Machine Learning from Small to Big Data

Luis L. Pérez, Ph.D.

luis.perez@singularities.com

lperez@ucenfotec.ac.cr

A bit about me…

- Chief Data Scientist at Singularities
  - http://www.singularities.com
  - AI/ML solutions for retail, manufacturing, finance.
  - Always accepting internship applications!
- I also teach distributed DBs at Cenfotec
- Before all of this, went to Rice for my CS Ph.D.
  - Research work on distributed databases + applied machine learning

What this talk is about

- What happens to ML when data small → big
- Definitions
  - No. of features / columns / attributes
  - No. of data points / rows
  - Machine generated input more common
  - On-line settings more common
- Lots of issues arise…

First off, a ML primer…

- (Will focus on *supervised* learning)
- Many definitions, but…
  - Explicitly programming vs. learning from data

```
public boolean isThisACat(byte[][][] pic) {

  // returns true if 'pic' is a picture of a cat.
}
```

The ML alternative

INPUT	OUTPUT
	True
	False
	False
	True
	True
...	...

Instead of explicitly coding up **`isThisACat`**, "learn" it from a bunch of labeled examples of the input and corresponding output.

Canonical supervised ML tasks

- In general, we want to *learn* the function

$$f(\boldsymbol{x}) = y$$

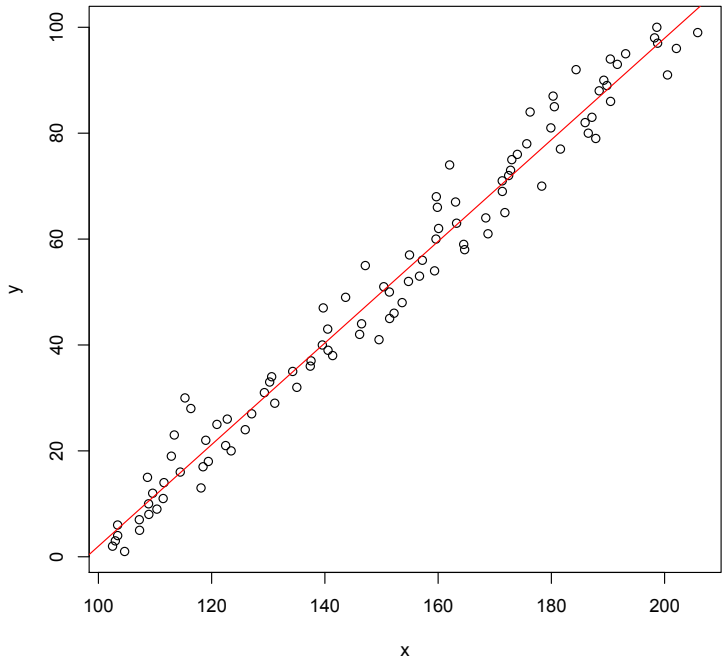
  From a bunch of examples of $(\boldsymbol{x}, y)$.
- Two forms
  - **Classification**: y is restricted to a finite set
    - a.k.a. *categorical*
    - e.g. cat/not-a-cat, marital status, which province, etc.
  - **Regression**: y is a real/natural number
    - e.g. prices, weights, days, etc.

The ML process

1. Divide data into training/test
   – e.g. 80% training, 20% test
2. Build model from training dataset
   – Model: operations + parameters
     - Operations are fixed for a given model
     - Parameter values are *variable* ← !
   – Training: algorithm finds **best** parameter values w.r.t. training data
3. Use test set to assess model performance

The ML process

- Remember this stuff from Stats 101?



Linear regression model:
$y \sim ax + b$

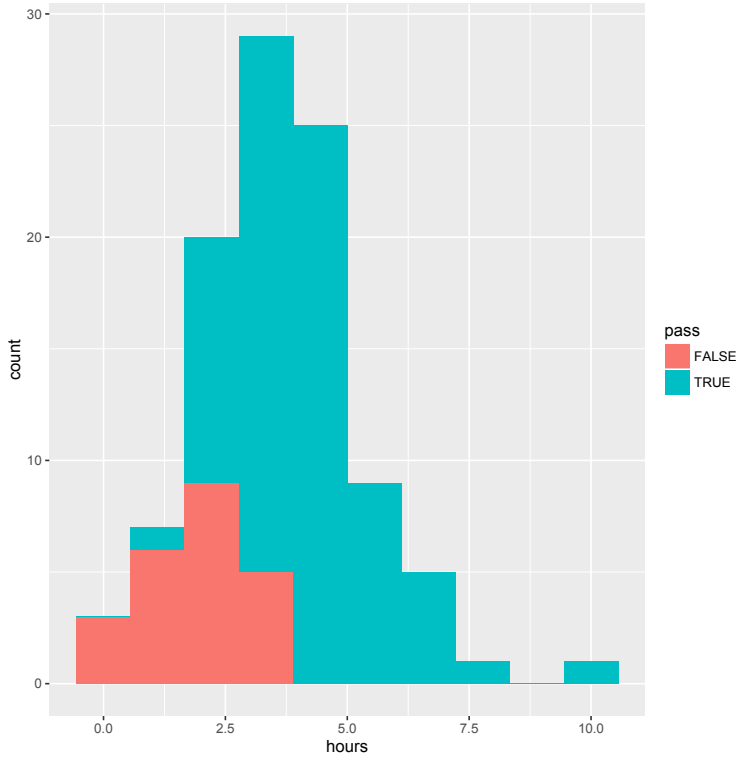
Training: find **best** values for a and b

"**Best**": minimize error measure

Training algorithm solves an optimization problem

The ML process

- Let's talk about about *linear* classification.
- Example dataset: (hours, pass)



$\text{pass} \sim g(a*\text{hours} + b)$

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

in $[0,1]$

(probability of passing)

Find a and b that minimize error.

Known as

***logistic regression* (!)**

The ML process

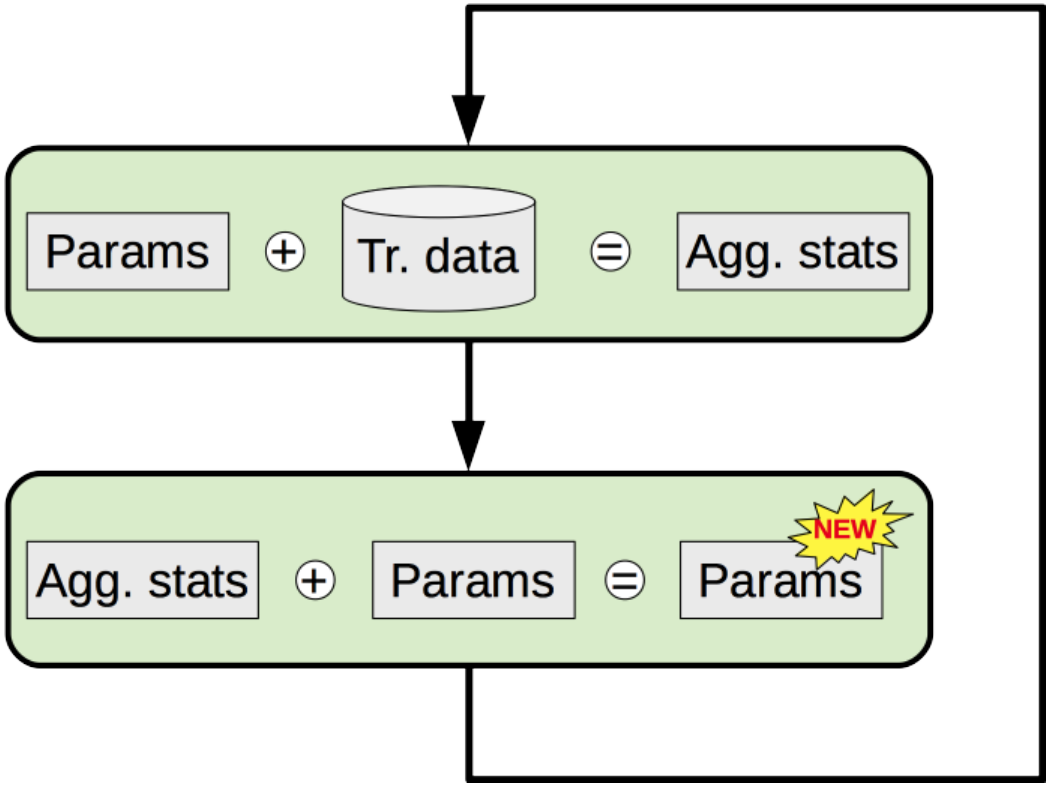
- Lots of models
  - Linear models, tree-based methods, neural networks, support vector machines, etc.
- Linear models
  - Input-output pair $(\boldsymbol{x}, y)$ is modeled as

$$y \sim g(a_1 x_1 + a_2 x_2 + \ldots + a_m x_m + b)$$

  - Given dataset $(\mathbf{X}, \mathbf{y})$ of n datapoints and m features, training finds values for a_1, a_2, …, a_m, and b that minimize error.

The ML process

- Iterative training algorithms (finding a's & b)
- Very common pattern!

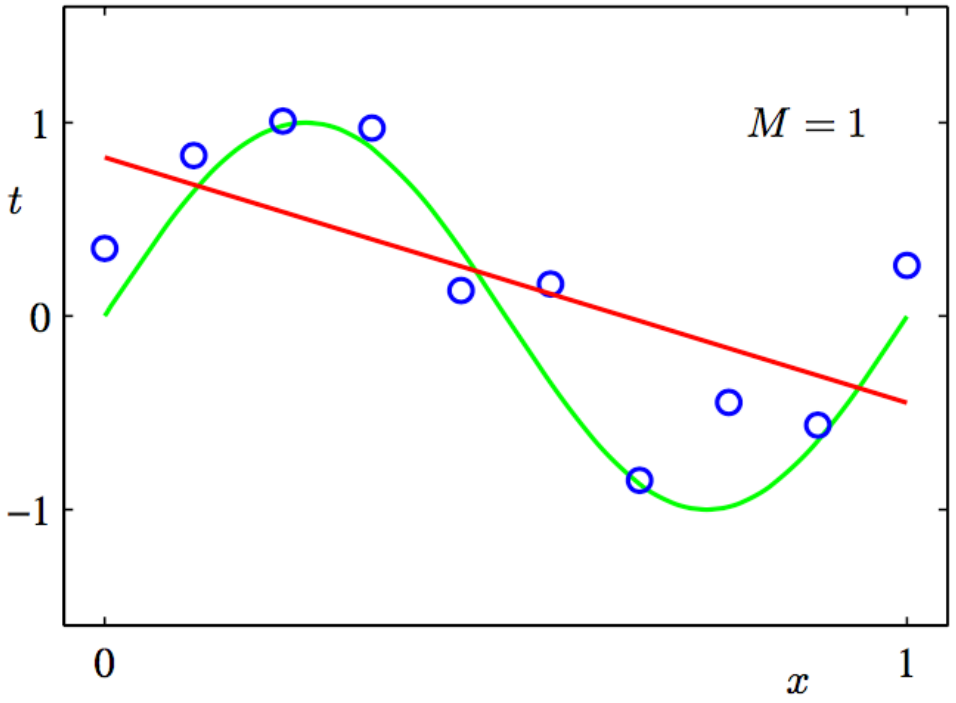


The ML process

- Final step: validation with test set
  - Measure resulting model's accuracy
  - Standard measures
    - Regression: RMSE
    - Classification: confusion matrix (and derived measures)
- Most importantly: diagnose fitting problems
  - Overfitting and underfitting
  - Good training error, but bad test error

The ML process

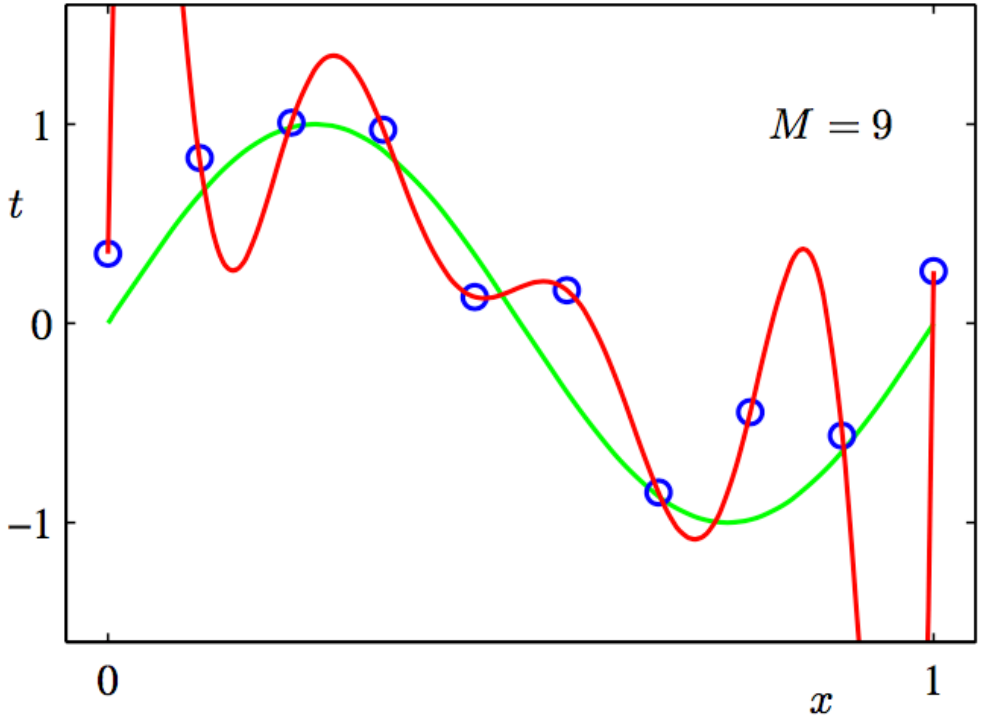
- Underfitting: I'm too simple-minded for the training data.



(Image from Bishop's PRML 1st Ed., p. 7)

The ML process

- Overfitting: I'm kinda memorizing the training data.



(Image from Bishop's PRML 1st Ed., p. 7)

The ML process

- What to do: underfitting?
  - Switch to a more sophisticated model
  - Engage in *feature engineering*
- What to do: overfitting?
  - Get more data
  - Apply a form of *regularization*

Getting more data

- More data = better, right?
  - Big data: the best setting. Right?
  - Yes, yes. But things change
- Lots of interesting issues arise
  - Systems issues
  - Stats issues
- Let's talk about the systems stuff first

Systems issues

- Data won't fit in memory or single HD
  - Or maybe it does, but training process is **slow**
- Need for parallel/distributed processing
  - Hadoop, Spark, Flink, etc.
- Need for parallelize algos that scale well
  - In terms of no. of machines and data size

Getting more data

- Let's move on to the stats issues
- What does having more features imply?
- **THE CURSE OF DIMENSIONALITY**
  - Things get weird in high-dimensional spaces
    - (Also computationally harder)
  - Affects ML, but also general data-sci and mining
- In ML, often appears as **overfitting**

Getting more data

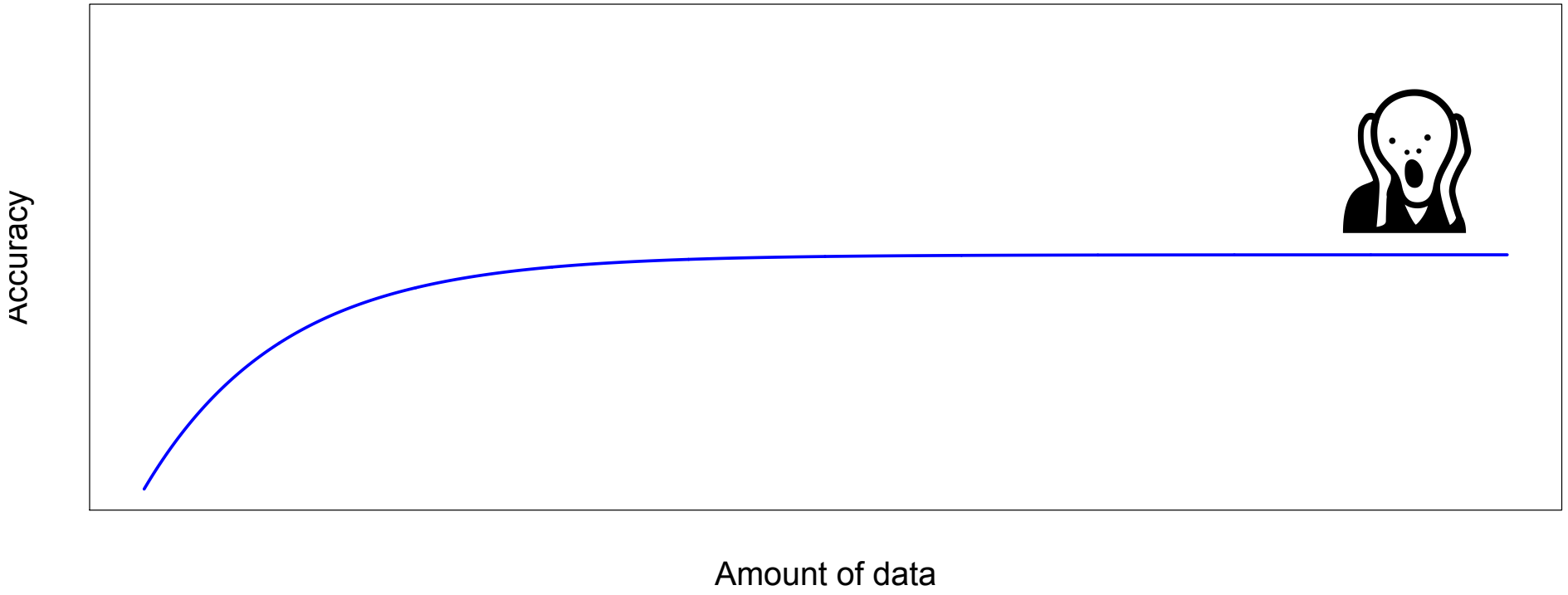
- We've already got more data!
- What can we do?
  - Many things, like regularization
  - Recall the linear model

$$y \sim g(a_1 x_1 + a_2 x_2 + \ldots + a_m x_m + b)$$

  - Core idea: get rid of as many a's as possible by driving them to zero.
  - Widely used in many models
    - LASSO, Ridge regression, Neural Networks.

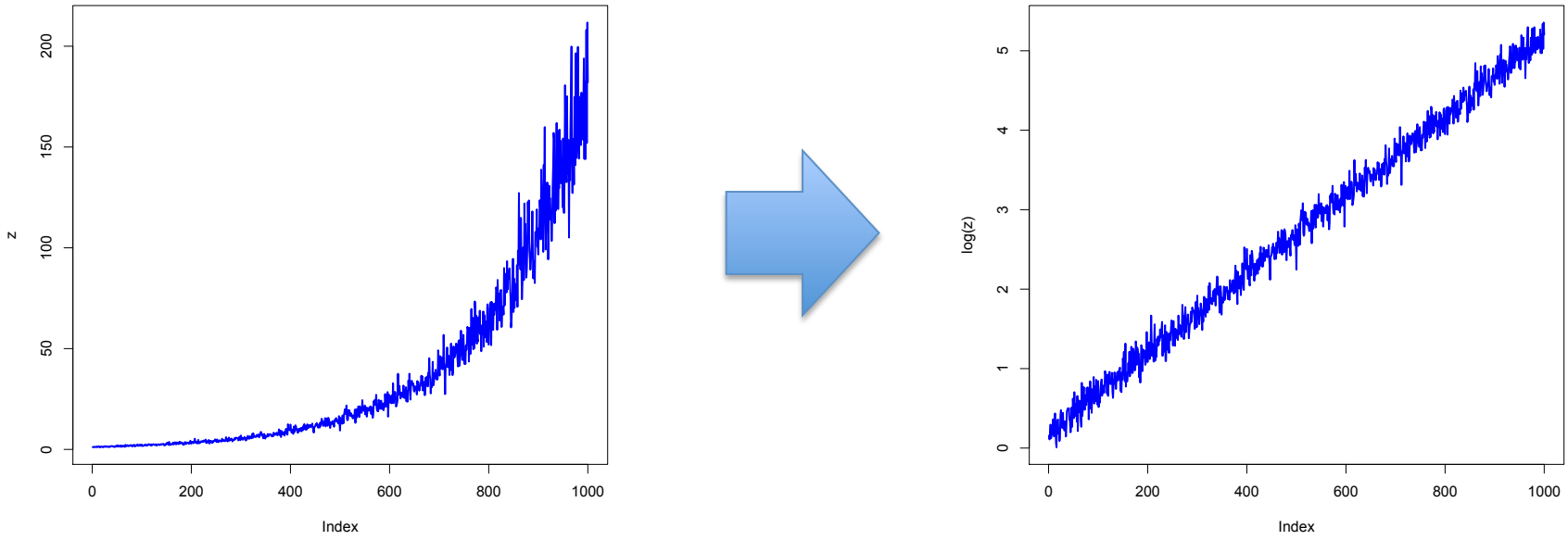
Getting more data

- We avoided overfitting (phew!), but the worst is yet to come…



Getting more data

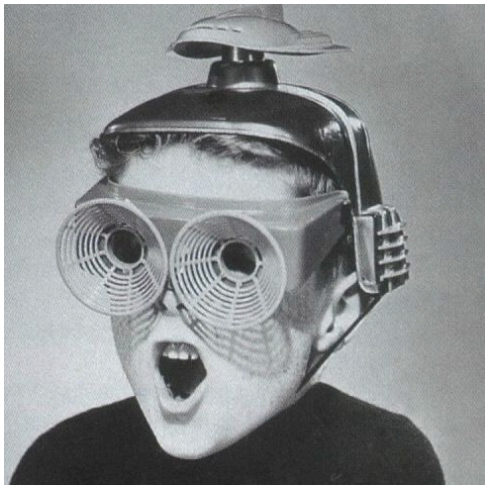
- One way out: engage in *feature engineering*
  - Adding new features or transforming existing ones
  - Super common data processing task in ML



Getting more data

- FE is very common
  - It's manual, expensive, time-consuming work
  - Hundreds, thousands of features
- Look at some better models
  - Decision trees/Random forests, Support vector machines
  - "Bump" the data-accuracy curve
  - **Neural networks / deep learning**
    - Let's talk about this

Deep Learning



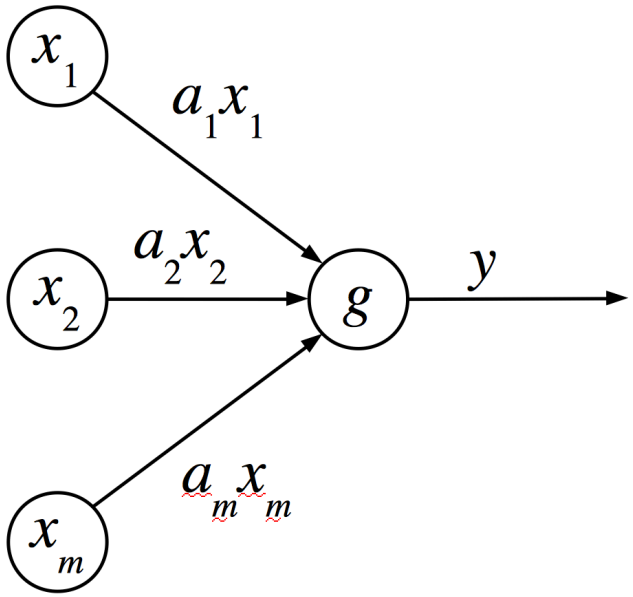
- Lots of media hype!
- What is it, really?
  - Latest research on neural networks
    - Learning algos, applications, architectures
  - Is that it? No.
    - At its core, DL is some form of **automatic feature engineering**.
      - More on this later!
    - Let's first see what a neural network is…

Deep Learning

- NNs are a powerful generalization
  - Recall:

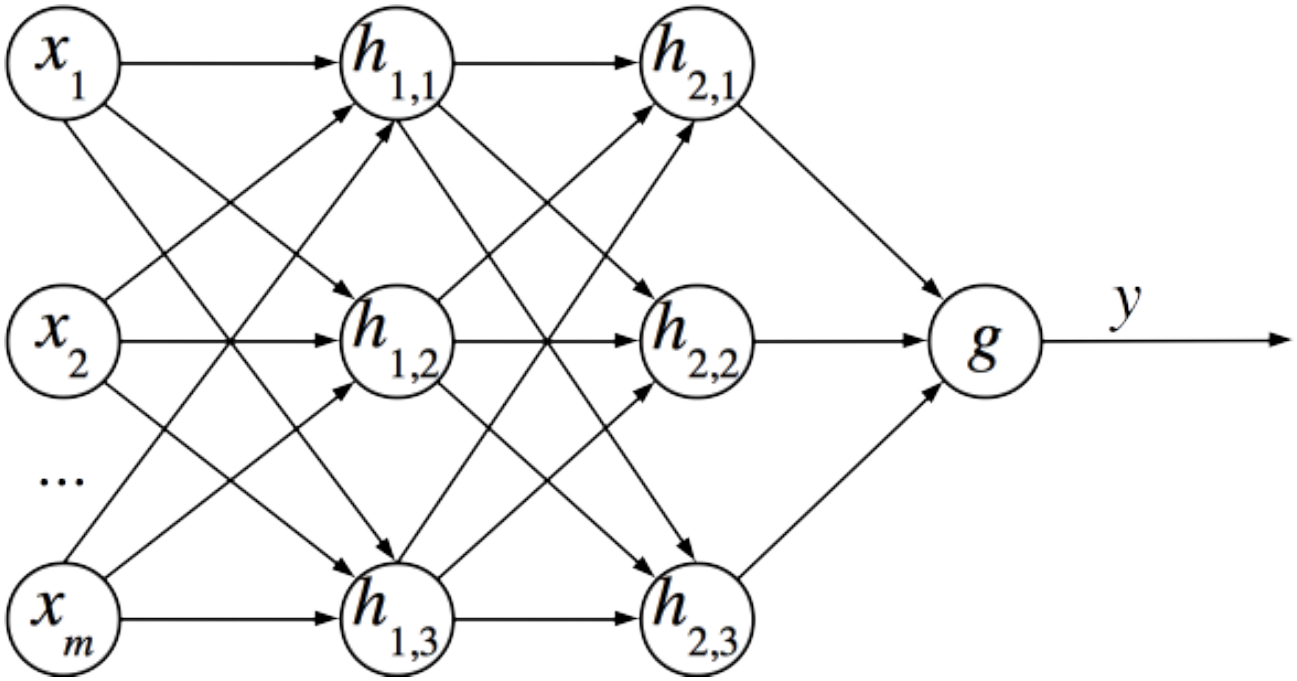
$$y \sim g(a_1 x_1 + a_2 x_2 + \ldots + a_m x_m + b)$$

Graphical representation:



Deep Learning

Stack them up:



Very general, powerful model!

Deep Learning

- But… NNs have been around for while, no?
- Training process is intensive
  - CPU (actually, GPU!) resources are available
- Numerical issues with very deep networks
- New architectures for specific applications
  - This is what's exploding as of late
  - Auto-FE is part of this

Deep Learning

- Example: Convolutional Neural Networks
- Incorporate **learned** convolutional filters into NN parameters
- Achieves auto-FE

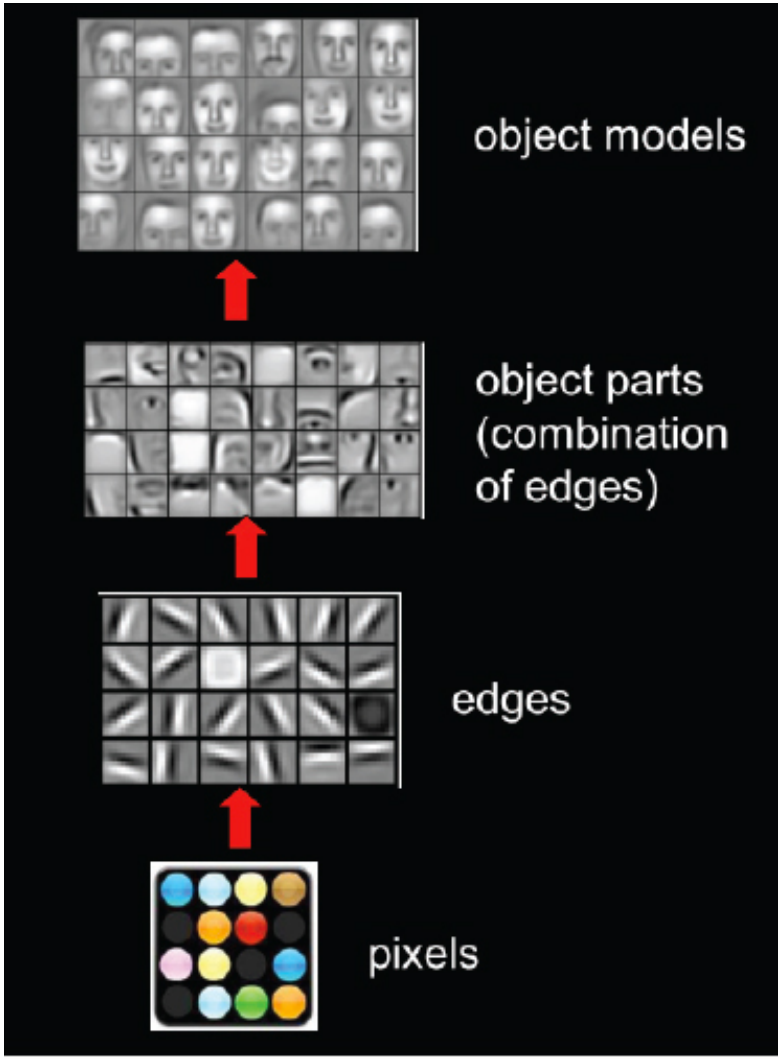
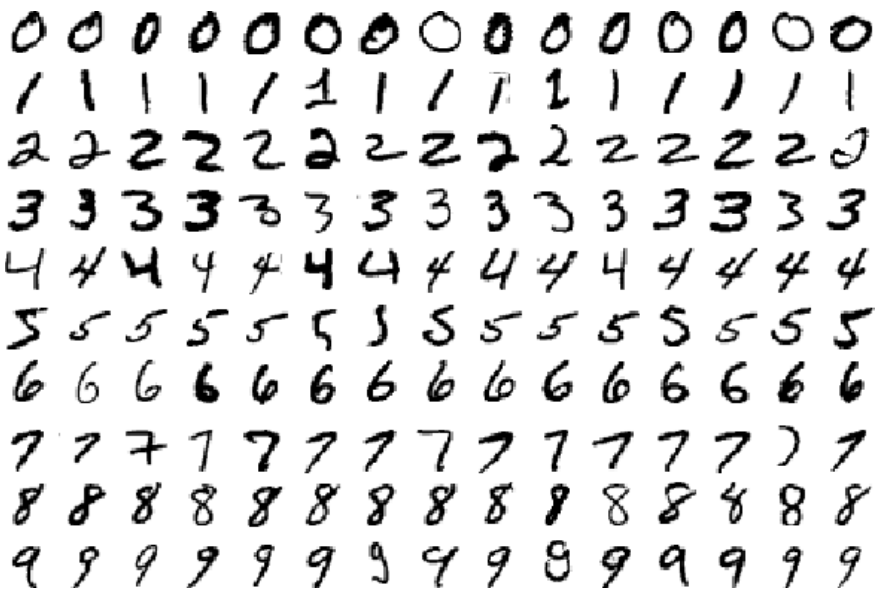


Image thanks to Honglak Lee @ Google Brain / U. Michigan

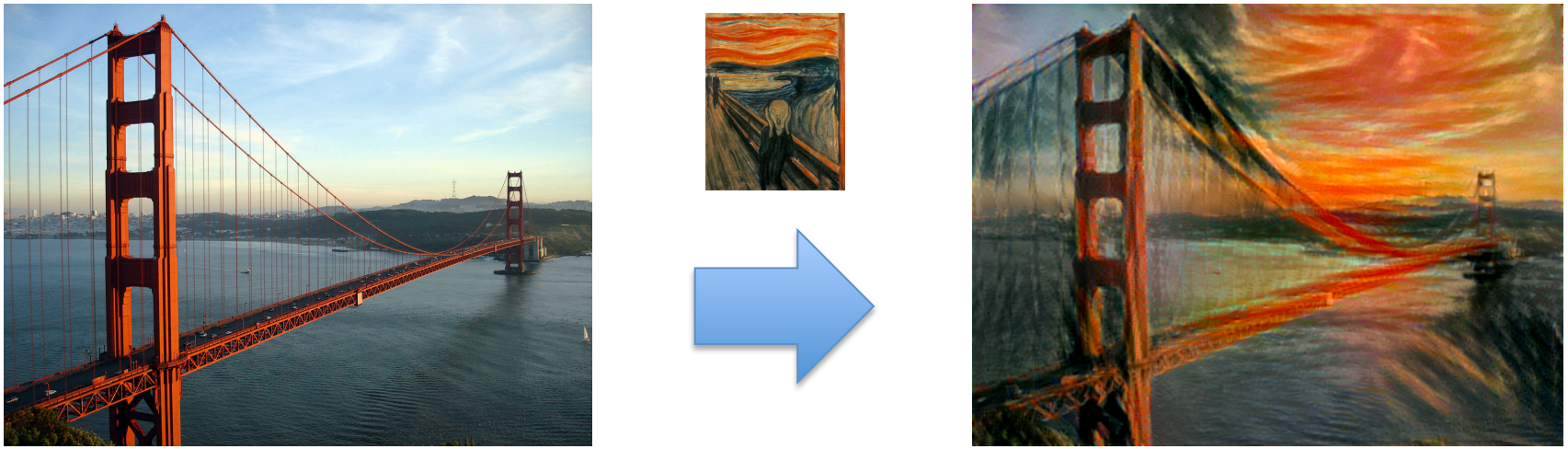
Deep Learning

- In terms of accuracy, DL architectures provide tremendous performance
- e.g. MNIST dataset
- CNNs consistently achieve accuracy >99.0%



Deep Learning

- CNNs can be adapted to a lot of image tasks
  - Face verification and recognition, object detection and localization
  - Cool stuff: neural style transfer!



Deep Learning

- Tailored architectures for many other applications
  - Recurrent NNs for text sequences and time series
  - Autoencoders for searching in high-D data
- Also, great tools
  - TensorFlow, Keras, Torch, etc.
- **Is DL going to replace "vanilla" ML?**

Boarding the ML bandwagon

- First off, some advice: **grok** it
  - Don't be one of *those* people
  - There are lots of good resources, use them!
- Many tools and libraries available for your language and Big Data processing platform of choice
- This week, you'll look at Python and Spark

Boarding the ML bandwagon

- Spark's MLlib
  - Complete ML stack: regression/classification algorithms, feature engineering tools, clustering and topic modeling
  - Main advantage: takes Spark RDDs and Datasets as input
- DeepLearning4j
  - Java (compatible Spark and Hadoop) DL library
  - GPU processing via ND4j

Boarding the ML bandwagon

- Since you'll be looking at Python…
- Keras, Caffe, Torch
  - Powerful NN modeling and inference libraries
- TensorFlow
  - Dataflow framework for ML/DL
  - Models are represented as computation graphs
  - Highly flexible with lots of features
  - Distributed and mobile implementations

Closing remarks

- ML/DL on big datasets: possible, great.
- Must keep in mind systems and stats challenges
- Find the right tool for the right problem
- Above all, have fun!

Thank you!

Questions?