Anomaly Detection for the IDEAS Mapping System

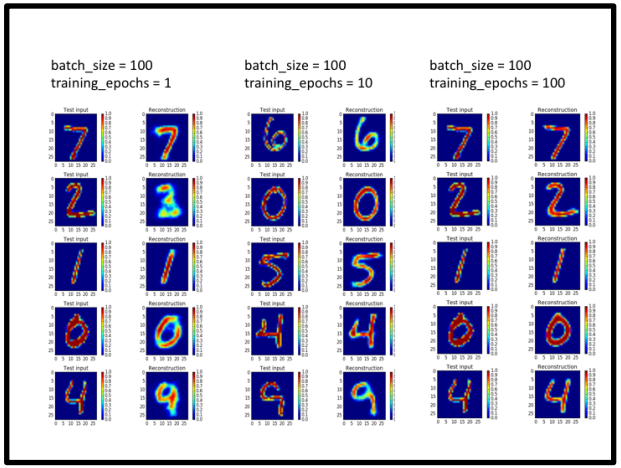
A Machine Learning Approach

David Meyer
Brocade Chief Scientist, VP and Fellow
Senior Research Scientist, Computer Science, University of Oregon
dmm@{brocade.com,uoregon.edu,1-4-5.net,..}

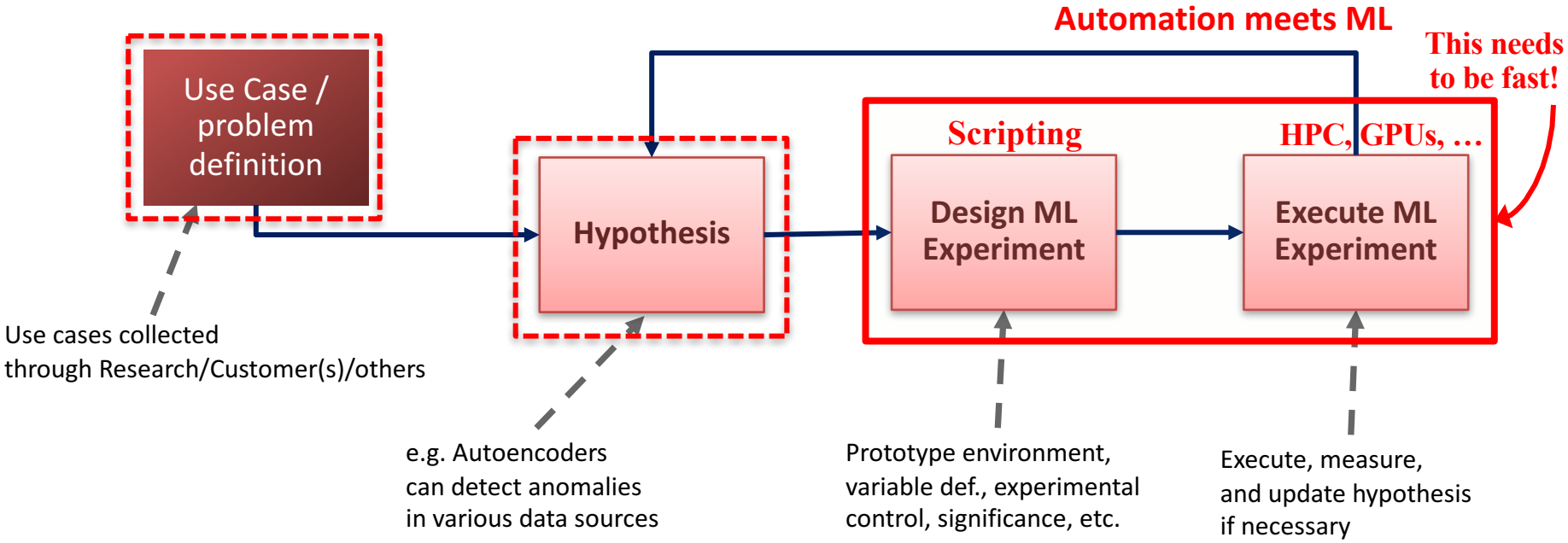
IDEAS Kickoff Meeting
IETF 97
13 – 18 Nov 2016
Seoul, Republic of Korea

Agenda

- Potential Use Cases
- What Can We Do Today, And What Are The Challenges?
- Discussion
- Technical explanations/code
  - https://github.com/davidmeyer/ml
  - http://www.1-4-5.net/~dmm/ml



First, What Does The Scientific Method Look Like When Applied To Machine Learning?



"If you want to increase your success rate, double your failure rate"

Thomas Watson Sr. (founder of IBM)

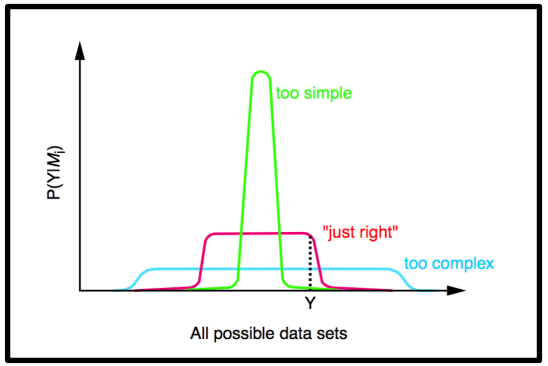
Original slide courtesy Armin Wasicek

Prototype Use Case

- We want to use Machine Learning (ML) to
  - Detect anomalous traffic flows that could be malicious
    - e.g., DDoS against the mapping infrastructure
  - Want to protect the Map Servers and Resolvers
    - and other system components that could benefit
    - Map-Registers, Map-Requests, Map-Replies

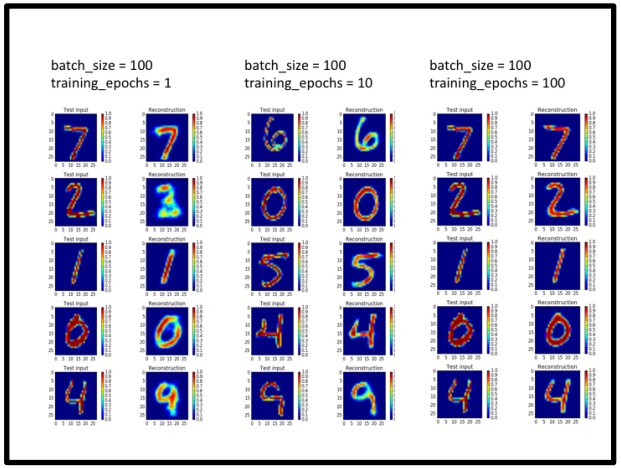
- How might we do this?
  - Collect KPIs from the Map {Server,Resolver} and surrounding infrastructure
  - ETL the data
  - Use ML to Detect Anomalies
  - Remediate
  - Iterate

- Proposal:
  - Occam's Razor: Do the simplest thing first
  - Use a simple autoencoder to do binary classification

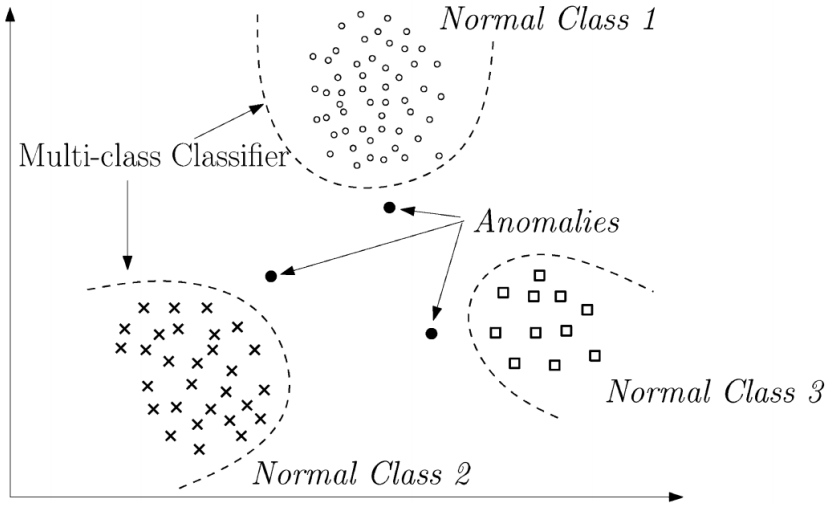


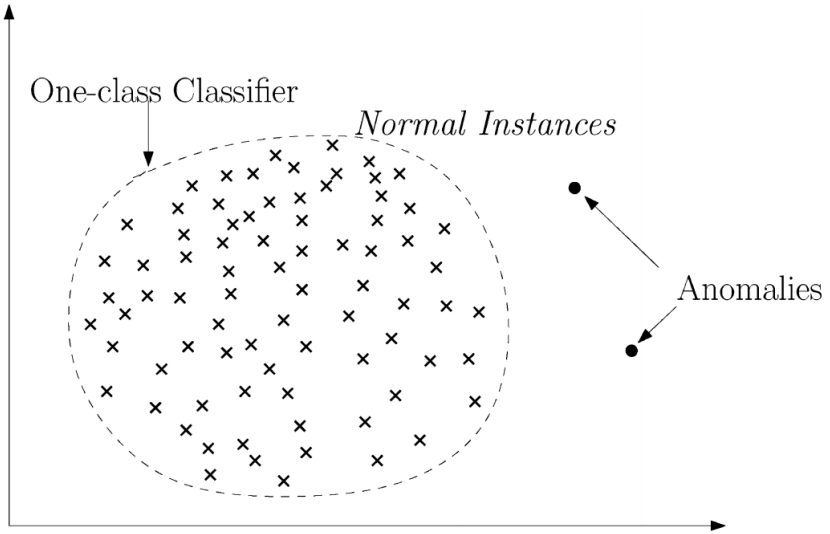
Agenda

- What is the Use Case?
- What Can We Do Today, And What Are The Challenges?
- Discussion
- Technical explanations/code
  - https://github.com/davidmeyer/ml
  - http://www.1-4-5.net/~dmm/ml



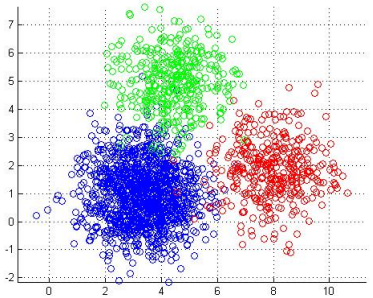
Briefly, What is Anomaly Detection?





What's going on here?

- Data don't fit an explanation model
  - Impossible, assuming the model is correct
- Data do not conform to some *normal* behavior



Clustering
- K-means
- K-NN
- …

PCA
Autoencoders
….

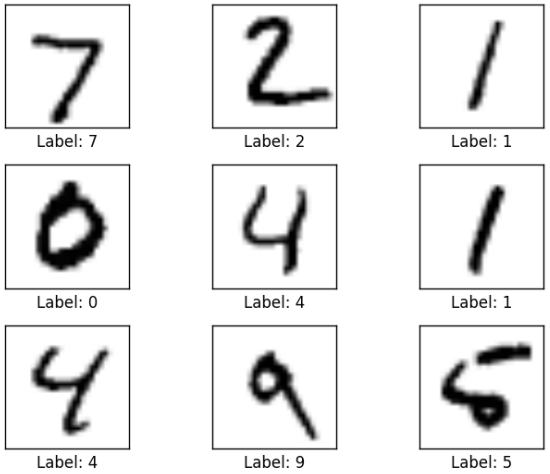
Dimension Reduction

We assume that the anomalous data are generated by a different process than our *baseline* → *stationary distribution*

Original slide courtesy Armin Wasicek

Autoencoder Example

Detecting Anomalies in Handwritten Digits



MNIST: 28x28 Grayscale Handwritten Digit Dataset

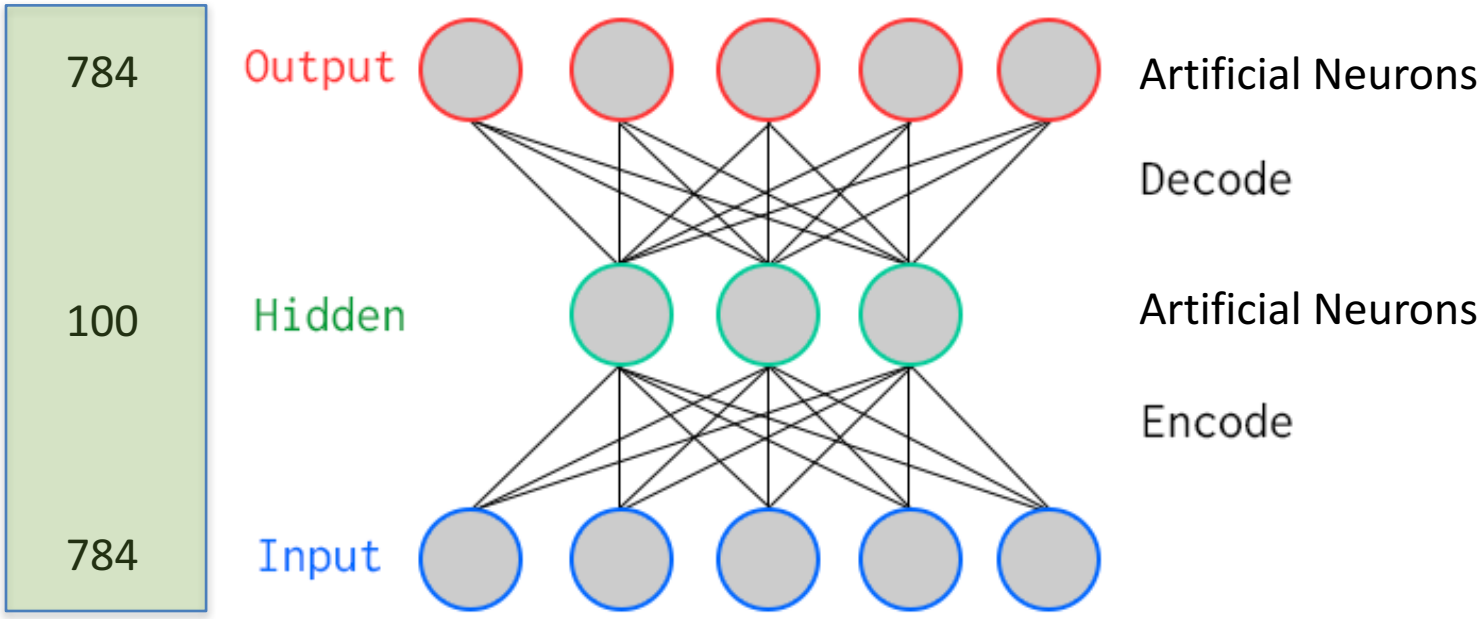
- Training set: 55000 images
- Test set: 10000 images
- Validation set: 5000 images

Features here are pixels (28x28 = 784)

I'm using MNIST here because you easily visualize what is going on. In our case, instead of the input being vectors of {0,1} we will have vectors of counters of map control messages, bandwidth, and other host based KPIs.

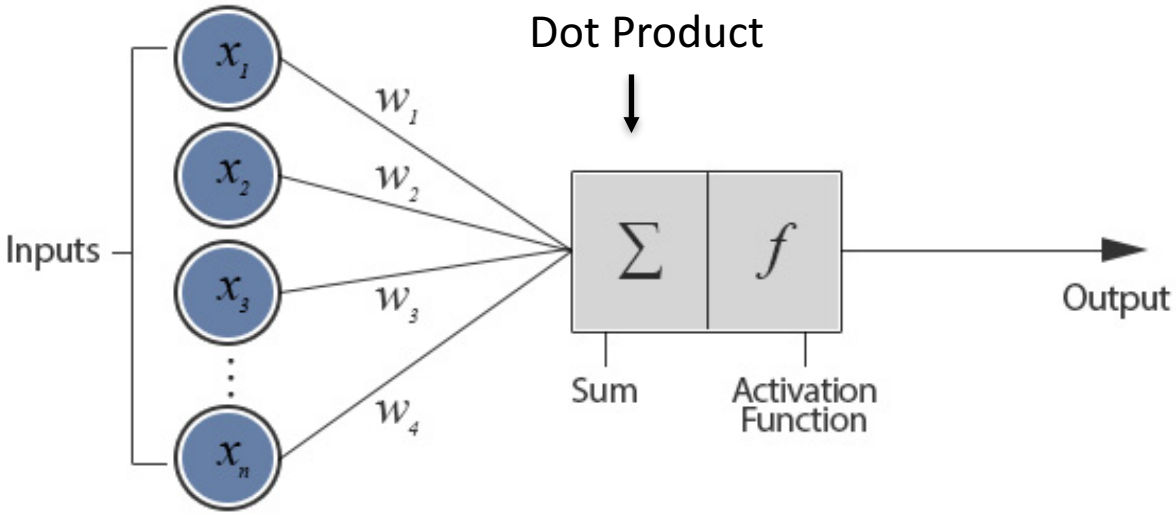
One Way To Learn To Recognize MNIST: Use An Autoencoder

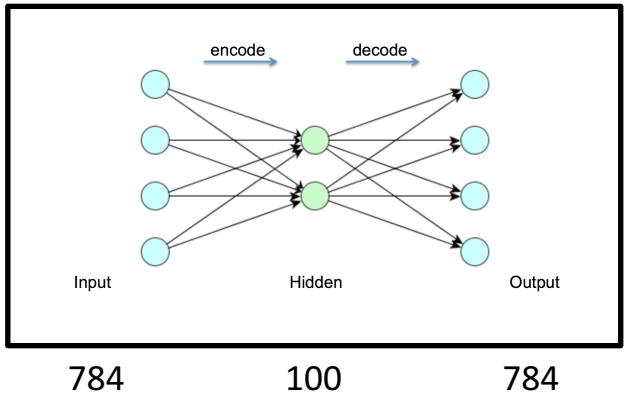
Special Kind of Neural Network



- 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

But First: What Does A Single Neuron Do?



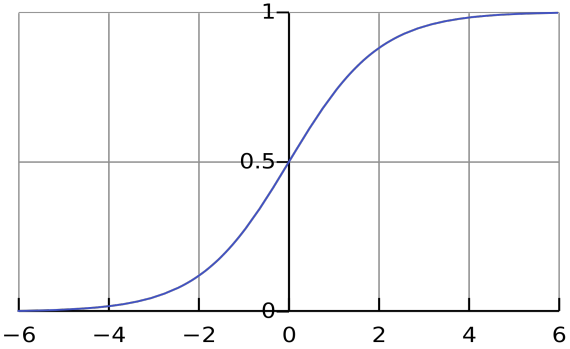


BTW, how many parameters?

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

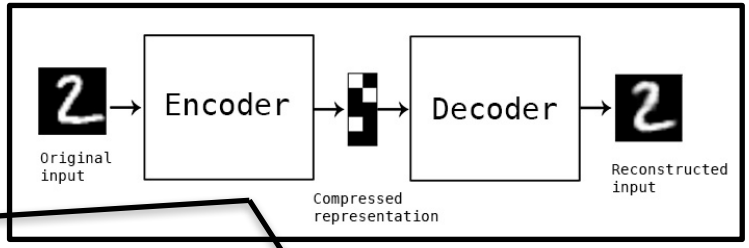
$$h^{(i)} = f_{\theta_e}(x^{(i)}) = \sigma(\theta_e^{\mathrm{T}} x^{(i)} + b_e^{(i)})$$
$$r^{(i)} = g_{\theta_r}(h^{(i)}) = \sigma(\theta_r^{\mathrm{T}} h^{(i)} + b_r^{(i)})$$

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

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)))
```



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

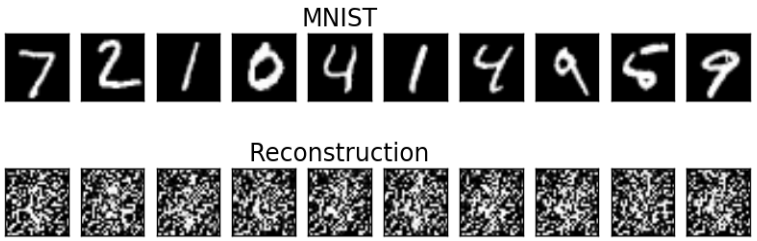
$$r^{(i)} = g_{\theta_r}(h^{(i)}) = \sigma(\theta_r^{\mathrm{T}} h^{(i)} + b_r^{(i)})$$

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

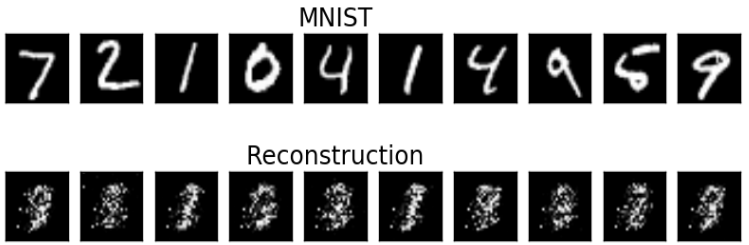
https://github.com/davidmeyer/ml/blob/master/tensorflow/autoencoder.{py,ipynb}

Autoencoder Output

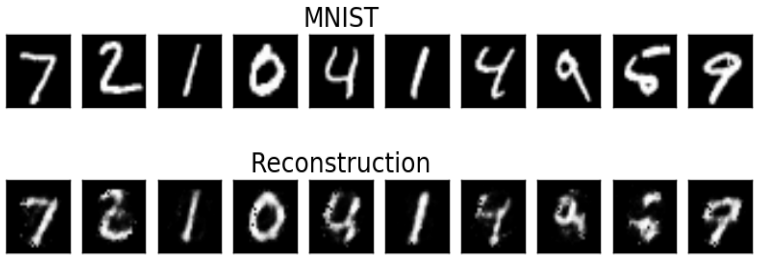
1 example



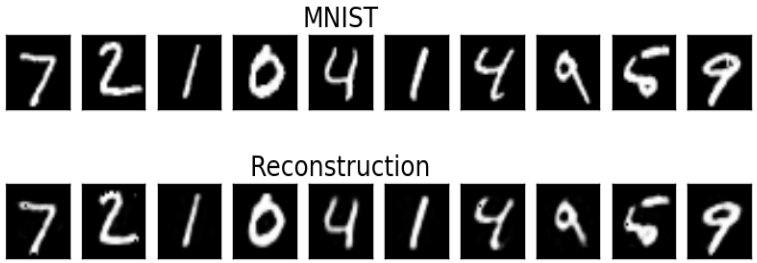
10 examples



100 examples



1000 examples

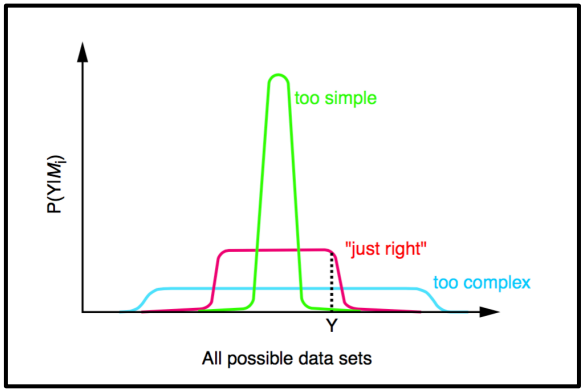


After training, the AE gets low *reconstruction error* on digits from MNIST and high reconstruction error on everything else: It has learned to recognize MNIST

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

BTW, How Much Of This Has Been Applied To Networking?

- Not too much. Many reasons:
- Still early days
- Diverse types of network data
  - Flows, logs, various KPIs, … with no obvious way to combine
  - Incomplete data sets, non-iid data data
  - Network data not designed for ML
- Different models for different data types
  - Still active area of investigation
  - Occam's Razor
- Is there a useful "Theory of Network"?
  - Consider the problem of object recognition/conv nets
  - Transfer learning



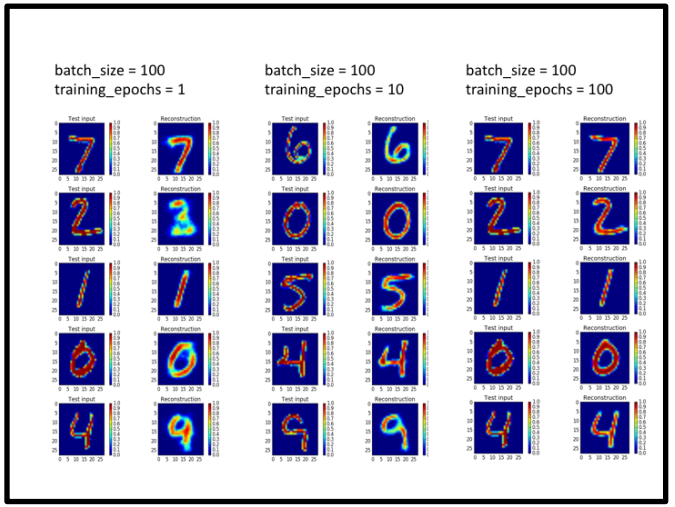
- *Community* challenges: Skill sets, proprietary data sets and use-cases, …
  - Concern about the probabilistic nature of ML

Next Steps

- Build a prototype
  - Dino
    - Provide raw data from the LISP mapping system
  - dmm
    - process data into appropriate form
  - Use ML to detect anomalies
    - Classify anomaly types
  - Remediation
    - Frequency hopping idea
  - Iterate
- Get feedback from group
- Iterate (again)

Agenda

- What is the Use Case?
- What Can We Do Today, And What Are The Challenges?
- Discussion
- Technical explanations/code
  - https://github.com/davidmeyer/ml
  - http://www.1-4-5.net/~dmm/ml



Q&A

Thank you

(have more questions/comments? dmm@1-4-5.net)