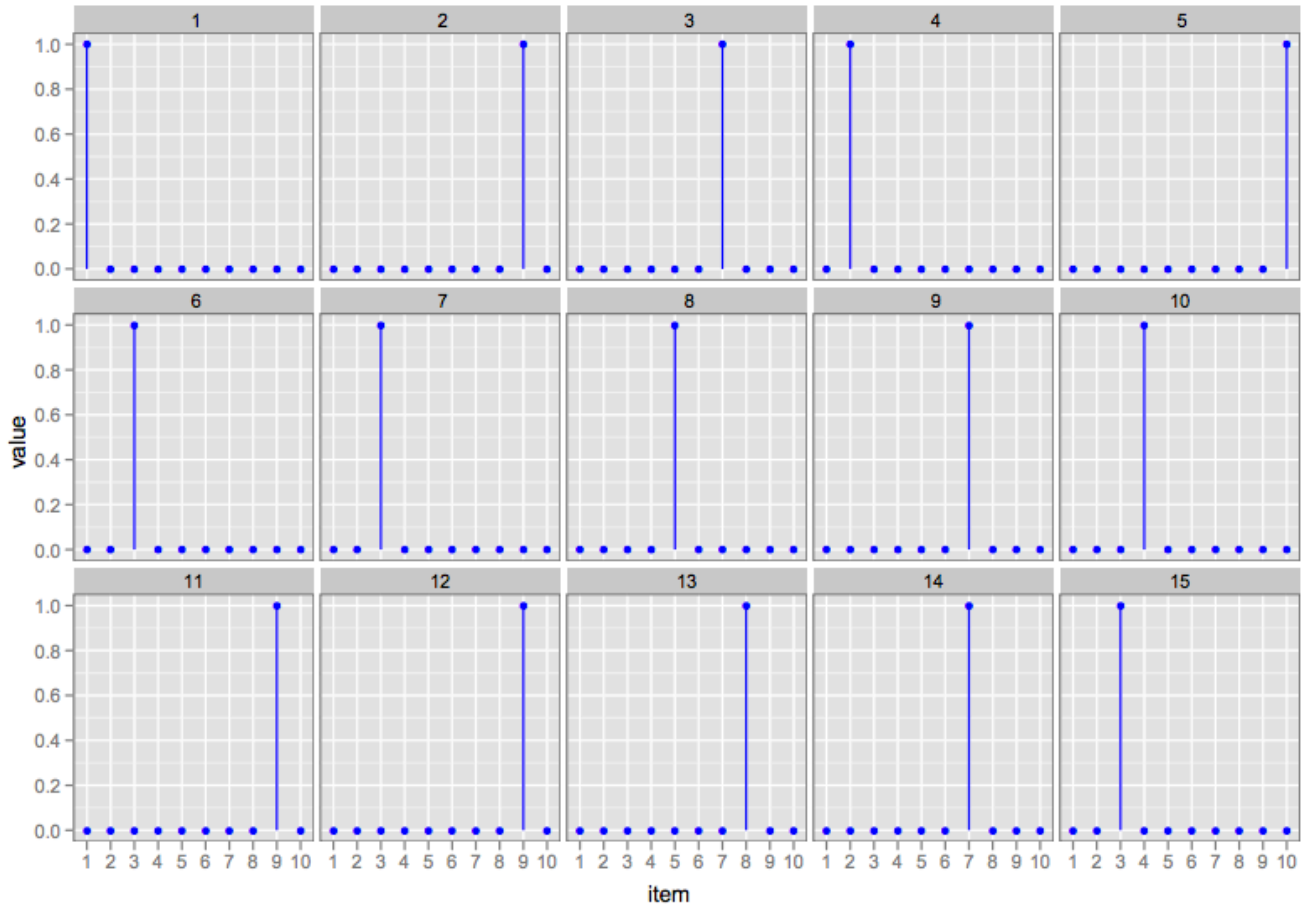


Graphic courtesy https://www.cs.princeton.edu/~blei/kdd-tutorial.pdf

$\alpha = 0.1$

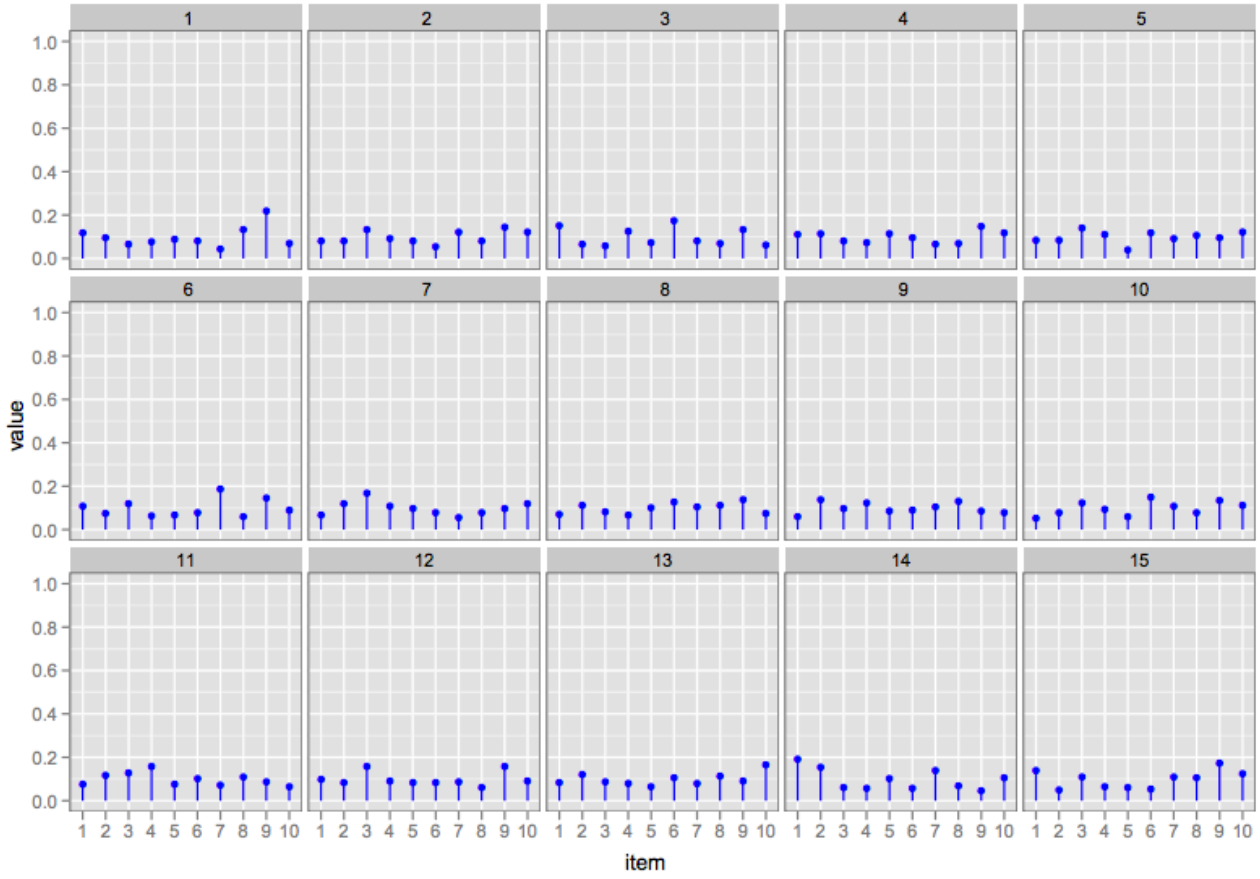

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
value
item
0.0
0.2
0.4
0.6
0.8
1.0
1 2 3 4 5 6 7 8 9 10

$\alpha = 0.001$

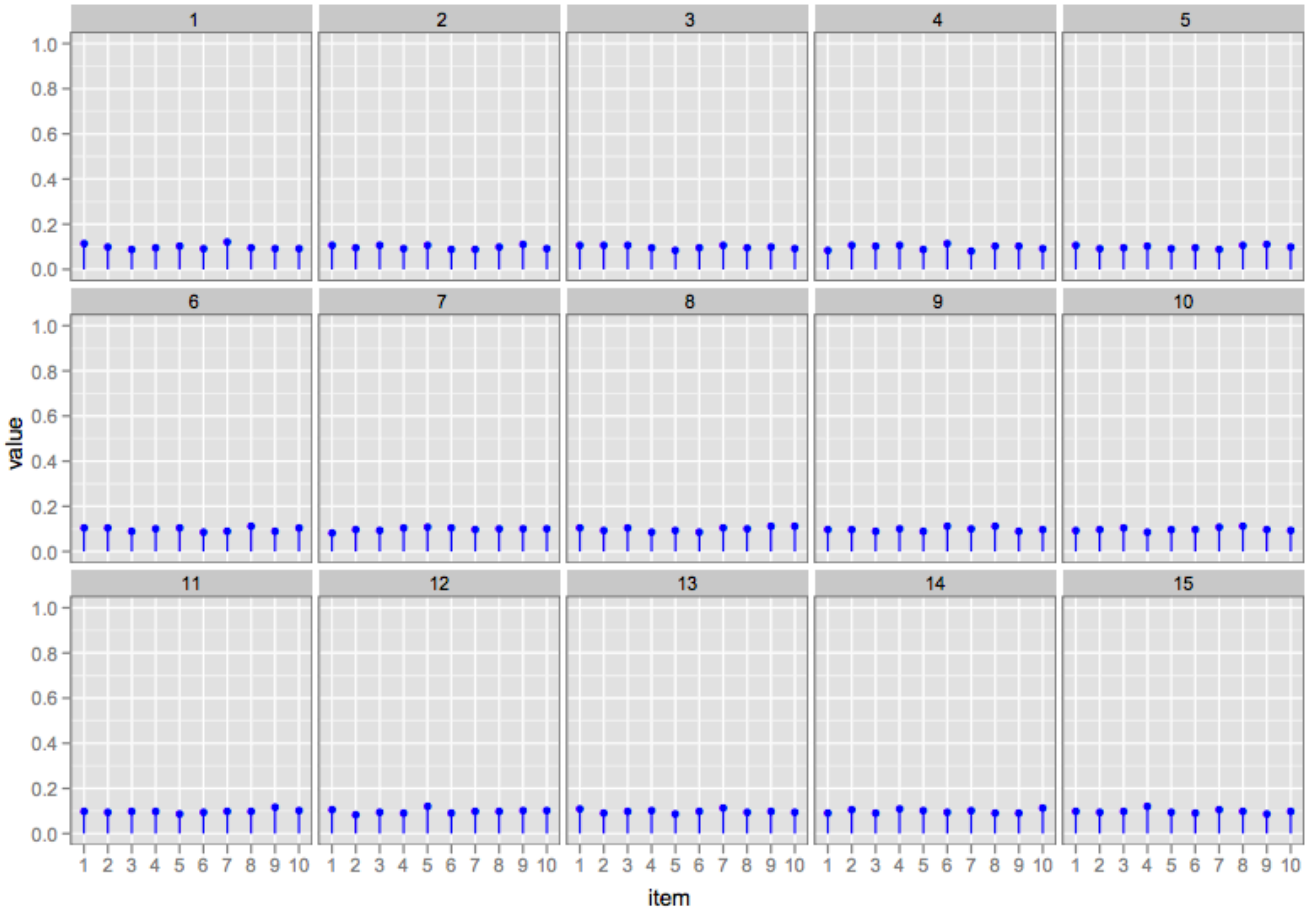


Essentially a mixture model (one topic/document)

α = 10


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
value
item
0.0
0.2
0.4
0.6
0.8
1.0
1 2 3 4 5 6 7 8 9 10

α = 100



Larger α spreads out the mass

Ok, but we need the Posterior

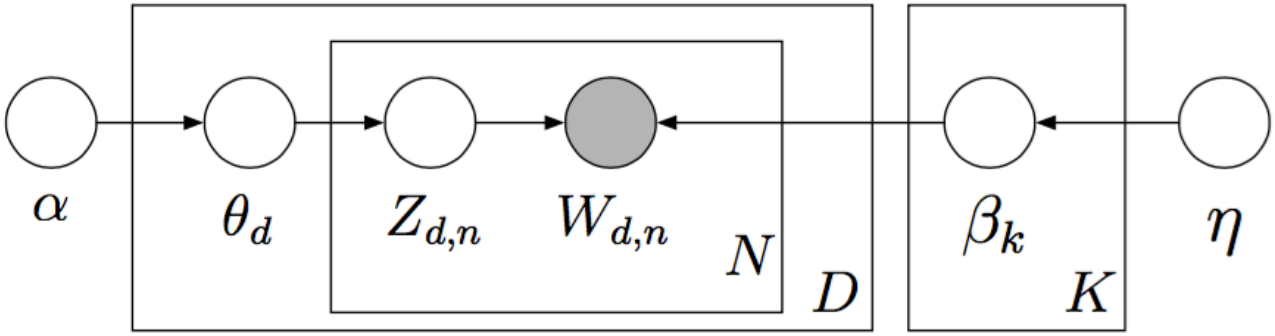
- For now, assume the topics $\beta_{1:K}$ are fixed.
  The per-document posterior is

$$\frac{p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta) p(w_n \mid z_n, \beta_{1:K})}{\int_\theta p(\theta \mid \alpha) \prod_{n=1}^{N} \sum_{z=1}^{K} p(z_n \mid \theta) p(w_n \mid z_n, \beta_{1:K})}$$

- This is intractable to compute
- It is a "multiple hypergeometric function" (see Dickey, 1983)
- Can be seen as sum of N^K (tractable) Dirichlet integral terms

So we need to use approximate inference

Approximate Inference



We appeal to approximate posterior inference of the posterior,

$$\frac{p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta) p(w_n \mid z_n, \beta_{1:K})}{\int_\theta p(\theta \mid \alpha) \prod_{n=1}^{N} \sum_{z=1}^{K} p(z_n \mid \theta) p(w_n \mid z_n, \beta_{1:K})}$$

- Gibbs sampling
- Variational methods
- Particle filtering

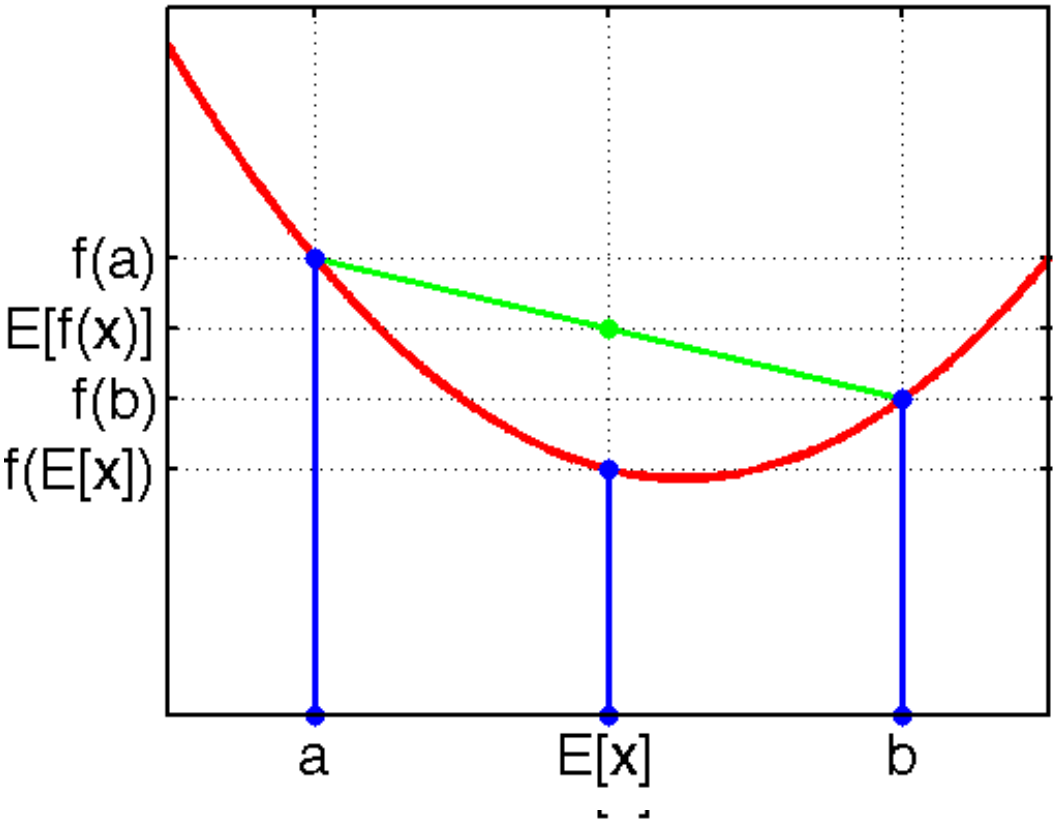
Variational Inference

- Variational methods are a deterministic alternative to MCMC.
- Let $x_{1:N}$ be observations and $z_{1:M}$ be latent variables
- Our goal is to compute the posterior distribution

$$p(z_{1:M} \mid x_{1:N}) = \frac{p(z_{1:M}, x_{1:N})}{\int p(z_{1:M}, x_{1:N}) dz_{1:M}}$$

- For many interesting distributions, the marginal likelihood of the observations is difficult to efficiently compute

Jensen’s Inequality



- Jensen's inequality generalizes the observation that the secant line of a convex function lies *above* the graph of the function

- In probability theory Jensen’s Inequality is generally stated in the following form: If X is a random variable and φ is a *convex* function, then $\varphi(E[X]) \leq E[\varphi(X)]$

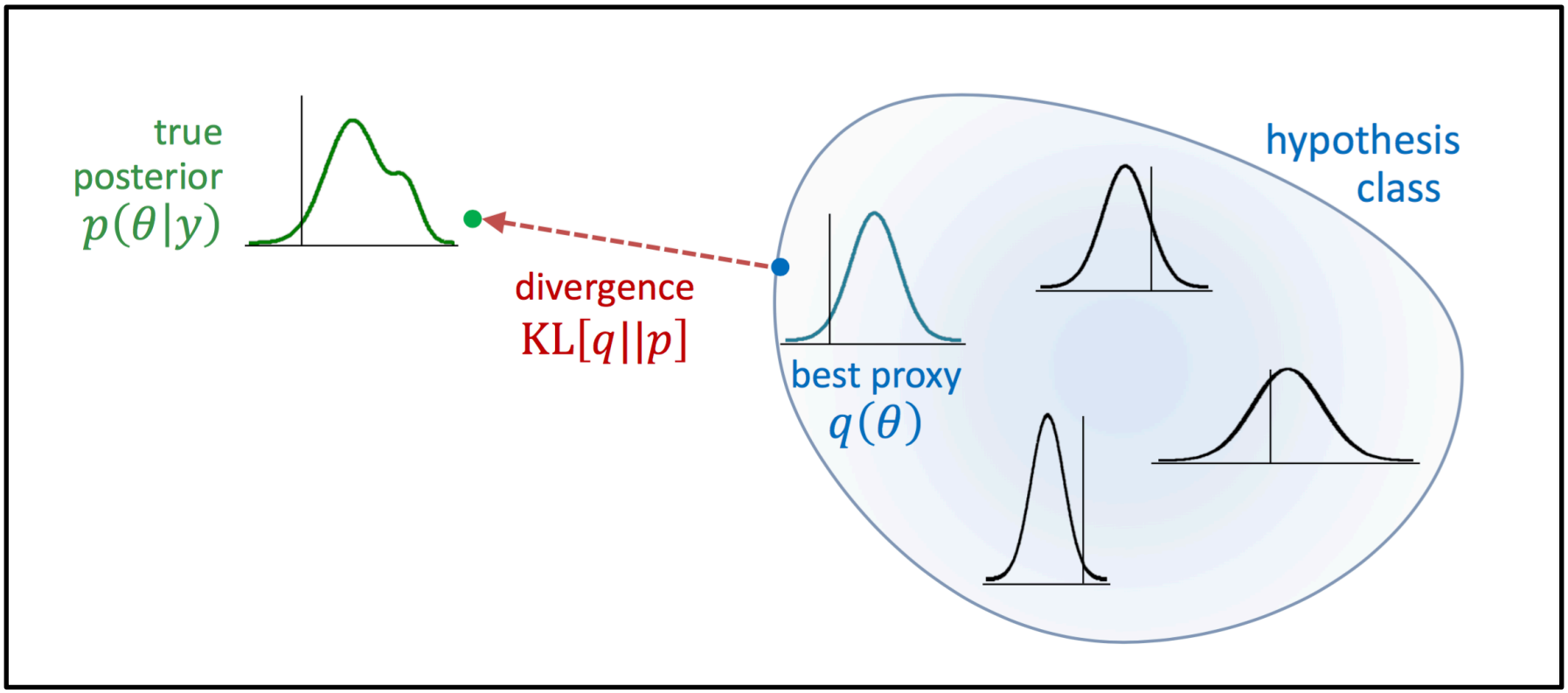
Jensen's Inequality, Bounds, and KL Divergence

- Use Jensen's inequality to bound the log prob of the observations:

$$\begin{aligned}
\log p(x_{1:N}) &= \log \int p(z_{1:M}, x_{1:N}) dz_{1:M} \\
&= \log \int p(z_{1:M}, x_{1:N}) \frac{q_\nu(z_{1:M})}{q_\nu(z_{1:M})} dz_{1:M} \\
&\geq \mathrm{E}_{q_\nu}[\log p(z_{1:M}, x_{1:N})] - \mathrm{E}_{q_\nu}[\log q_\nu(z_{1:M})]
\end{aligned}$$

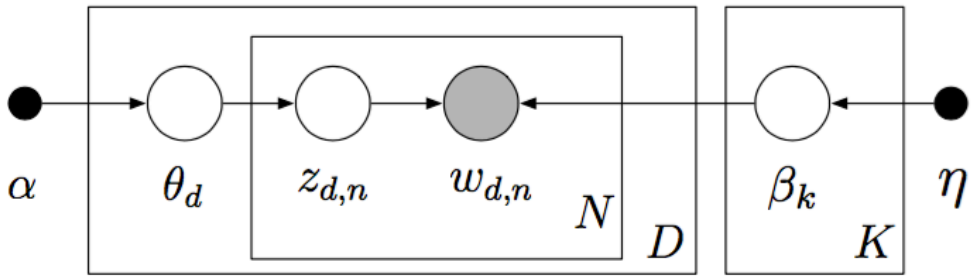
- We have introduced a distribution of the latent variables with free *variational parameters* ν.
- We optimize those parameters to tighten this bound.
- This is the same as finding the member of the family q_ν that is closest in KL divergence to $p(z_{1:M} \mid x_{1:N})$.

KL-Divergence



$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

Why Does LDA Work?

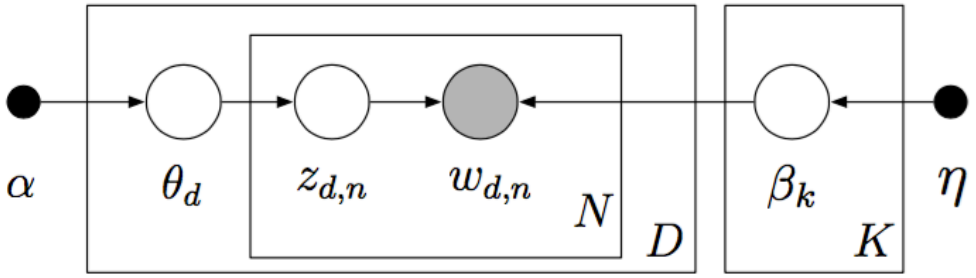


- LDA trades off two goals
  1. In each **document**, allocate its words to **few topics**.
  2. In each **topic**, assign high probability to **few terms**.
- We see this from the joint

$$\log p(\cdot) = \ldots + \sum_d \sum_n \log p(z_{dn} \mid \theta_d) + \log p(w_{dn} \mid \beta_{z_{dn}}) + \ldots$$

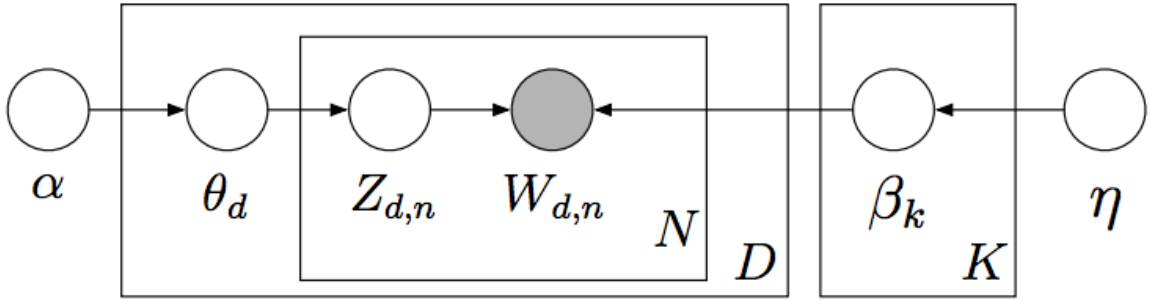
- Sparse proportions come from the 1st term.
  Sparse topics come from the 2nd term.

Why Does LDA Work?



- LDA trades off two goals
  1. In each **document**, allocate its words to **few topics**.
  2. In each **topic**, assign high probability to **few terms**.
- These goals are at odds.
  - Putting a document in a single topic makes #2 hard.
  - Putting very few words in each topic makes #1 hard.
- Trading off these goals finds groups of tightly co-occurring words.

LDA Summary

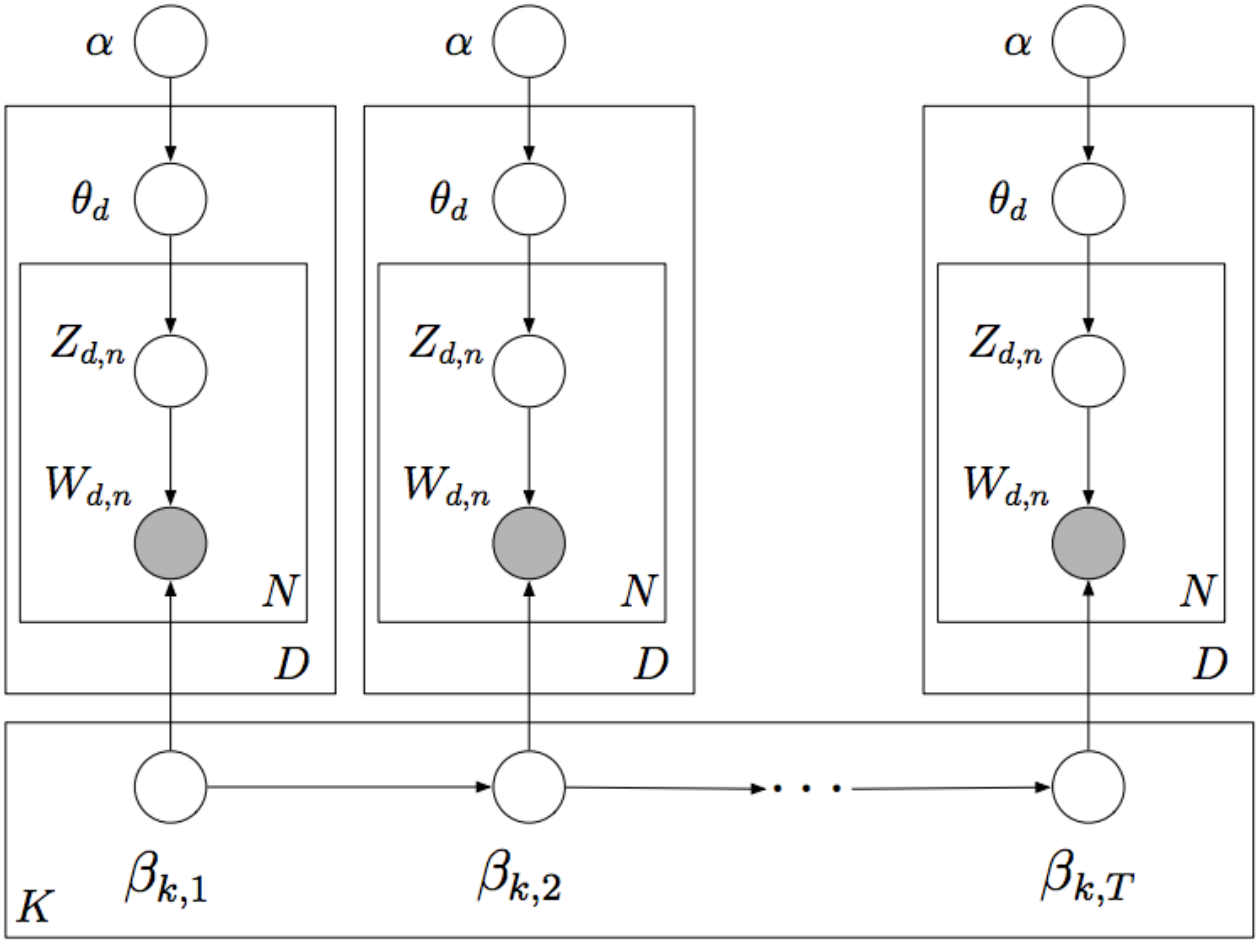


- LDA can
  - visualize the hidden thematic structure in large corpora
  - generalize new data to fit into that structure
- Builds on Deerwester et al. (1990) and Hofmann (1999)
  It is a *mixed membership model* (Erosheva, 2004).
  Relates to *multinomial PCA* (Jakulin and Buntine, 2002)
- Was independently invented for genetics (Pritchard et al., 2000)

Agenda

- ~~What is Topic Modeling?~~
- ~~Parametric vs. Non-Parametric Models~~
- ~~Latent Dirichlet Allocation~~
- ~~Probabilistic Graphical Models~~
- ~~The Effect of the Dirichlet parameter α~~
- Dynamic LDA

Beyond LDA: Dynamic Topic Models

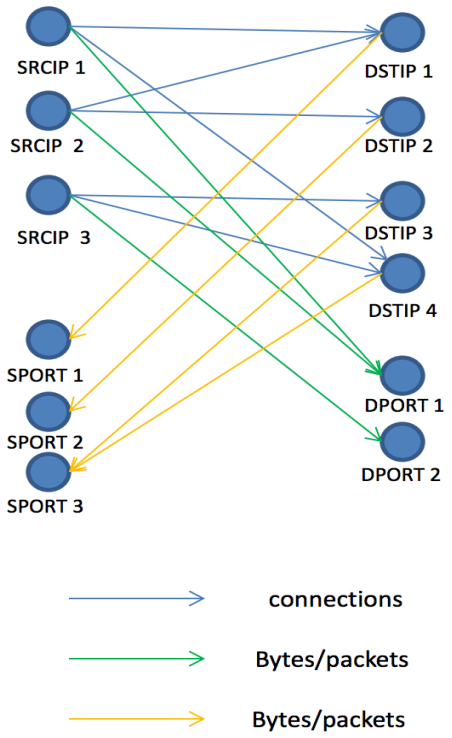


Graphic courtesy https://www.cs.princeton.edu/~blei/kdd-tutorial.pdf

So What are Our Words, Documents, …

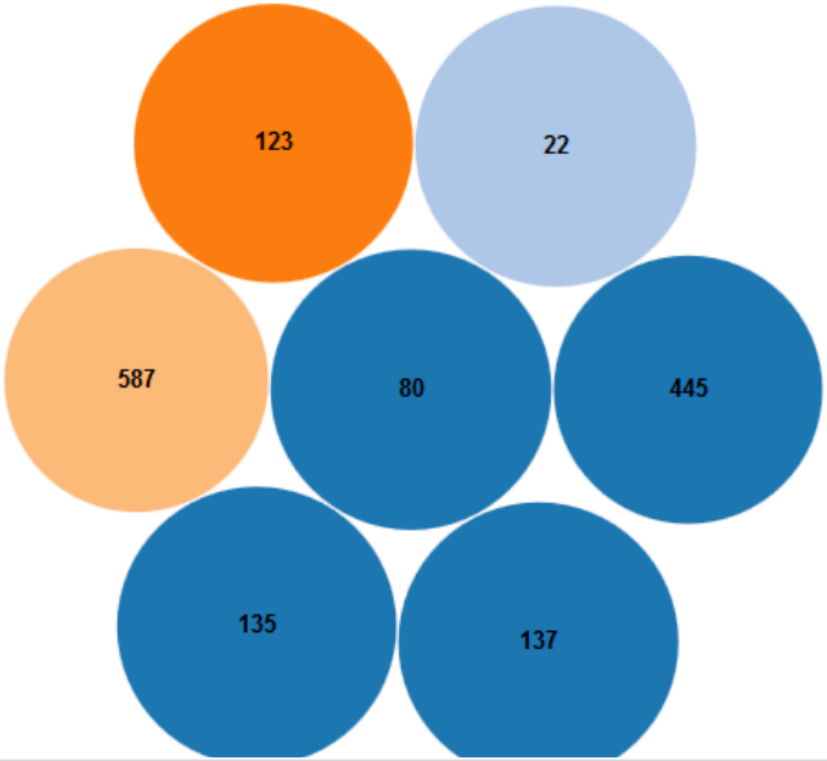
- Documents: Source/Dest IP address pairs
- Words: ports, TCP flags, …

  - Tries to put a population into sub-groups based on their similarity
  - Used with documents and the words in them to suggest “topics”
  - IP addresses are nodes, flow details are edges
  - Use to cluster on known (profiling) or unknown (automated behavior)



What Kinds Of Questions Can We Ask?

- Question: What are the strongest matches for groups based on automated communication to well-known ports ?
- Answer: Seven ports in four different groups are the strongest matches




```
In [ ]: gname = 'netflow_topic'
        g = get_graph(gname)
        graph_result1 = g.query.gremlin("g.V.has('dport').has('lda_result',T.gte,0.9f).has('dport',T.lte,1024)")
        print 'results retrieved'
```

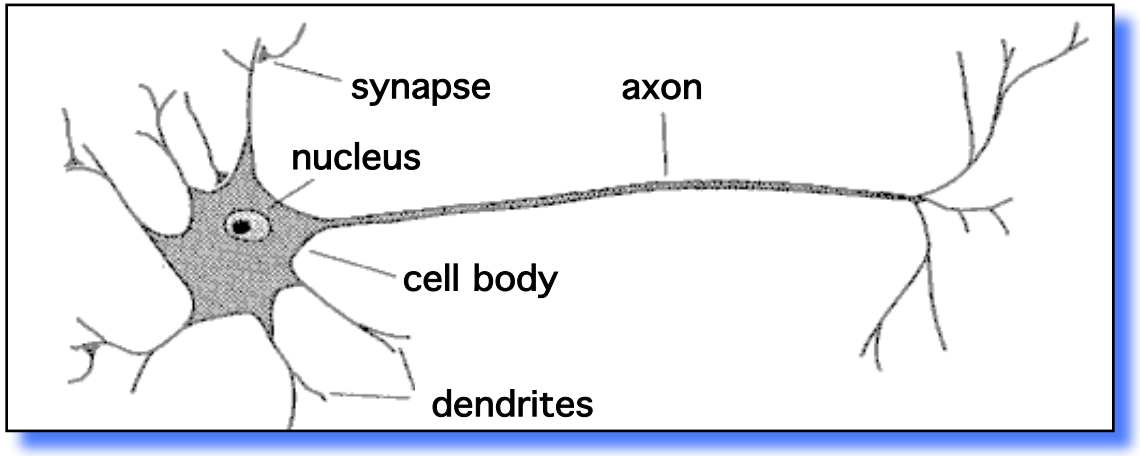
Artificial Neural Networks

- A Bit of History
- Biological Inspiration
- Artificial Neurons (AN)
- Artificial Neural Networks (ANN)
- ~~Computational Power of Single AN~~
- ~~Computational Power of an ANN~~
- Training an ANN -- Learning

Brief History of Neural Networks

- **1943:** McCulloch & Pitts show that neurons can be combined to construct a Turing machine (using ANDs, ORs, & NOTs)
- **1958:** Rosenblatt shows that perceptrons will converge if what they are trying to learn can be represented
- **1969:** Minsky & Papert showed the limitations of perceptrons, killing research for a decade
- **1985:** The backpropagation algorithm revitalizes the field
  - Geoff Hinton et al
- **2006:** The Hinton lab solves the training problem for DNNs

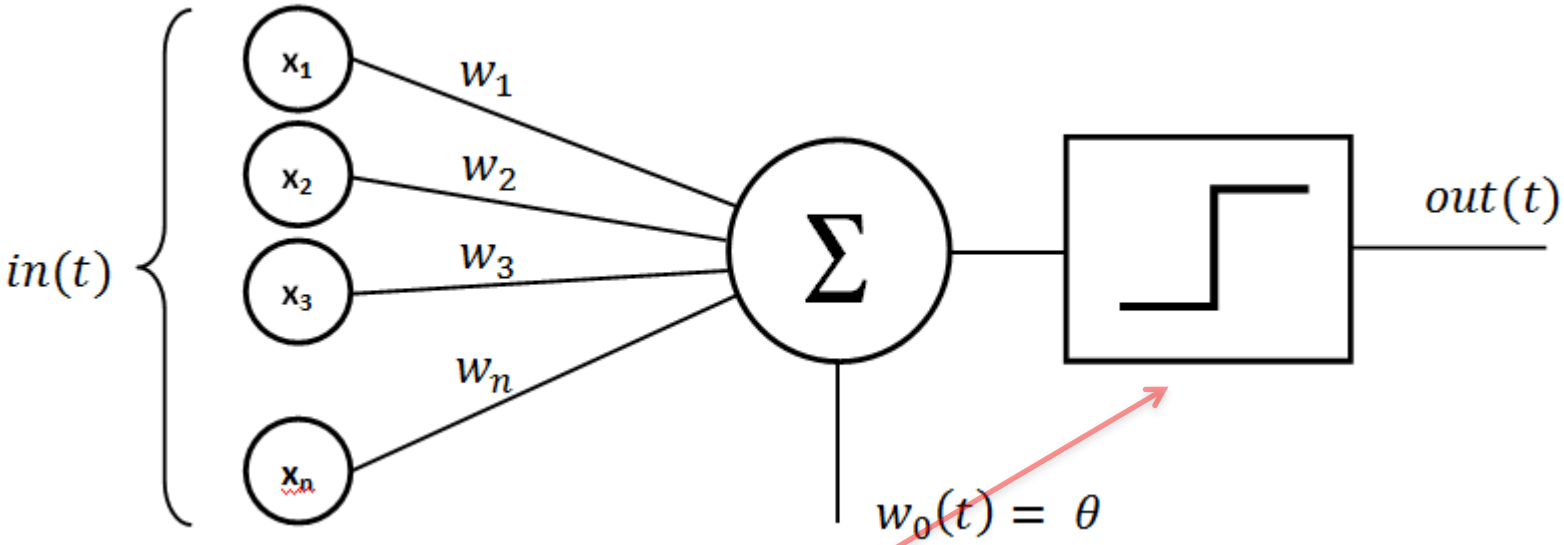
Biological Inspiration: Neurons



- A neuron has
  - Branching input (dendrites)
  - Branching output (the axon)
- Information moves from the dendrites to the axon via the cell body
- Axon connects to dendrites via synapses
  - Synapses vary in strength
  - Synapses may be excitatory or inhibitory

Basic Perceptron

(Rosenblatt, 1950s and early 60s)

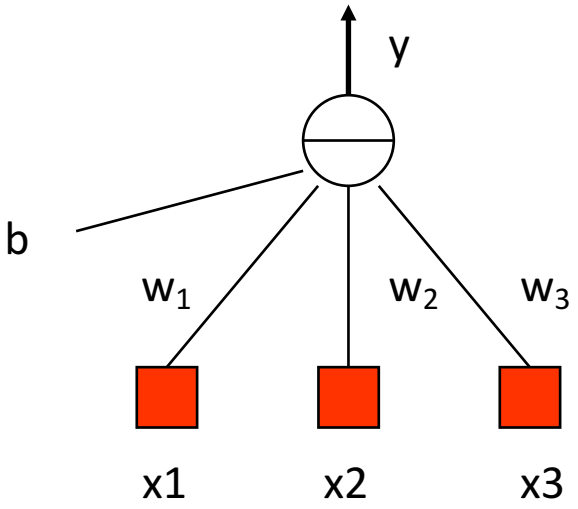


$$O = \begin{cases} 1 : \left(\sum_i w_i x_i \right) + b > 0 \\ 0 : otherwise \end{cases}$$

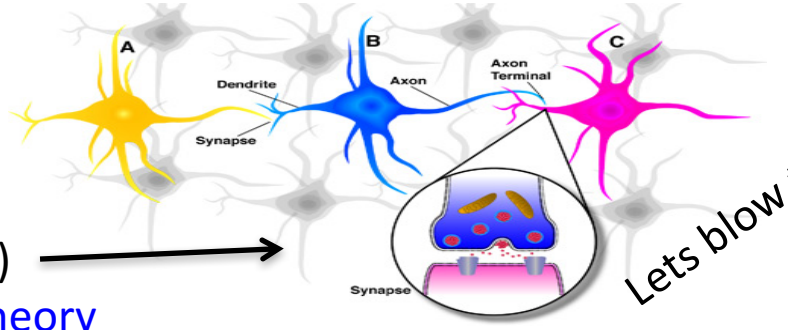
What was the problem here?

What is an Artificial Neuron?

- An Artificial Neuron (AN) is a non-linear parameterized function with restricted output range



$$y = f\left(b + \sum_{i=1}^{n-1} w_i x_i\right)$$

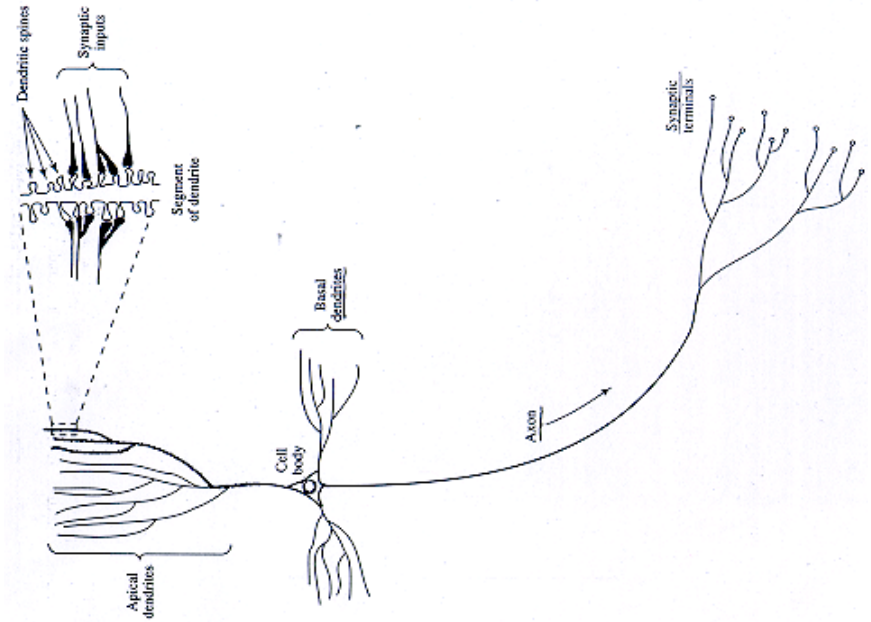


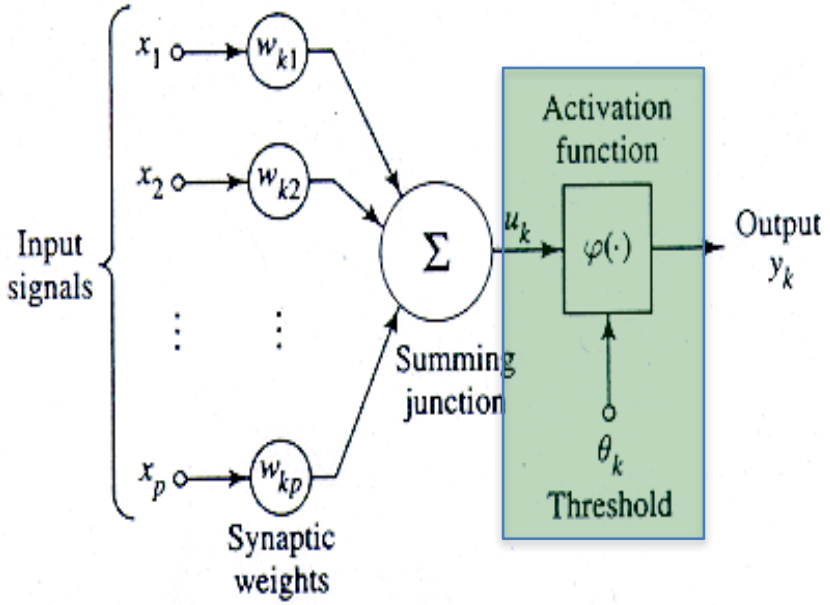
Lets blow this up a bit

Spike-Timing Dependent Plasticity (STDP)

http://en.wikipedia.org/wiki/Hebbian_theory

Mapping to Biological Neurons

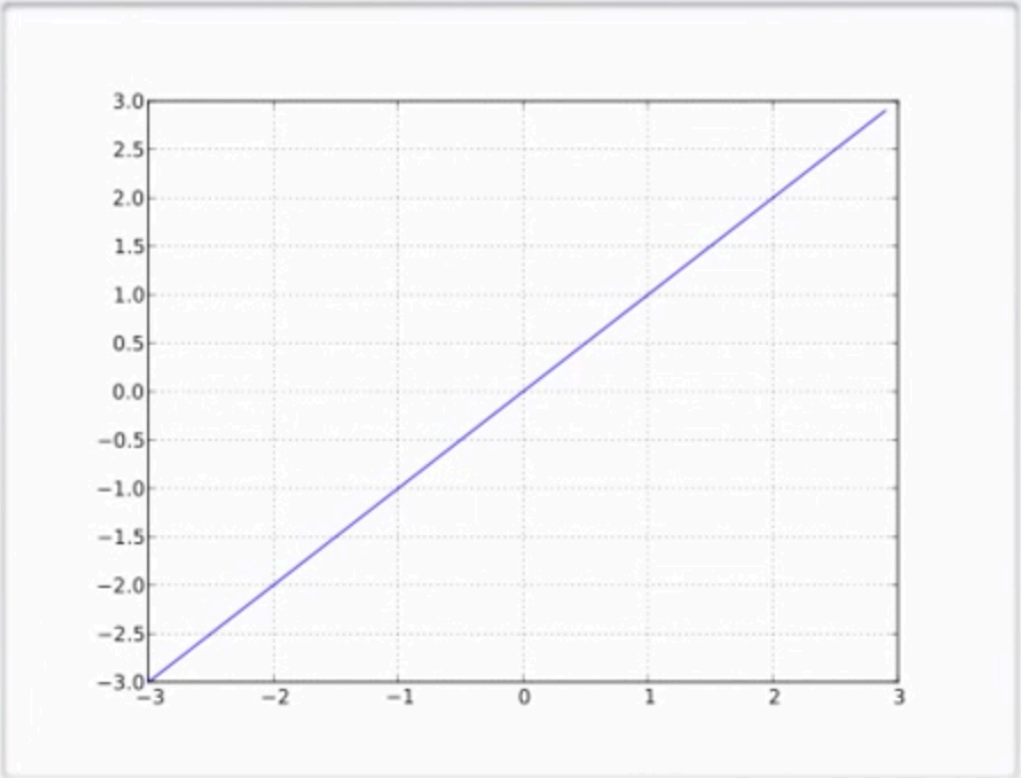




Dendrite Cell Body Axon

Activation Functions – Linear Function

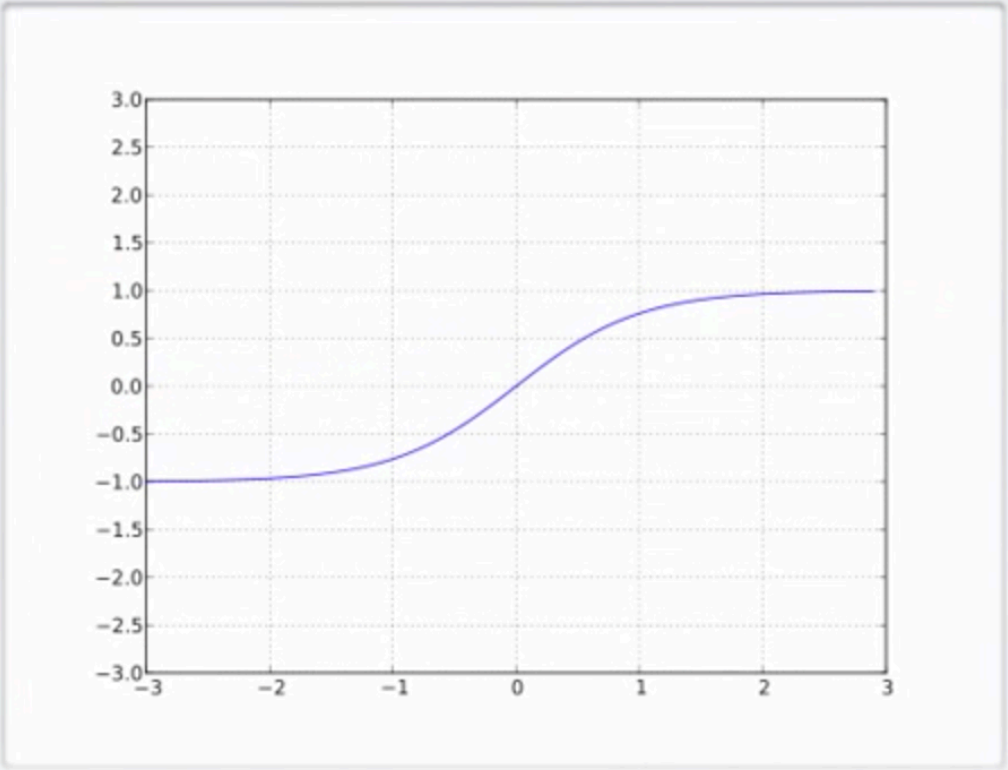
- Performs no input squashing
- Not very interesting...



$$g(a) = a$$

Activation Functions – Hyperbolic Tangent

- Squashes the neuron's input between -1 and 1
- Can be positive or negative
- Bounded
- Strictly increasing



$$g(a) = \tanh(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)} = \frac{\exp(2a) - 1}{\exp(2a) + 1}$$

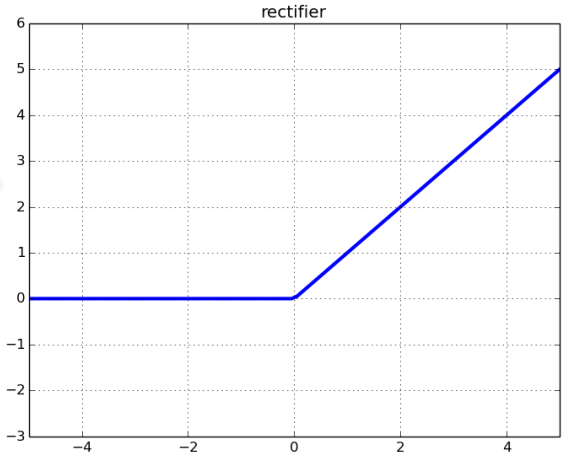
Activation Functions – Sigmoid Function

Recent successes (e.g., Baidu Deep Speech) use (clipped) Rectifier Linear Units:

$h_\theta(x) = g(z)$

$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

f(x) = max(0,x) – rectifier →



f(x) = min(max(0,x), clip)

Smooth approximation (softplus):

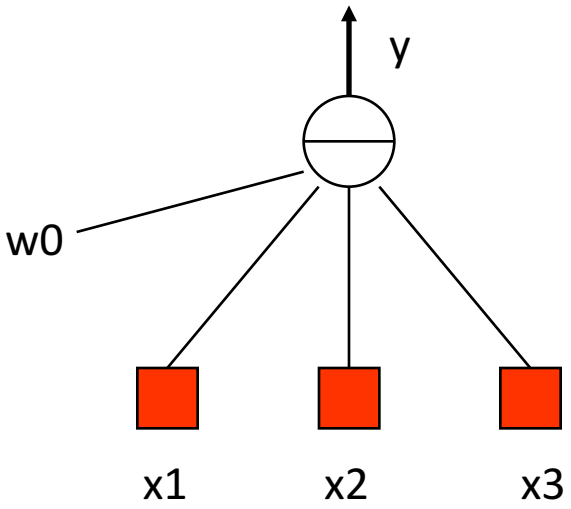
$$f(x) = \log(1 + e^x)$$

Squashes the input (x) onto the open interval [0,1]

$$f'(x) = e^x/(e^x + 1) = 1/(1 + e^{-x})$$

Summary: Artificial neurons

- An *Artificial Neuron* is a non-linear parameterized function with restricted output range



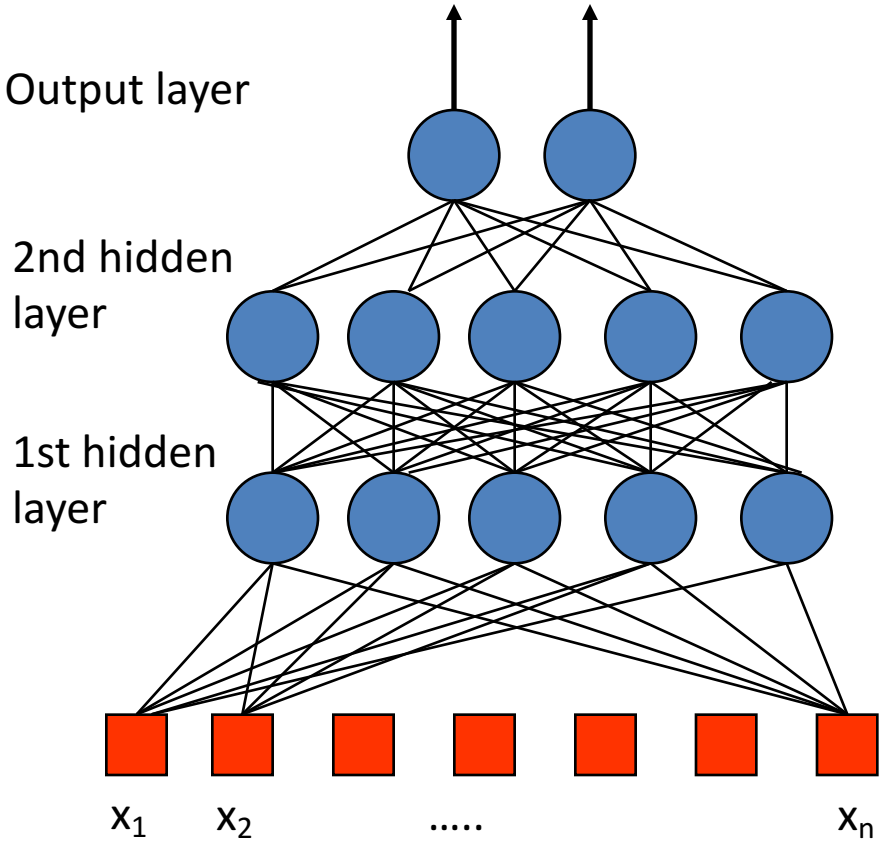
$$y = f\left(w_0 + \sum_{i=1}^{n-1} w_i x_i \right)$$

w_0 also called *a bias term (b_i)*

Ok, Then What is an Artificial Neural Network (ANN)?

- An ANN is mathematical model designed to solve engineering problems
  - Group of highly connected artificial neurons to realize compositions of non-linear functions (usually one of the ones we just looked at)
- Tasks
  - Classification
  - Discrimination
  - Estimation
- 2 main types of networks
  - Feed forward Neural Networks
  - Recurrent Neural Networks

Feed Forward Neural Networks

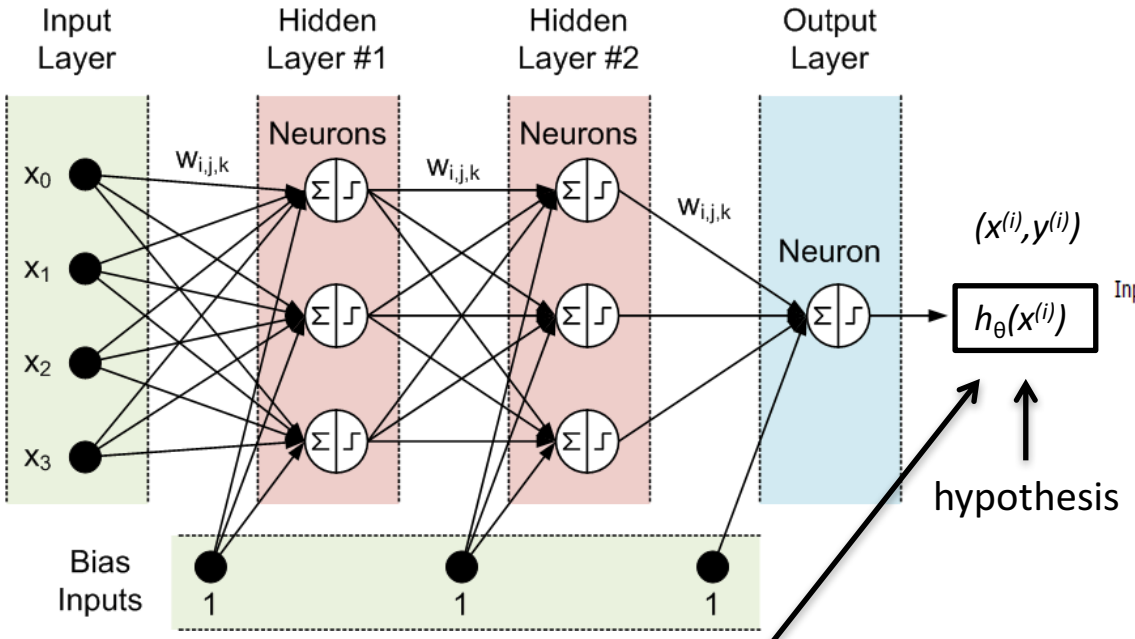


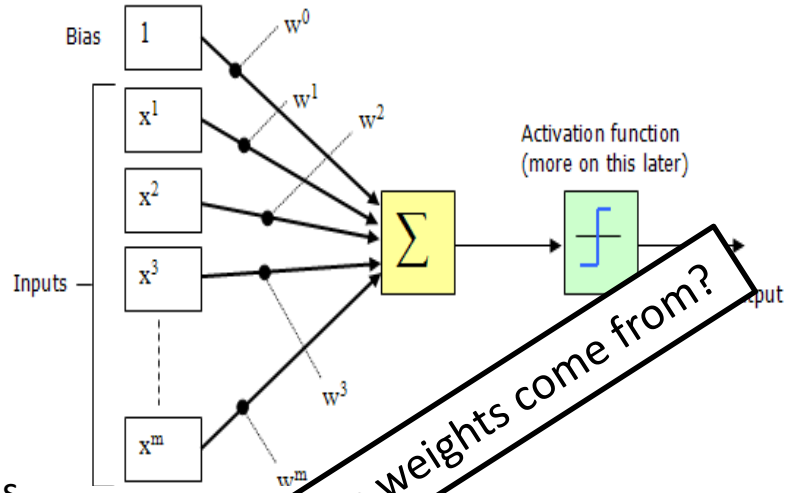
- The information is propagated from the inputs to the outputs
  - Directed Acyclic Graph (DAG)
- Computes one or more non-linear functions
  - Computation is carried out by composition of some number of algebraic functions implemented by the connections, weights and biases of the hidden and output layers
- Hidden layers compute intermediate representations
  - Dimension reduction
- Time has no role -- no cycles between outputs and inputs

We say that the input data, or features, are *n* dimensional

Deep Feed Forward Neural Nets

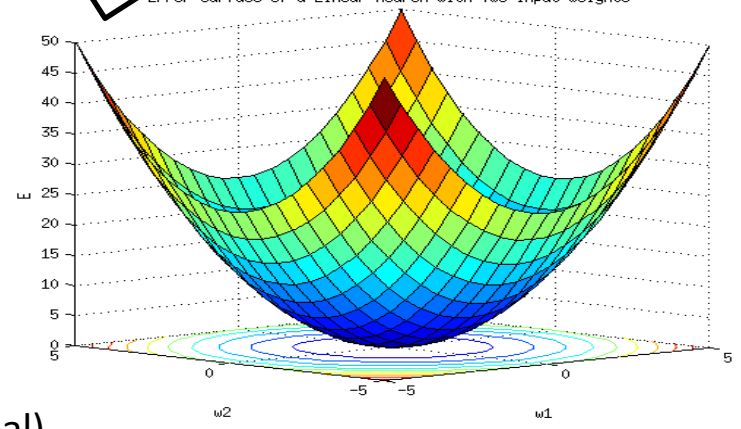
(in 1 Slide (☺))





Forward Propagation

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

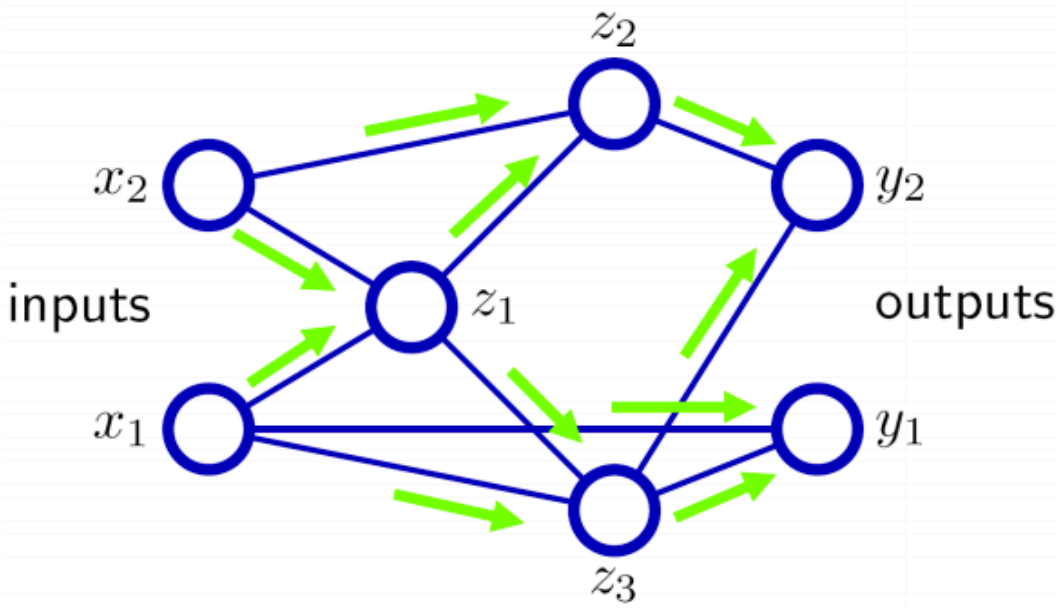


Learning is the adjusting of the weights $w_{i,j}$ such that the cost function $J(\boldsymbol{\theta})$ is minimized (a form of Hebbian learning).

Simple learning procedure: *Back Propagation* (of the error signal)

Forward Propagation Cartoon

- Forward Propagation :
  - Sum inputs, produce activation, feed-forward



Back propagation Cartoon

http://chronicle.com/article/The-Believers/190147

$$) = \sum_{i=1}^{n} y^{(i)} log\left(h_\theta(x^{(i)})\right) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)}))$$

More Formally
Empirical Risk Minimization

- Empirical risk minimization
  - framework to design learning algorithms

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t} l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)}) + \lambda \Omega(\boldsymbol{\theta})$$

  - $l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$ is a loss function (loss function also called "cost function" denoted $J(\theta)$)
  - $\Omega(\boldsymbol{\theta})$ is a regularizer (penalizes certain values of $\boldsymbol{\theta}$)
- Learning is cast as optimization
  - ideally, we'd optimize classification error, but it's not smooth
  - loss function is a surrogate for what we truly should optimize (e.g. upper bound)

Any interesting cost function is complicated and non-convex

Solving the Risk (Cost) Minimization Problem
Gradient Descent – Basic Idea

Have some function $J(\theta_0, \theta_1)$

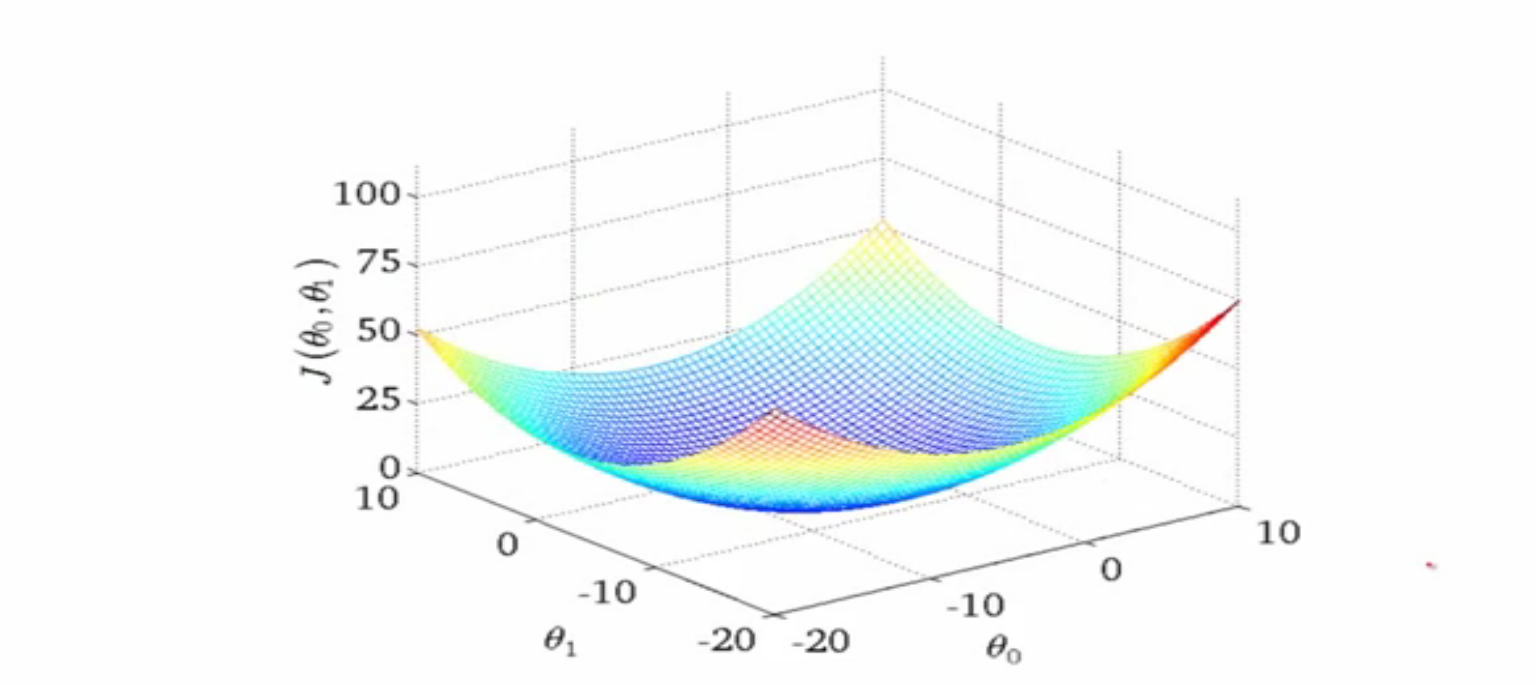
Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

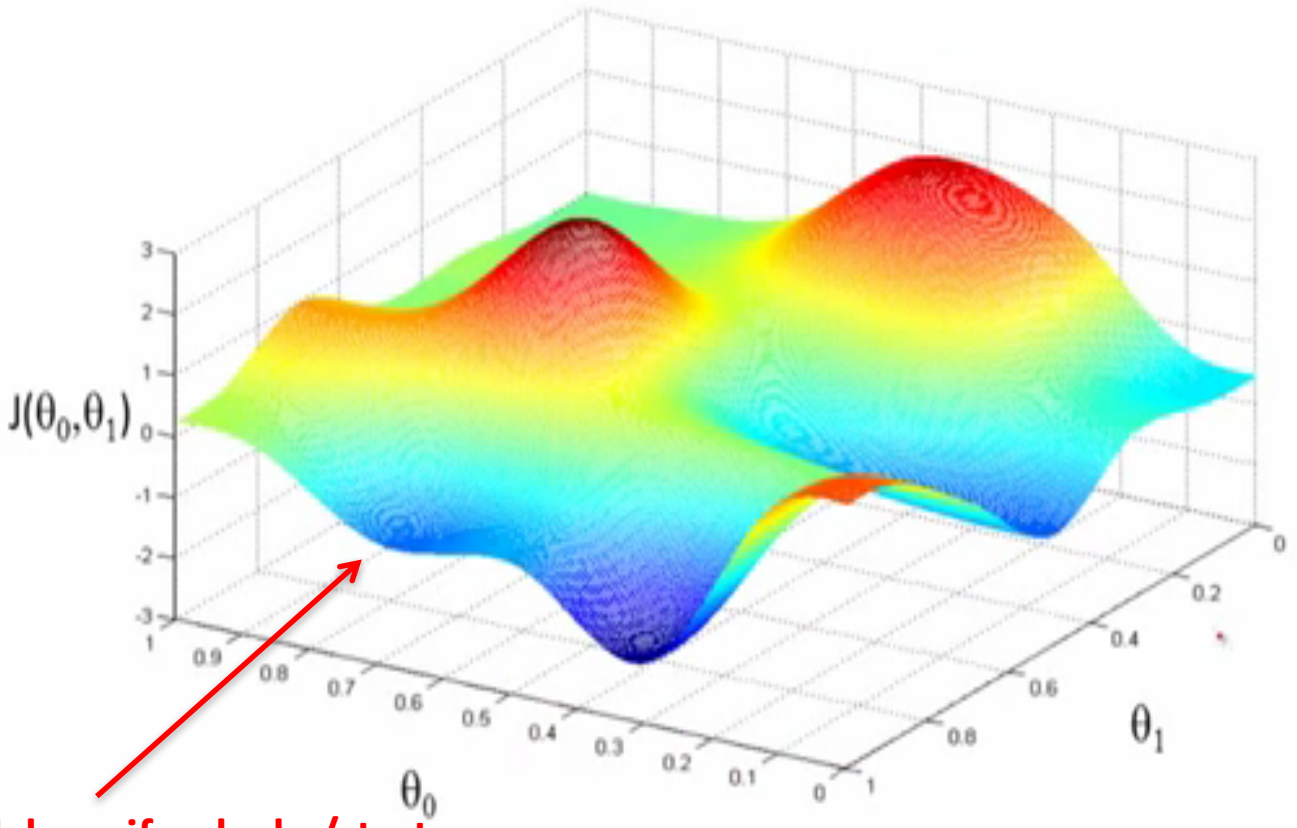
  until we hopefully end up at a minimum

Gradient Descent Intuition 1
Convex Cost Function



One of the many nice properties of convexity is that any local minimum is also a global minimum

Gradient Decent Intuition 2



Can get stuck here if unlucky/start at the wrong place

Unfortunately, any interesting cost function is likely non-convex

Solving the Optimization Problem
Gradient Descent for Linear Regression

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

}

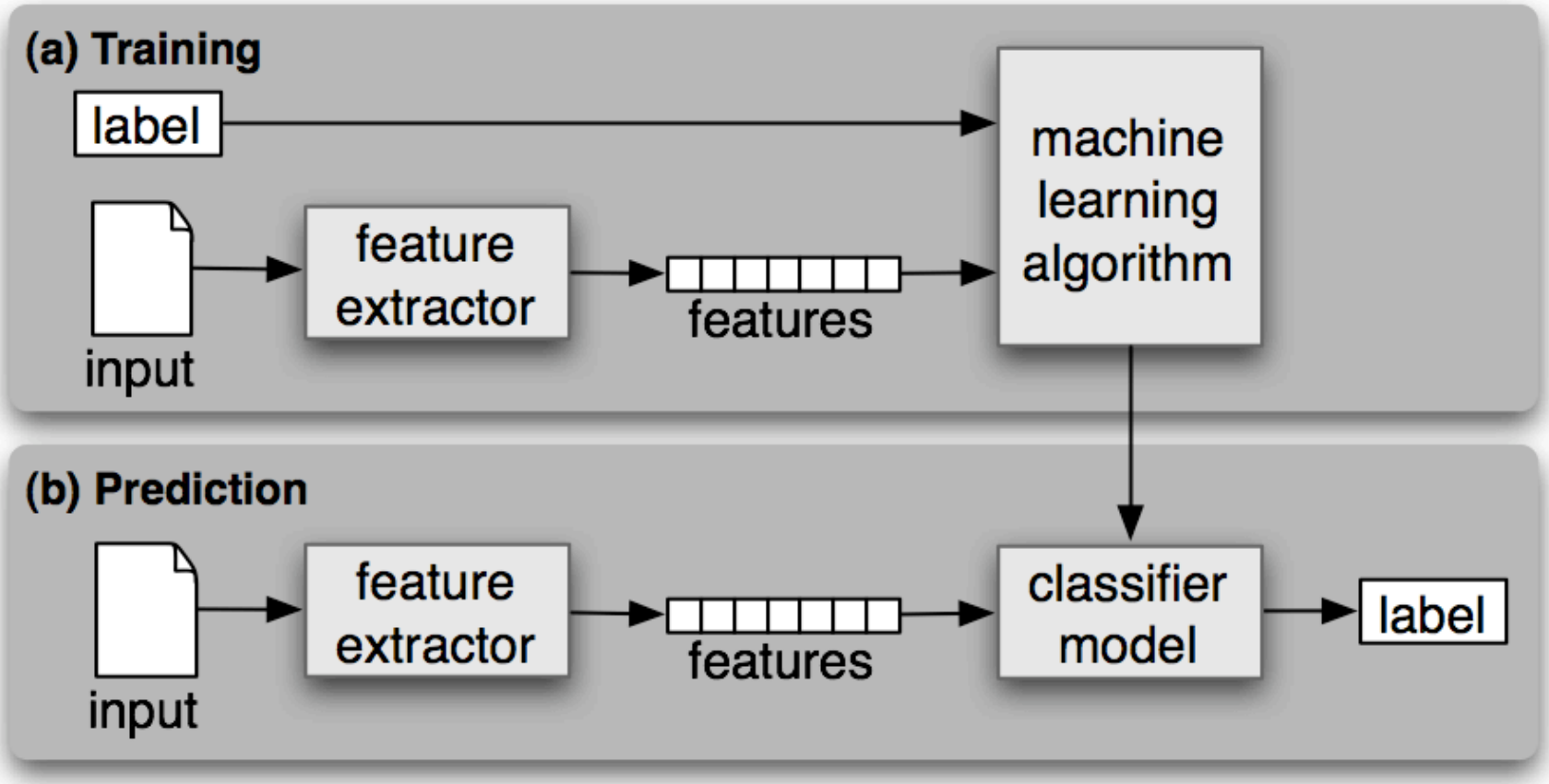
Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

The big breakthrough came from the Hinton lab at UToronto in the mid 80's where the back propagation algorithm was discovered (or perhaps re-discovered). "Backprop" is a simple way of computing the gradient of the loss function with respect to the model parameters θ

Summary: Supervised Learning Process



Making this Real
(what do we have to do?)

- Choose the labels of interest
  - What are the classes of interest, what might we want to classify/compute/predict?
- Get the data sets (this is always the "trick")
  - Labeling?
  - Split into training, test, cross-validation
    - Avoid generalization error (bias, variance)
  - Avoid data leakage
- Choose a model
  - I would try supervised DNN
    - Extremely powerful model
    - We want to find "non-obvious" features, which likely live in high-dimensional space
- Write code
  - Then write more code
- Test on (previously) unseen examples
- Iterate

Issues/Challenges

- Is there a unique model that we can use?
  - Unlikely → online learning
  - Ensemble learning, among others
- Network data is *non-perceptual* (we think)
  - Does the Manifold Hypothesis hold for non-perceptual data sets?
  - Seems to (Google PUE, etc)
- Unlabeled vs. Labeled Data
  - Most commercial successes in ML have come with deep supervised learning → labeled data
  - We don't have ready access to large labeled data sets (always a problem)
- Time Series Data
  - With the exception of Recurrent Neural Networks, most ANNs do not explicitly model time (e.g., Deep Neural Networks)
  - Flow data/sampling
- Training vs. {prediction,classification} Complexity
  - Stochastic (online) vs. Batch vs. Mini-batch
  - Where are the computational bottlenecks, and how do those interact with (quasi) real time requirements?

Agenda

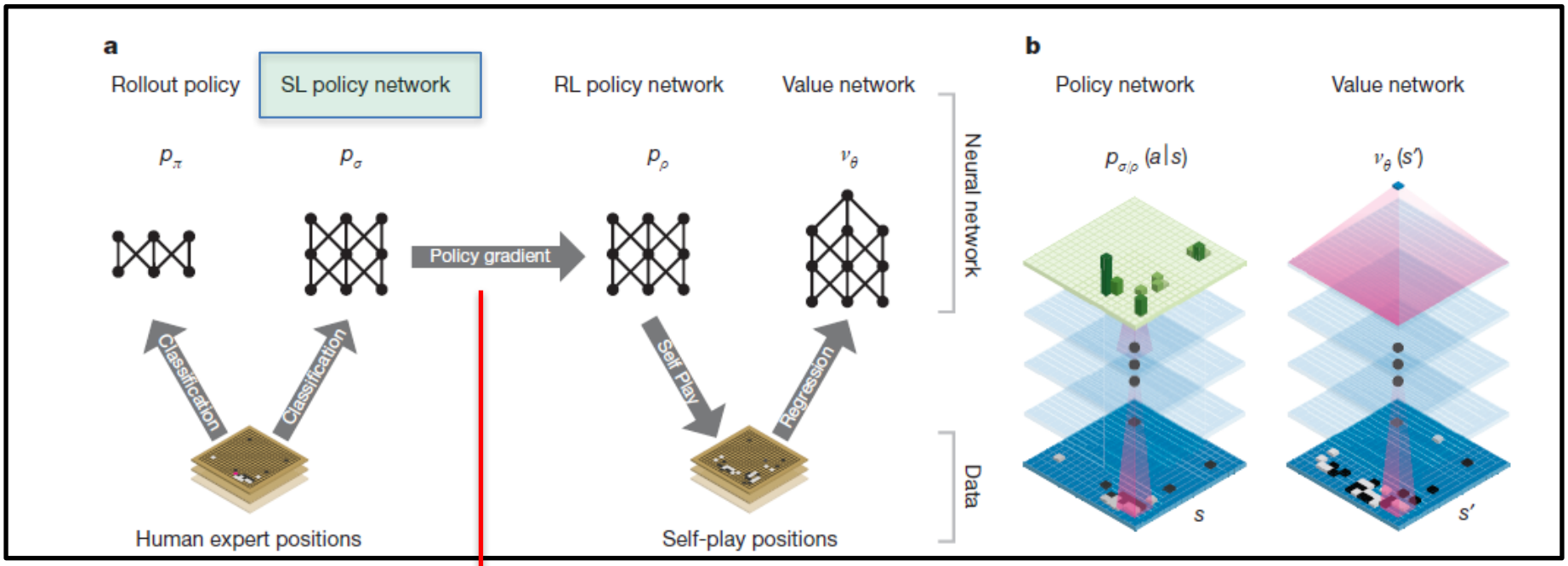
- Who Am I?
- Level Set: What Is Machine Learning?
- What is all the (Machine Learning) Excitement About?
- Integrated Approaches
- Histograms, LDA, and Neural Networks
- Machine Learning Excitement Redux: Beyond Static Learning
- Technical explanations/code
  - https://github.com/davidmeyer/ml

What is All the Excitement About, Redux

***Reinforcement learning* meets Deep Learning and Monte Carlo Tree Search**

TPUs AlphaGo Used To Beat Lee Sedol

One of the many ”AlphaGo Breakthroughs”



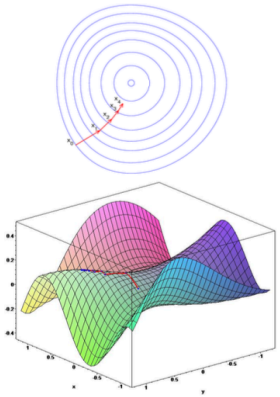
- Let $J(\theta)$ be any policy objective function
- Policy gradient algorithms search for a *local* maximum in $J(\theta)$ by ascending the gradient of the policy, w.r.t. parameters θ

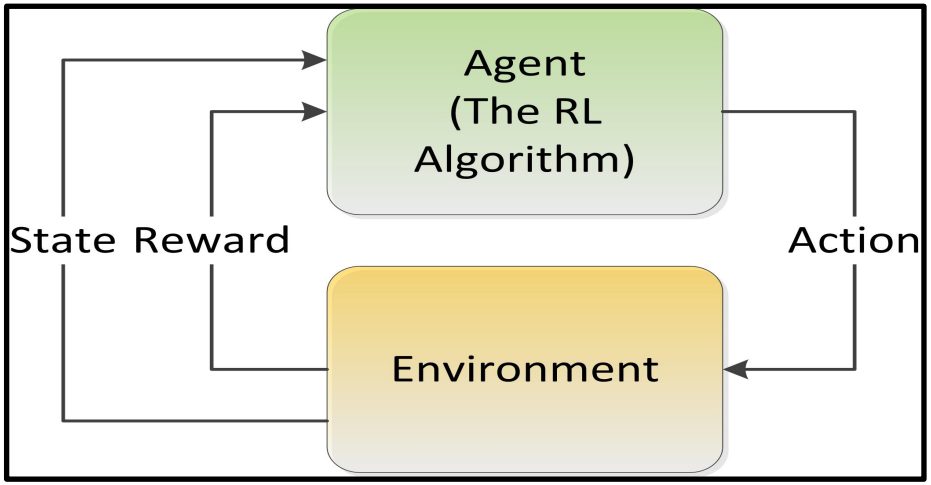
$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$

- Where $\nabla_\theta J(\theta)$ is the policy gradient

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$

- and α is a step-size parameter

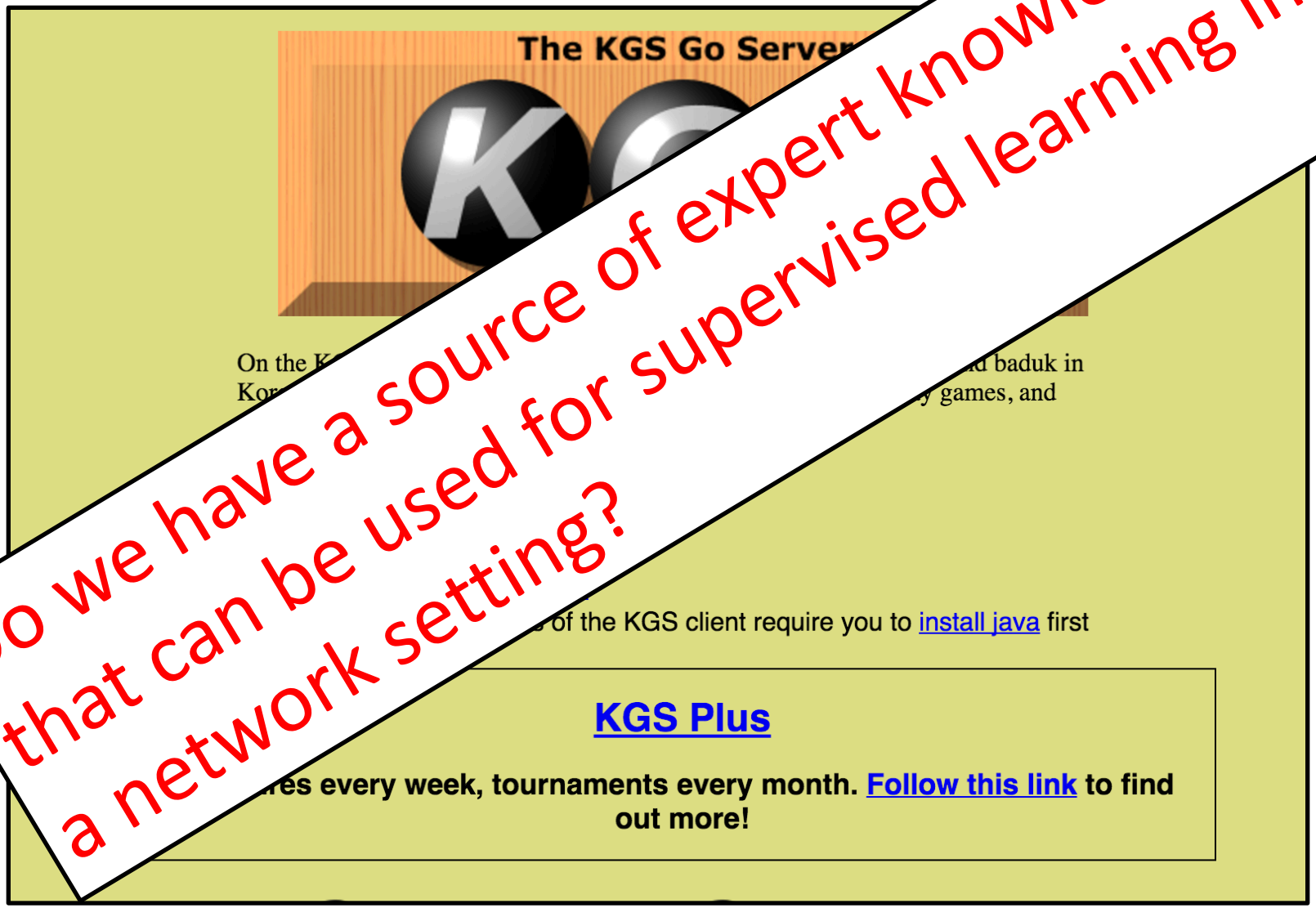




http://www.1-4-5.net/~dmm/ml/log_derivative_trick.pdf

Training the SL Policy Network

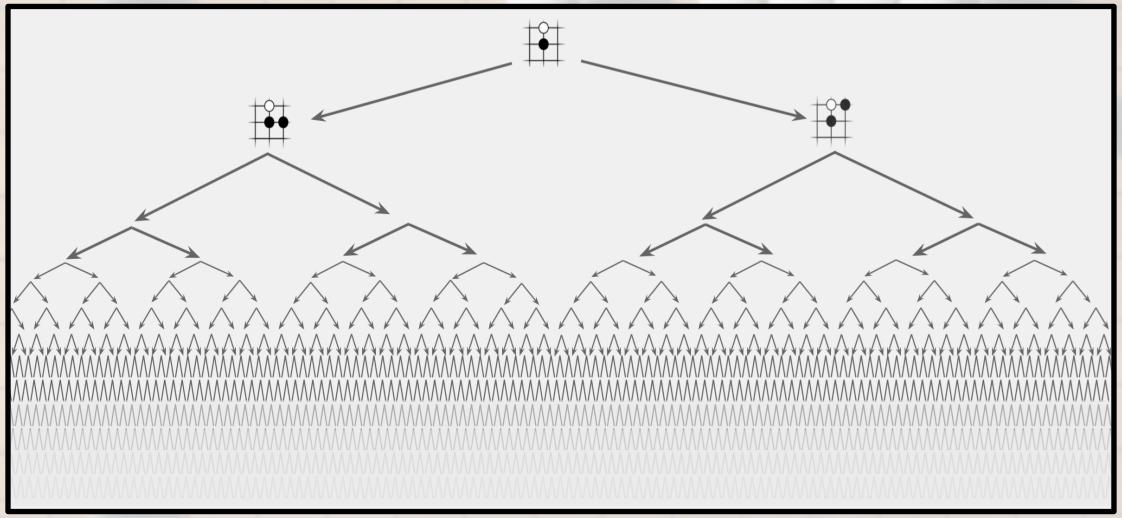
Deep Convolution Neural Network Captures the *Intuition* of Ex



Do we have a source of expert knowledge that can be used for supervised learning in a network setting?

BTW, Why Is *Go* So Hard?

- Game Tree Complexity = b^d



- Brute Force Search Intractable
  - Search space is huge (10^721)
  - Difficult to evaluate who is winning

- Can we characterize network state and action spaces?
  - And why is this important?

Typical Workflow Schematic

Importantly, control is not learned

Domain Knowledge

Analytics Platform

Presentation Layer

Learning

Intelligence

Data Collection
Packet brokers, flow data, …

Preprocessing
Big Data, Hadoop, Data Science, …

Learning

Inference

Remediation/Optimization/…

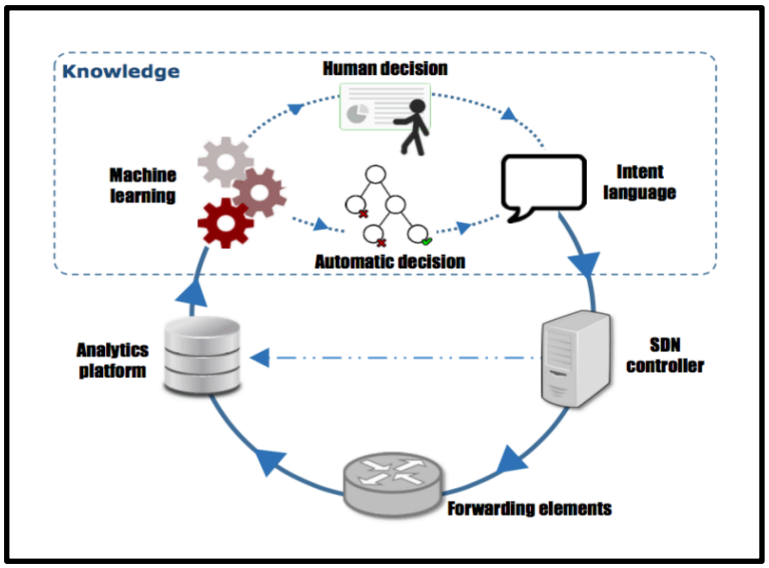
Business Logic

Intent

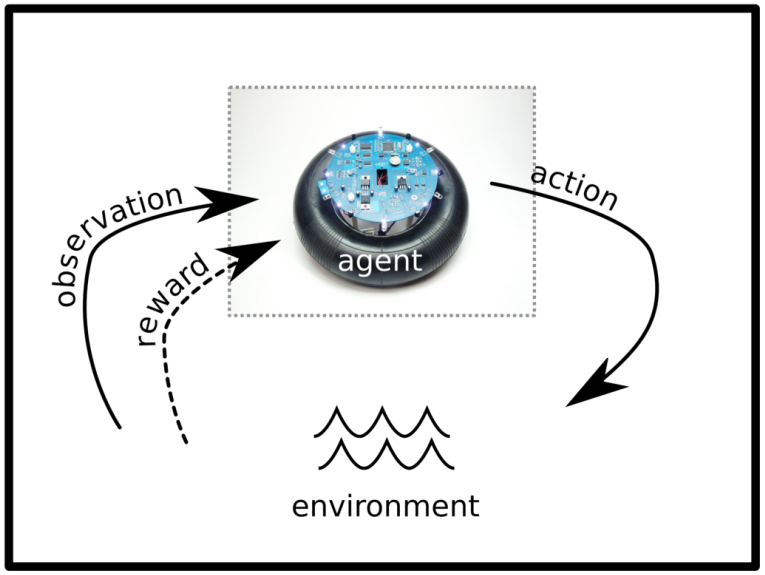
Recommender Sytems

Topology, Anomaly Detection, Root Cause Analysis, Predictive Insight, ….

Static vs. Reinforcement Learning Architectures



- Today's Static ML Architectures
  - ML doesn't have "agency"
  - Hard-coded or open loop control
- Doesn't learn control
- Assumes stationary DGD
- Largely off-line



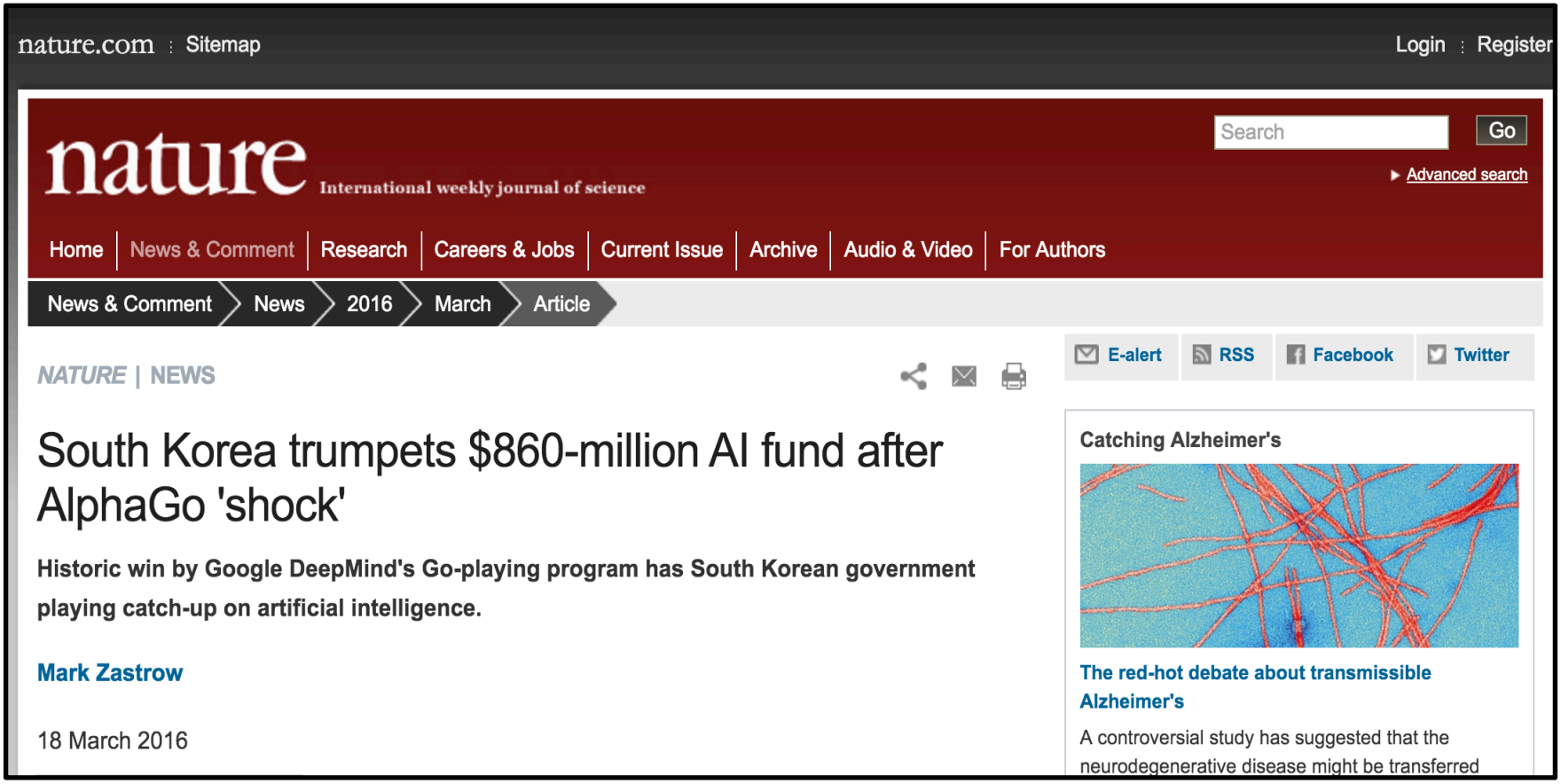
- Reinforcement Learning Architecture
  - Agent architecture
  - Agent learns control/action selection
- Adapts to evolving environment
- Non-stationary distributions
- *Network gamification*

Network Gamification?

- *Gamification* is the application of game theoretic approaches and design techniques to what have traditionally been non-game problems, including business, operations and social impact challenges
  - Online learning/optimization
  - Security
  - Automation
  - Almost any application that interacts with its environment

- Idea: Envision network and other automation tasks as 2-player games, and use Deep Learning, Reinforcement Learning and Monte Carlo Tree Search to learn optimal control in an online setting

Summary: Why Is AlphaGo Important/Relevant?



nature.com : Sitemap

Login : Register

nature International weekly journal of science

Search Go

Advanced search

Home | News & Comment | Research | Careers & Jobs | Current Issue | Archive | Audio & Video | For Authors

News & Comment > News > 2016 > March > Article

E-alert RSS Facebook Twitter

NATURE | NEWS

South Korea trumpets $860-million AI fund after AlphaGo 'shock'

Historic win by Google DeepMind's Go-playing program has South Korean government playing catch-up on artificial intelligence.

Mark Zastrow

18 March 2016

Catching Alzheimer's

The red-hot debate about transmissible Alzheimer's

A controversial study has suggested that the neurodegenerative disease might be transferred

- Security: Agent can learn dynamic/evolving behavior of adversary
- DevOPs: Agent can learn workflow automation (e.g. Openstack Mistral/StackStorm)
- Orchestration: Agent can learn dynamic behavior of VNFs and system as a whole
- General: Deep learning can capture human intuition

Example: Gamifying Workflows

```
---
chain:
    -
        name: "c1"
        ref: "core.local"
        parameters:
            cmd: "echo c1"
        on-success: "c2"
        on-failure: "c4"
    -
        name: "c2"
        ref: "core.local"
        parameters:
            cmd: "echo \"c2: parent exec is {{action_context.parent.execution_id}}.\""
        on-success: "c3"
        on-failure: "c4"
    -
        name: "c3"
        ref: "core.local"
        parameters:
            cmd: "echo c3"
        on-failure: "c4"
    -
        name: "c4"
        ref: "core.local"
        parameters:
            cmd: "echo fail c4"
default: "c1"
```

Boolean state test/actions

- Workflows can be learned/optimized
- Model as a POMDP: <S,A,T,R,Ω,O,γ>
- Estimate with MDP: <S,A,T,R,γ>
  - Model free
- Deep Value (Q) and Policy (π) networks

State Transitions

$$\pi(s) = a$$
$$s \in S, a \in A(s)$$

https://github.com/StackStorm

https://gym.openai.com/

https://github.com/StackStorm/st2/blob/master/contrib/examples/actions/chains/echochain.yaml

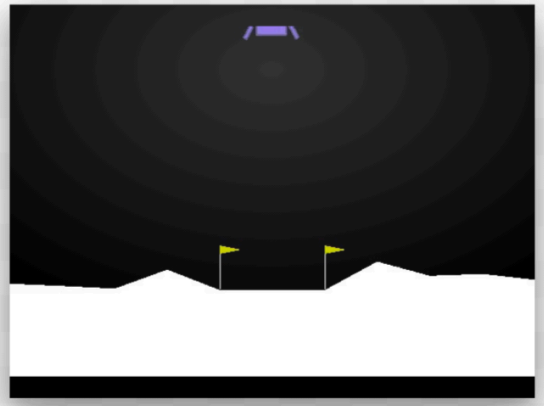
Interested in Reinforcement Learning?

OpenAI Gym BETA

A toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Go.

Read the launch blog post ›
View documentation ›
View on GitHub ›

jcoreyes's algorithm on Breakout-v0



ceobillionaire's algorithm on LunarLander-v1

One of the Many Important Things OpenAI is Doing

Algorithms Environments Docs Chat Credits

Sign in with GitHub

Breakout-v0

Maximize your score in the Atari 2600 game Breakout. In this environment, the observation is an RGB image of the screen, which is an array of shape (210, 160, 3) Each action is repeatedly performed for a duration of k frames, where k is uniformly sampled from $\{2, 3, 4\}$.

Breakout-v0 is an unsolved environment, which means it does not have a specified reward threshold at which it's considered solved.

The game is simulated through the Arcade Learning Environment [ALE], which uses the Stella [Stella] Atari emulator.

jcoreyes's algorithm, 86.95 ± 5.10 2 months ago

Breakout-v0 Evaluations

ALGORITHM	BEST 100-EPISODE PERFORMANCE	SUBMITTED
tambetm's algorithm writeup	105.27 ± 7.02	a month ago
jcoreyes's algorithm writeup	86.95 ± 5.10	2 months ago
carpedm20's algorithm writeup	67.43 ± 4.14	a month ago
carpedm20's algorithm writeup	67.35 ± 3.59	a month ago

Agenda

- Who Am I?
- Level Set: What Is Machine Learning?
- What is all the (Machine Learning) Excitement About?
- Integrated Approaches
- Neural Networks
- Machine Learning Excitement Redux: Beyond Static Learning
- Technical explanations/code
  - https://github.com/davidmeyer/ml
  - http://www.1-4-5.net/~dmm/ml

Q&A

Thank you