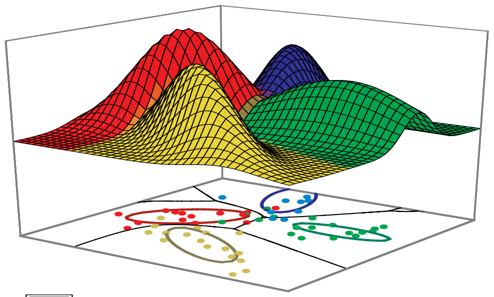
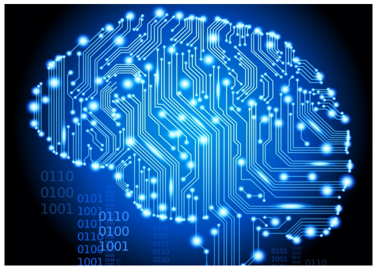
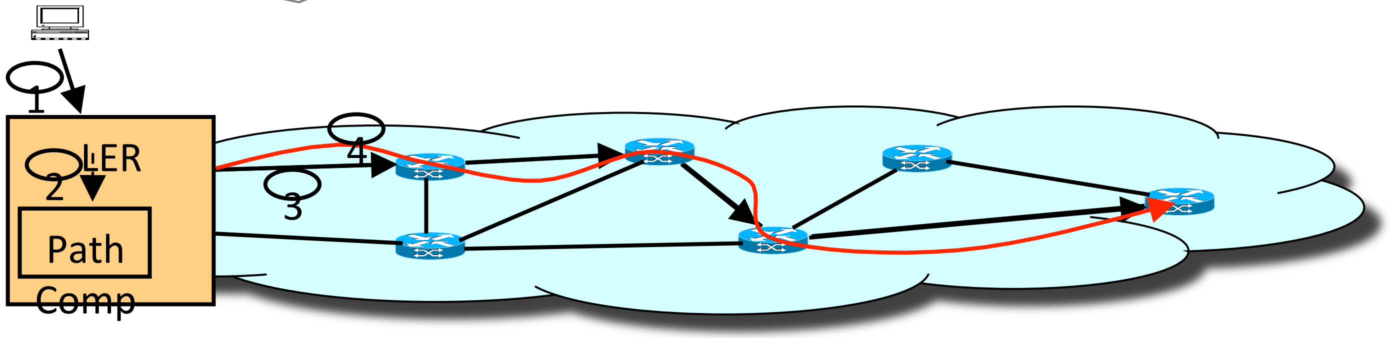
A *Very* Brief Introduction to Machine Learning and its Application to PCE



+



=



David Meyer

Next Steps for the Path Computation Element Workshop

Feb 17-18, 2015

http://ict-one.eu/pace/public_wiki/mediawiki-1.19.7/index.php?title=Workshops

dmm@{brocade.com,uoregon.edu,1-4-5.net,…}

Agenda

- Goals for this Talk
- What is Machine Learning?
  - Kinds of Machine Learning
- ~~Machine Learning Fundamentals~~
  - ~~Shallow dive~~
  - ~~Regression and Classification – Inductive Learning~~
  - ~~Focus on Artificial Neural Networks (ANNs)~~
- ~~A Bit on Unsupervised Learning~~
- ~~Deep Learning~~
- ~~Google *Power Usage Effectiveness* (PUE) Optimization Application~~
- PCE?

With figures courtesy Yoshua Bengio and others

Goals for this Talks

To give us a basic common understanding of machine learning so that we can discuss the application of machine learning to PCE

Before We Start

What is the SOTA in Machine Learnin

- "Building High-level Features Using Large S
  Andrew Ng, et. al, 2012
  - http://arxiv.org/pdf/1112.6209.pdf
  - Training a *deep neural netwo*
  - Showed that it is possib
    entirely *unlabeled*
  - In particular
    human
    (In
    es,
    s out-of-plane

t this is later in this deck)

achines, 16K cores

tegorizing 22K object classes

provement over current results

andom guess achieves less than 0.005% accuracy for this dataset

Andrew Ng and his crew at Baidu have recently beat this record with their Deep Speech work. See **http://arxiv.org/abs/1412.5567**

1 http://www.image-net.org/

Agenda

- ~~Goals for this Talk~~
- What is Machine Learning?
- ~~Machine Learning Fundamentals~~
  - ~~Shallow dive~~
  - ~~Regression and Classification – Inductive Learning~~
  - ~~Focus on Artificial Neural Networks (ANNs)~~
- PCE?

What is Machine Learning?

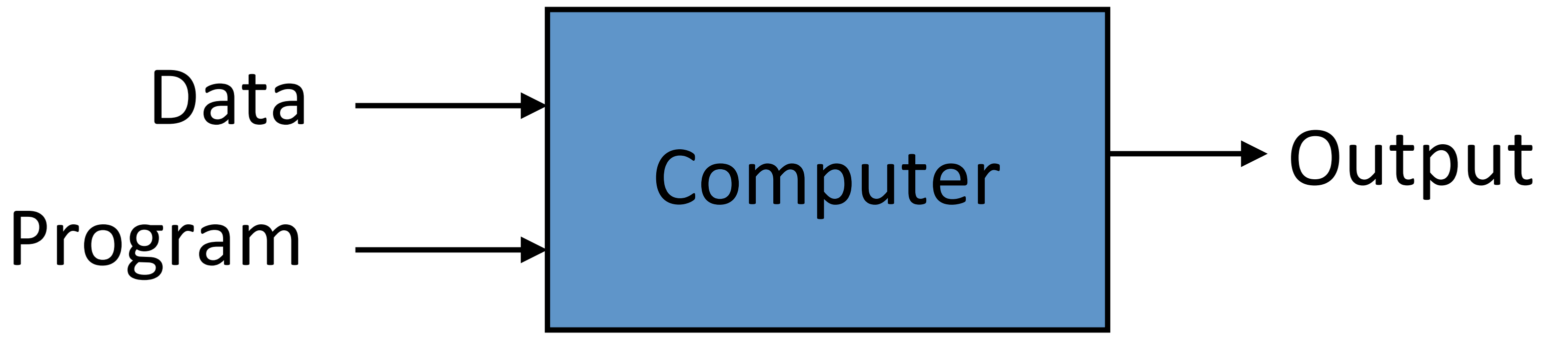
The complexity in traditional computer programming is in the code (programs that people write). In machine learning, algorithms (programs) are in principle simple and the complexity (structure) is in the data. Is there a way that we can automatically learn that structure? That is what is at the heart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

-- Andrew Ng

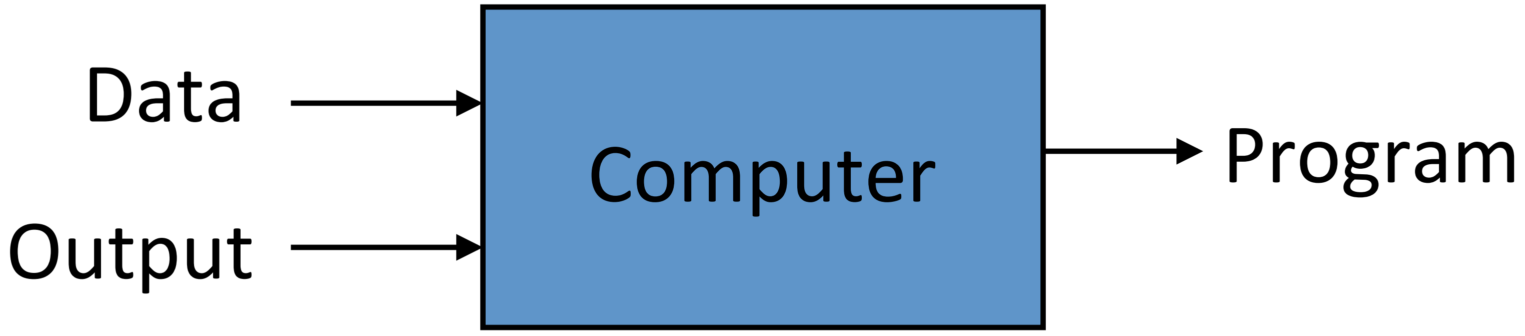
That is, machine learning is the about the construction and study of systems that can learn from data. This is very different than traditional computer programming.

The Same Thing Said in Cartoon Form

Traditional Programming



Machine Learning

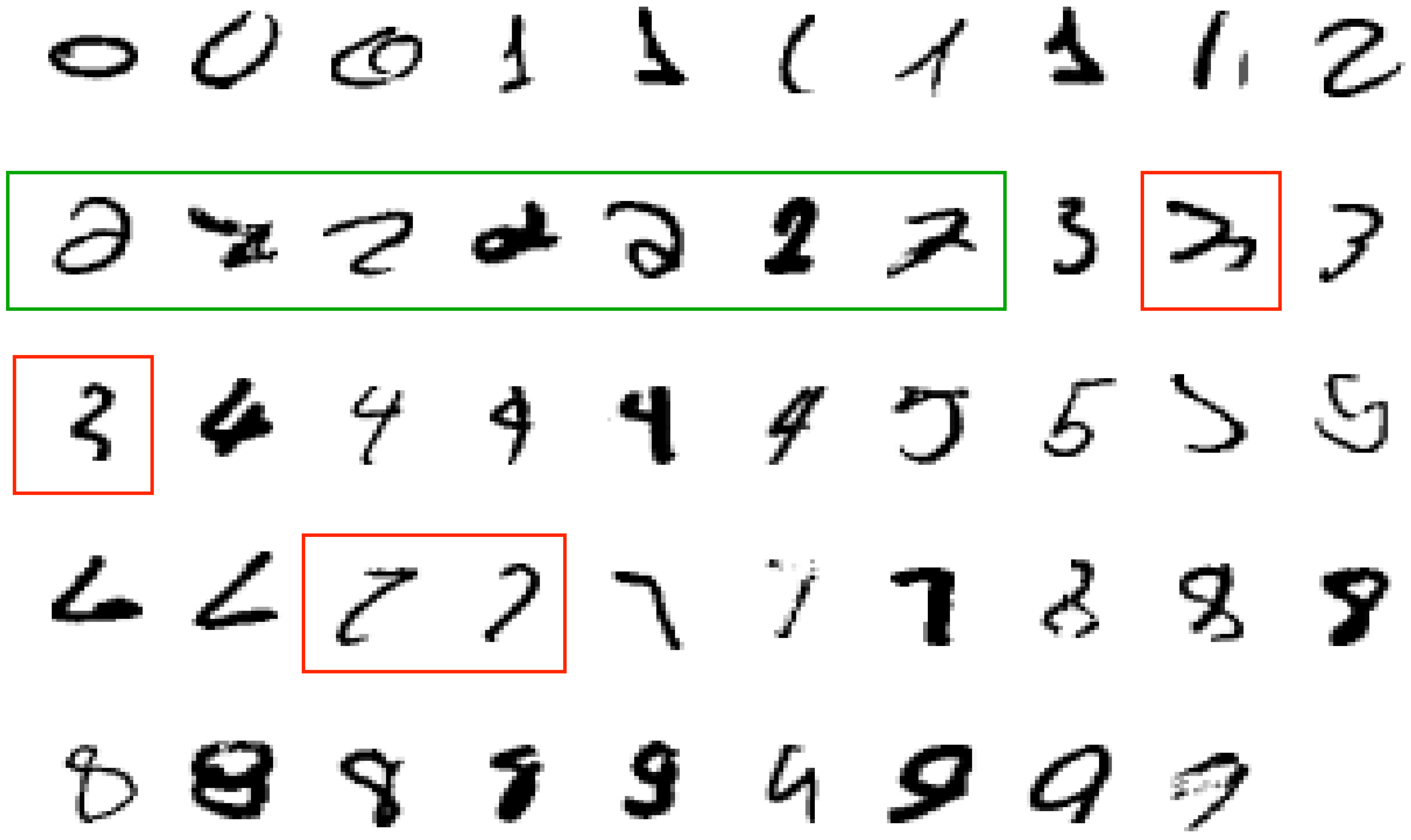


When Would We Use Machine Learning?

- 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
    - See e.g., libsvm (Support Vector Machines): http://www.csie.ntu.edu.tw/~cjlin/libsvm/
    - Most of my code in python: http://scikit-learn.org/stable/ (many others)
    - Languages (e.g., octave: https://www.gnu.org/software/octave/)

Why Machine Learning Is Hard

What is a “2”?



Examples of Machine Learning Problems

- Pattern Recognition
  - Facial identities or facial expressions
  - Handwritten or spoken words (e.g., Siri)
  - Medical images
  - Sensor Data/IoT
- Optimization
  - Many parameters have "hidden" relationships that can be the basis of optimization
  - Obvious PCE use case
- 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

Agenda

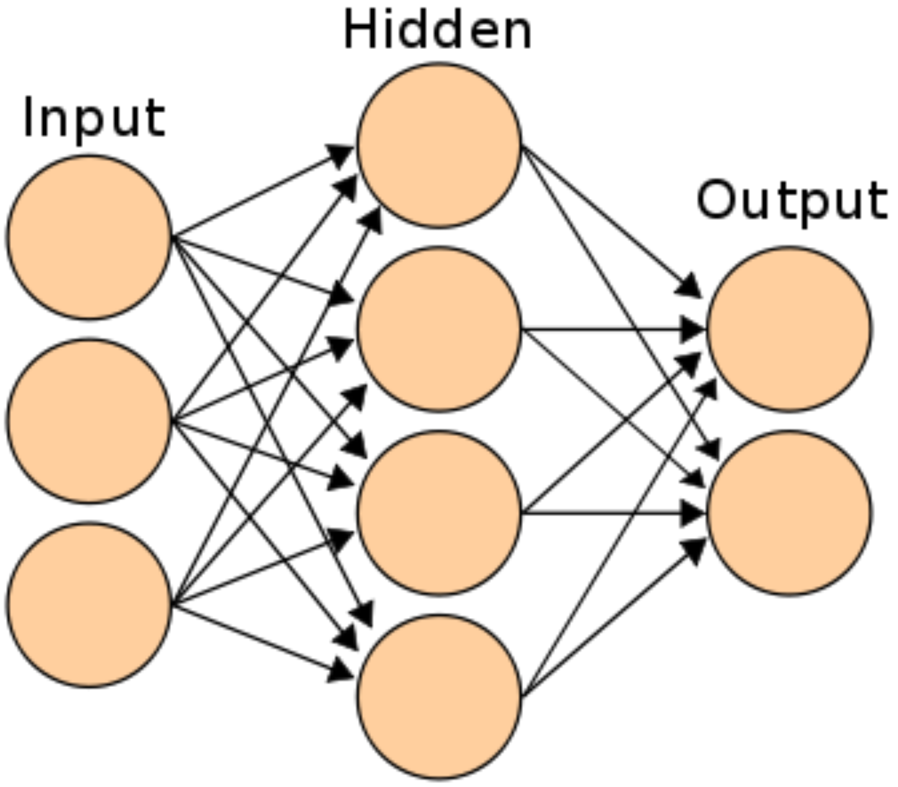
- ~~Goals for this Talk~~
- ~~What is Machine Learning?~~
- ~~Machine Learning Fundamentals~~
  - ~~Shallow(er) dive~~
  - ~~Inductive Learning: Regression and Classification~~
  - ~~Focus on Artificial Neural Networks (ANNs)~~
- PCE?

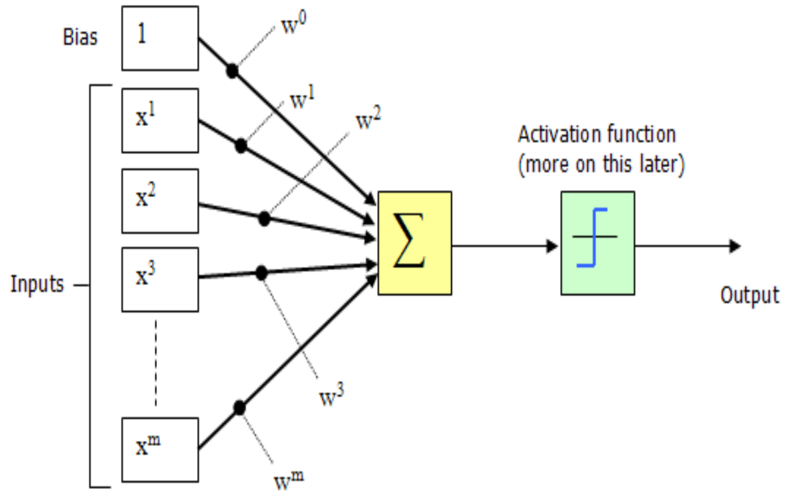
So What Is Inductive Learning?

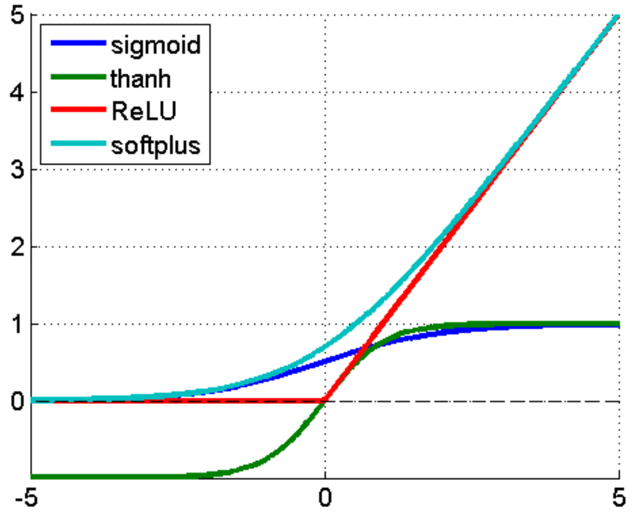
- Given examples of a function *(x, f(x))*
  - Supervised learning (because we're given *f(x)*)
  - Don't explicitly know *f* (rather, trying to fit a model to the data)
  - Labeled data set (i.e., the *f(x)*'s)
  - Training set may be noisy, e.g.*, (x, (f(x) + ε))*
  - Notation: $(x_i, f(x_i))$ denoted $(x^{(i)}, y^{(i)})$
  - $y^{(i)}$ 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

Neural Nets in 1 Slide (☺)

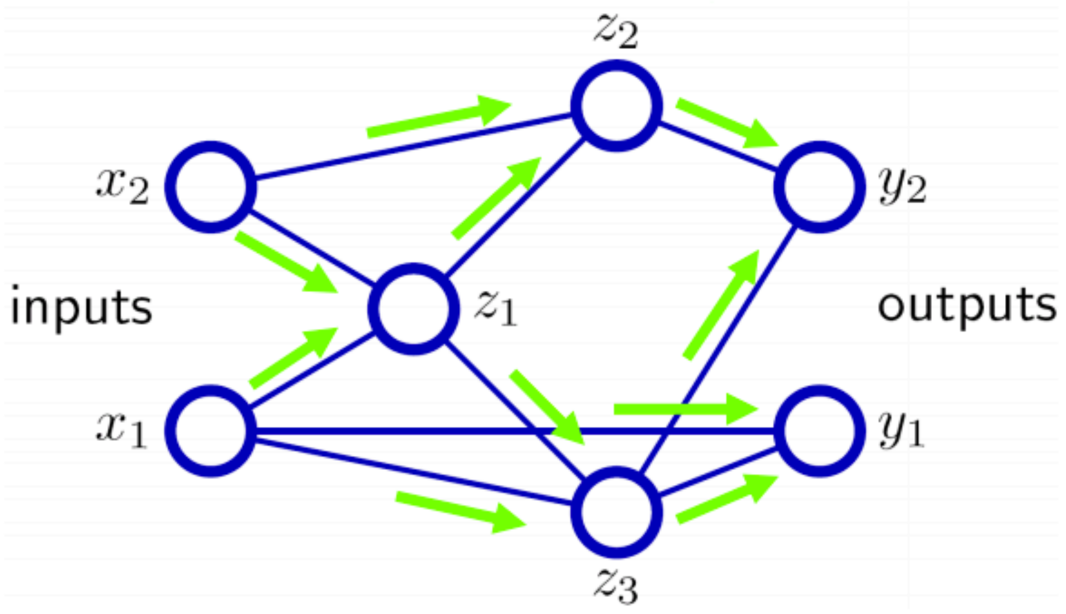






Forward Propagation Cartoon

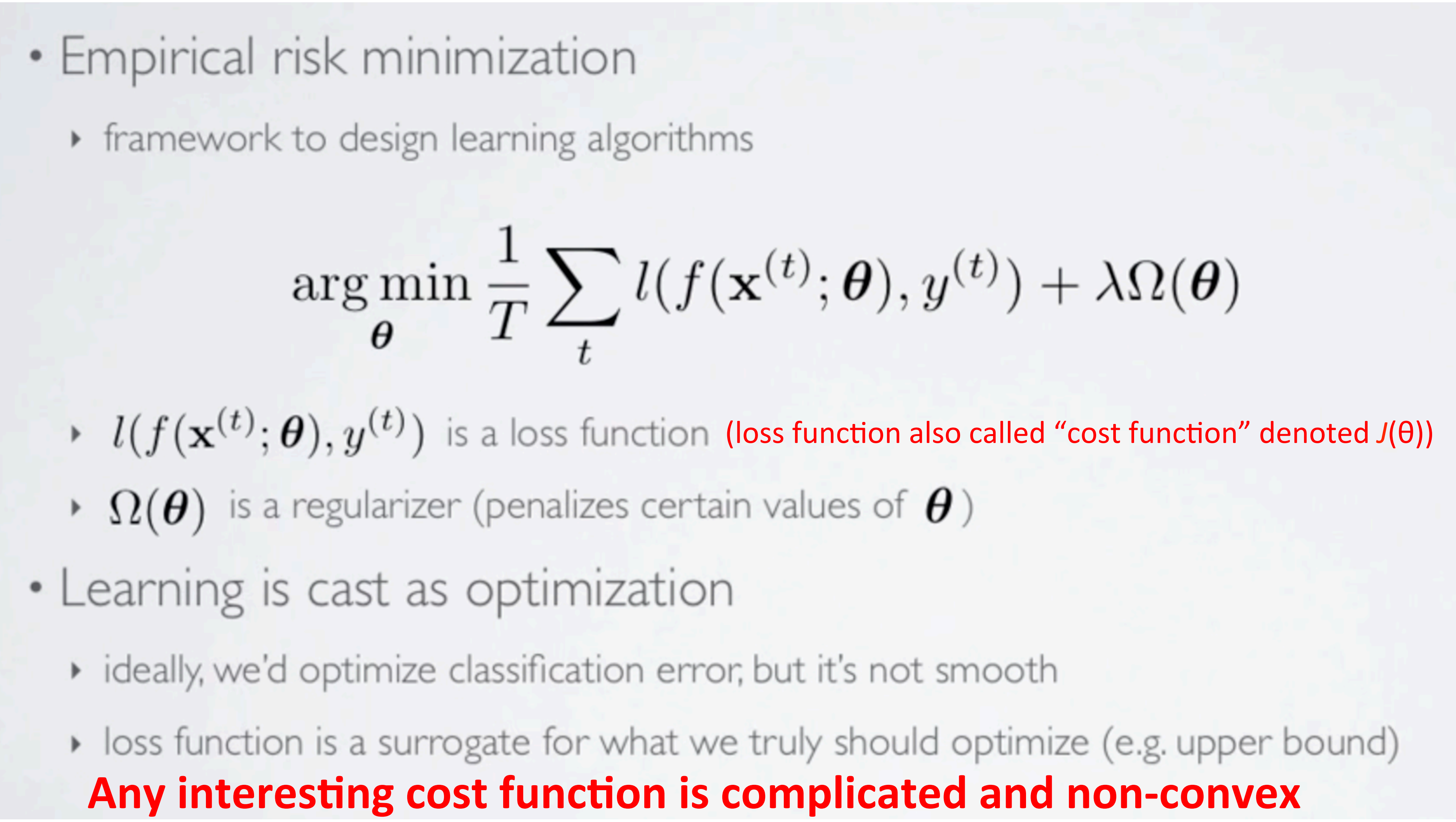
- Forward Propagation :
  - Sum inputs, produce activation, feed-forward



Backpropagation Cartoon

$\overline{i=1}$

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 $J(\theta_0, \theta_1)$

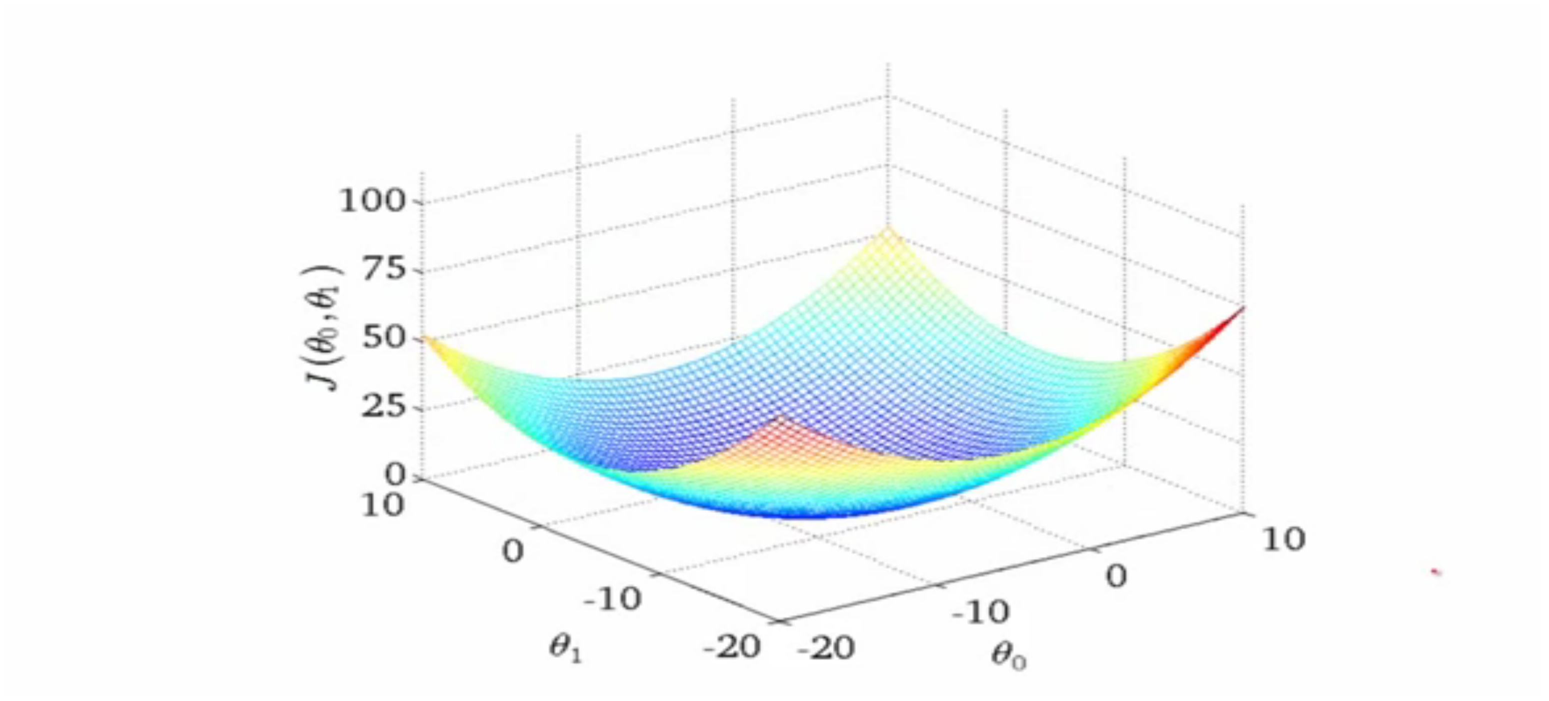
Want $\min\limits_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

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

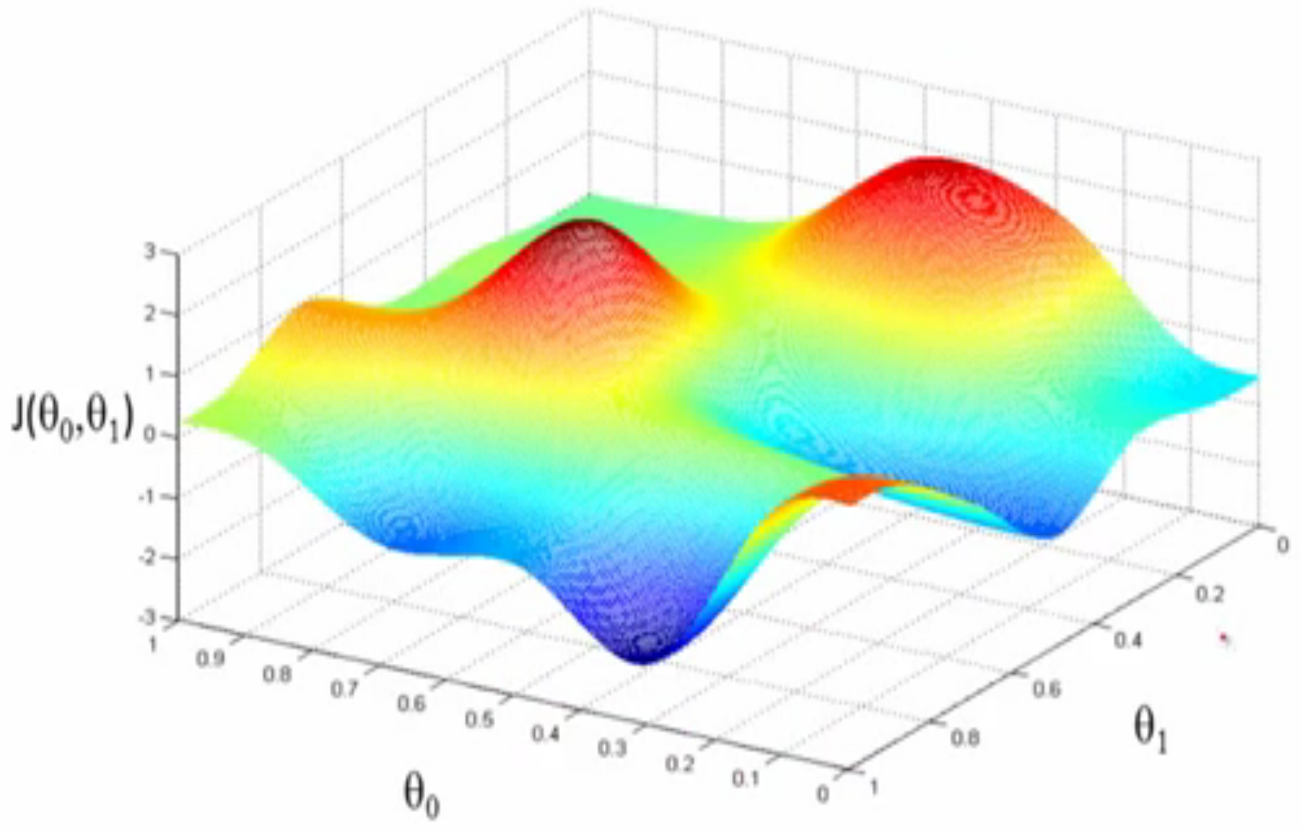
  until we hopefully end up at a minimum

Gradient Descent Intuition 1
Convex Cost Function



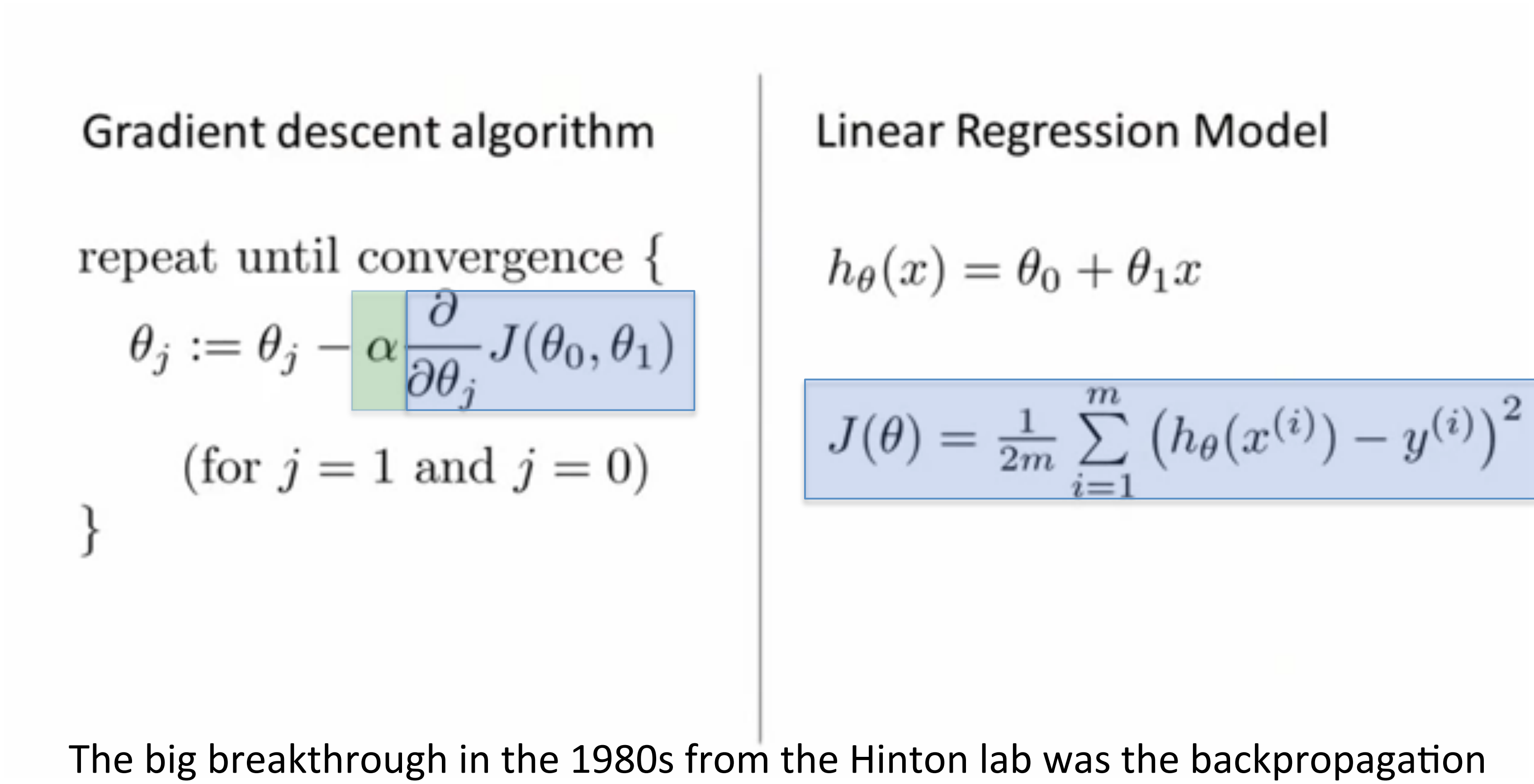
One of the many nice properties of convexity is that any local minimum is also a global minimum

Gradient Decent Intuition 2



Unfortunately, any interesting cost function is likely non-convex

Solving the Optimization Problem *Gradient Descent* for Linear Regression



Gradient descent algorithm

```
repeat until convergence {
    $\theta_j := \theta_j - \boxed{\alpha}\boxed{\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1)}$
        (for $j = 1$ and $j = 0$)
}
```

Linear Regression Model

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

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

The big breakthrough in the 1980s from the Hinton lab was the backpropagation algorithm, which is a way of computing the gradient of the loss function with respect to the model parameters θ

Agenda

- ~~Goals for this Talk~~

- ~~What is Machine Learning?~~
  - ~~Kinds of Machine Learning~~

- ~~Machine Learning Fundamentals~~
  - ~~Shallow dive~~
  - ~~Inductive Learning: Regression and Classification~~
  - ~~Focus on Artificial Neural Networks (ANNs)~~

- PCE?

Now, How About PCE?

- PCE ideally suited to SDN and Machine Learning
- Can we infer properties of paths we can't directly see?
  - Likely living in high-dimensional space(s)
  - i.e., those in other domains
- Other inference tasks?
  - Aggregate bandwidth consumption
  - Most loaded links/congestion
  - Cumulative cost of path set
  - Uncover unseen correlations that allow for new optimizations
- How to get there from here
  - The PCE was always a form of "SDN"
  - Applying Machine Learning to the PCE requires understanding the problem you want to solve and what data sets you have

PCE Data Sets

- Assume we have labeled data set
  - {($X^{(1)}$,$Y^{(1)}$),…,($X^{(n)}$,$Y^{(n)}$)}
    - Where $X^{(i)}$ is an m-dimensional vector, and
    - $Y^{(i)}$ is usually a k dimensional vector, $k < m$

$$\mathbf{X} \in \mathbb{R}^m$$
$$\mathbf{Y} \in \mathbb{Z}^k$$

- Strawman X (the PCE knows this information)

- $X^{(i)}$ = (Path end points,
  Desired path constraints,
  Computed path,
  Aggregate path constraints (e.g. path cost),
  Minimum cost path,
  Minimum load path,
  Maximum residual bandwidth path,
  Aggregate bandwidth consumption,
  Load of the most loaded link,
  Cumulative cost of a set of paths,
  Other (possibly exogenous) data)

The $Y^{(i)}$'s are a set of classes we want to predict, e.g., congestion, latency, …

What Might the Labels Look Like?

$$\mathbf{Y} = \begin{bmatrix} Congestion \\ Latency \\ Class2 \\ Class3 \\ Class4 \\ \ldots \end{bmatrix} \quad \xrightarrow[\text{(instance)}]{} \quad Y^{(i)} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ \ldots \end{bmatrix}$$

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 predict?
- Get the (labeled) data set (this is always the “trick”)
  - Split into training, test, cross-validation
    - Avoid generalization error (bias, variance)
  - Avoid data leakage
- Choose a model
  - I would try supervised DNN
    - We want to find “non-obvious” features, which likely live in high-dimensional space
- Test on (previously) unseen examples
- Write code
- Iterate

Issues/Challenges

- Is there a unique model that PCEs would use?
  - Unlikely → online learning
- PCE is a *non-perceptual* tasks (we think)
  - Most if not all of the recent successes with ML have been on perceptual tasks (image recognition, speech recog/generation, …)
  - Does the Manifold Hypothesis hold for non-perceptual data sets?
- 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)
- 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?

Q & A

Thanks!

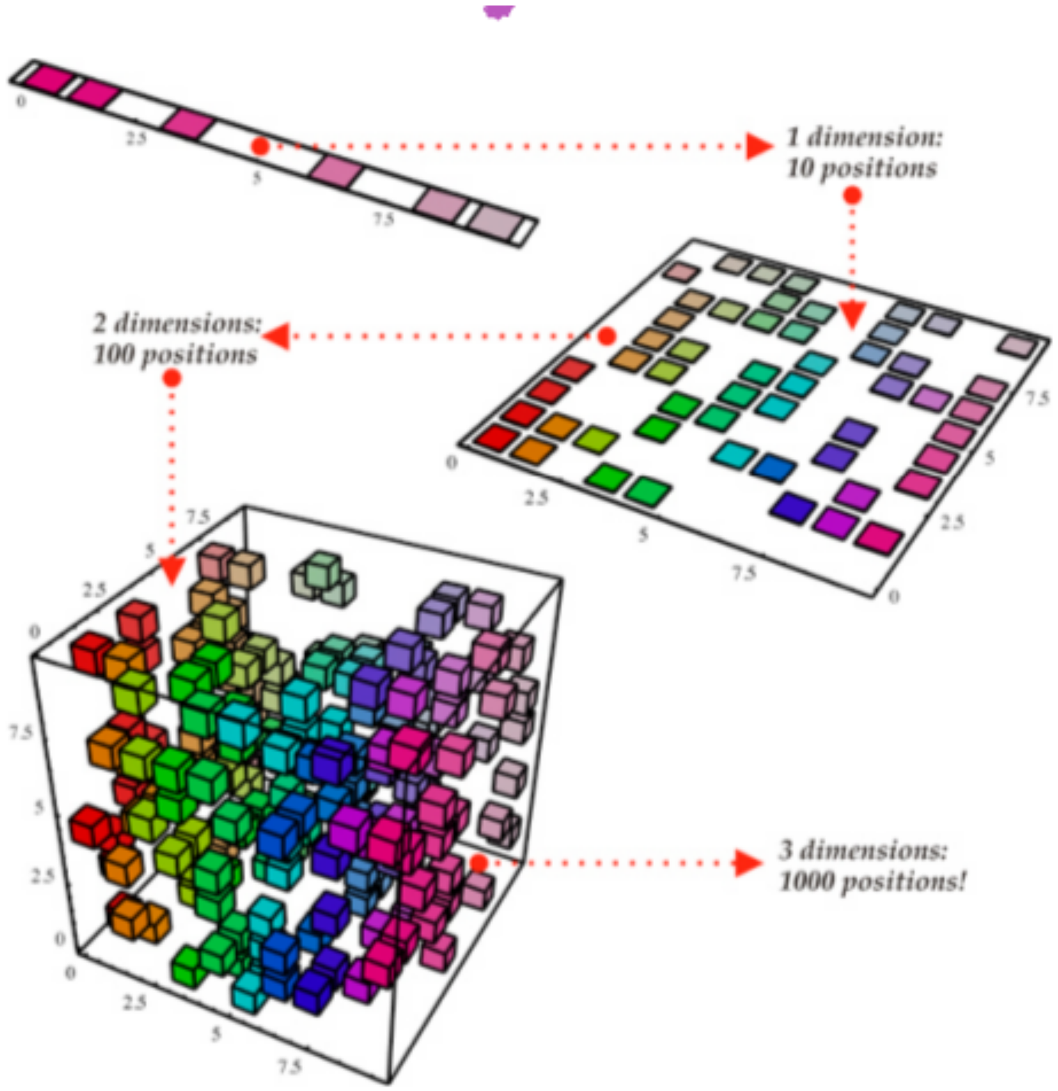
BTW, How Can Machine Learning Possibly Work?

- You 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
  - 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?

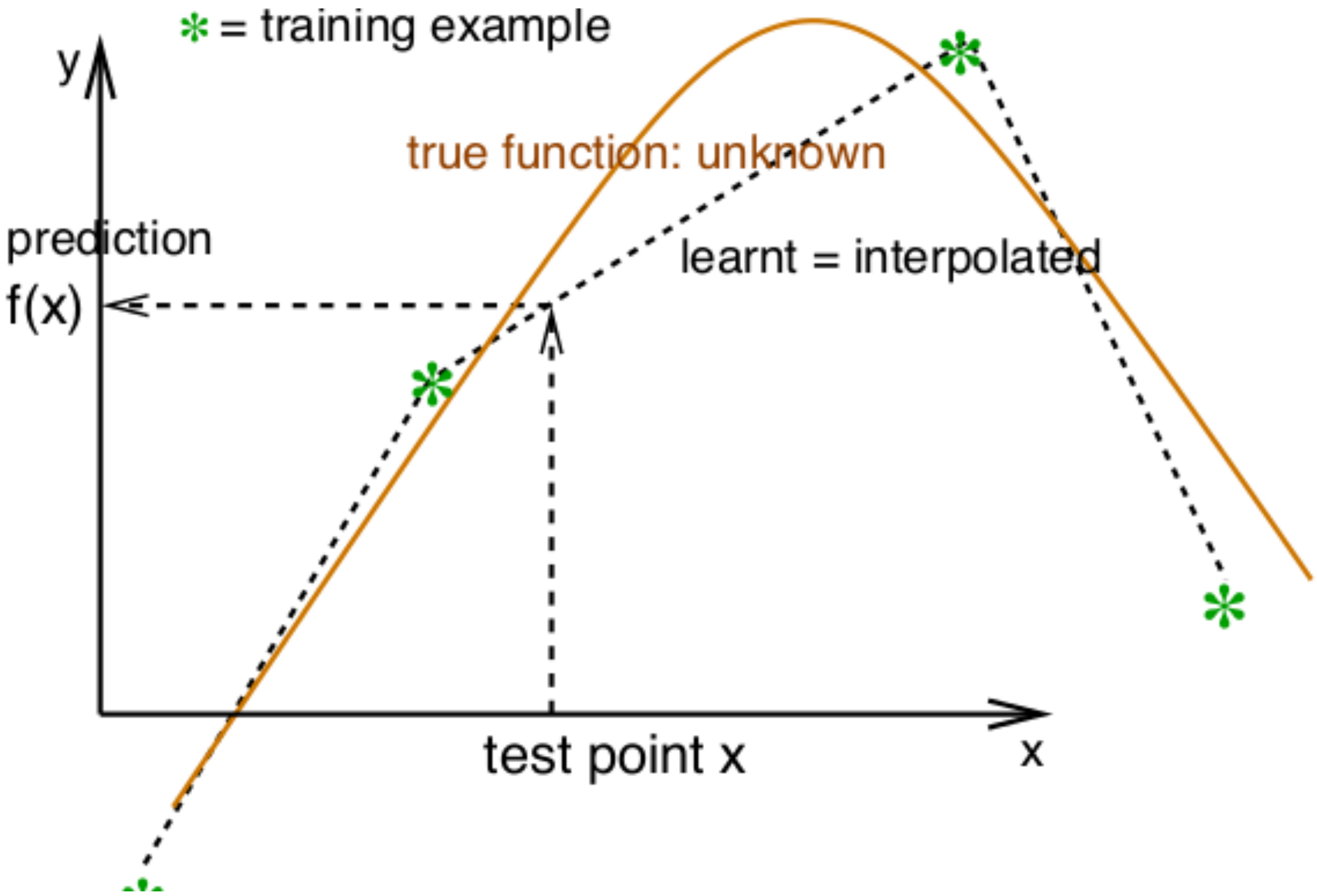
What We Are Fighting: The Curse Of Dimensionality

To generalize locally, need representative examples for all relevant variations!

Classical solution: hope for a smooth enough target function, or make it smooth by handcrafting good features / kernel



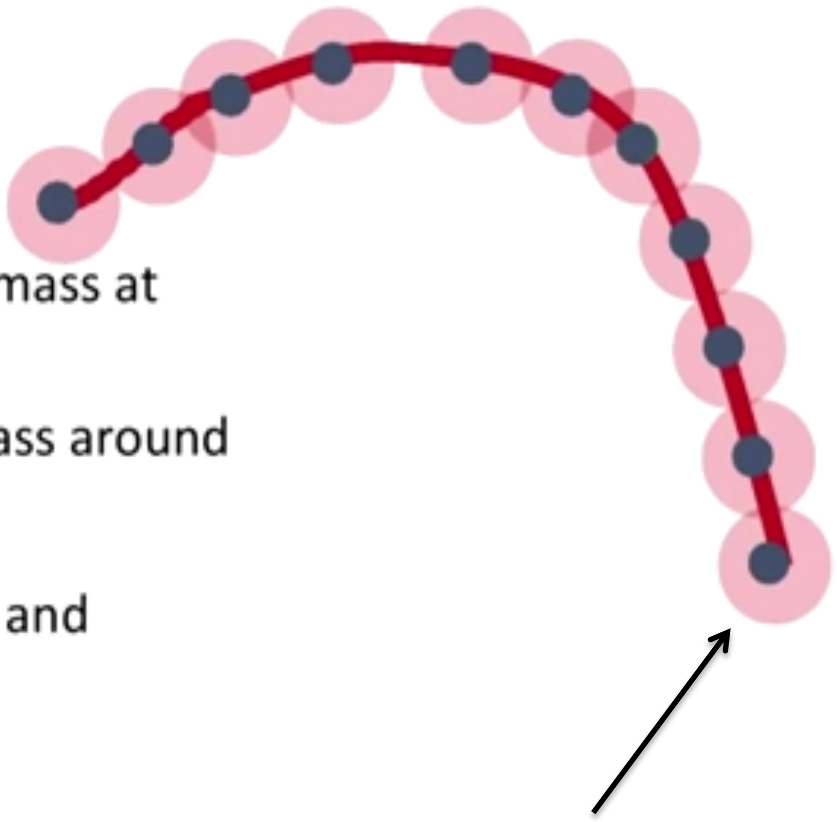
Smoothness



Smoothness assumption: If **x** is geometrically close to **x’** then **f(x) ≈ f(x’)**

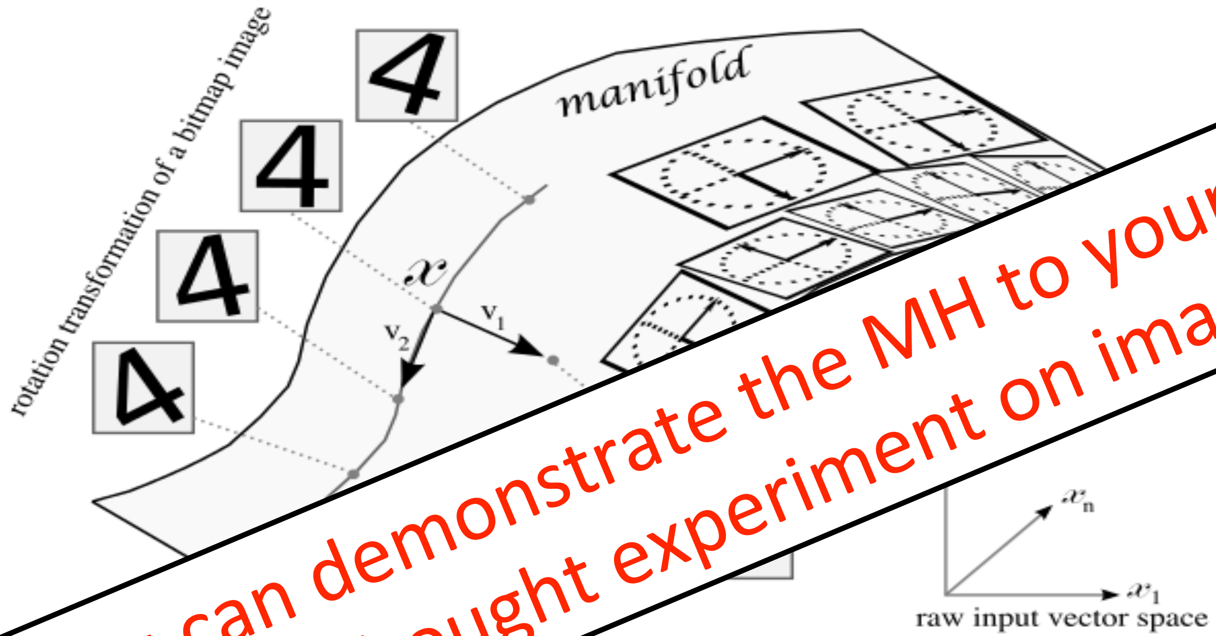
Smoothness, basically…

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



Probability mass **P(Y=c|X;θ**)

Manifold Hypothesis



BTW, you can demonstrate the MH to yourself with a simple thought experiment on image data…

pothesis states that ***natural data*** forms lower dimensional manifolds ding space. Why should this be? Well, it seems that there are both th oretical 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.