Introduction to Machine Learning

Author: Victor Carbune

July 16, 2014
ROSEdu Summer Workshops

Outline

Introduction
- Workshop Overview and Schedule
- Current Trends in Research and Industry

Getting Technical
- Machine Learning: The Formal Setting
- Supervised Learning: Linear Models

Model Complexity
- Overfitting and Regularization

scikit-learn

Goals

- Get a broad overview of the field in research and industry

Goals

- Get a broad overview of the field in research and industry
- Taste the more formal and math-focused parts of ML

Goals

- Get a broad overview of the field in research and industry
- Taste the more formal and math-focused parts of ML
- Understand some common baseline algorithms

Goals

- Get a broad overview of the field in research and industry
- Taste the more formal and math-focused parts of ML
- Understand some common baseline algorithms
- Get your hands dirty with specific problems and libraries

Goals

- Get a broad overview of the field in research and industry
- Taste the more formal and math-focused parts of ML
- Understand some common baseline algorithms
- Get your hands dirty with specific problems and libraries

- Meet others, ask questions, have fun - it's summer :D

Agenda

- `http://workshop.rosedu.org/2014/sesiuni/ml`
- Schedule, Slides and Resources

- Problem topics - Pattern Recognition, Computer Vision
- Plans are to make use of scikit-learn and OpenCV

- Introductory ML has a lot of theory, not as much coding.
- We'll try to keep a balance :)

How popular is ML today?

- What do you think?

How popular is ML today?

- What do you think?

- Lots of companies built their own research labs
- Current focus in the industry is largely on **deep learning**
- Some of the DL pioneers lead these large industry labs

How popular is ML today?

- What do you think?

- Lots of companies built their own research labs
- Current focus in the industry is largely on **deep learning**
- Some of the DL pioneers lead these large industry labs

- Geoffrey Hinton leads some of the research at Google
- Yann LeCun leads Facebook's NY Research Lab
- Andrew Ng leads Baidu's Institute of Deep Learning

How popular is ML today?

- World Cup 2014!

	Prediction Accuracy - Quarterfinals	Prediction Accuracy - Round of 16	Prediction Accuracy - Group Stage
Baidu 百度	100%	100%	58.33%
Microsoft	100%	100%	56.25%
Goldman Sachs	100%	100%	37.5%
Google	75%	100%	/

http://www.marketing-interactive.com/

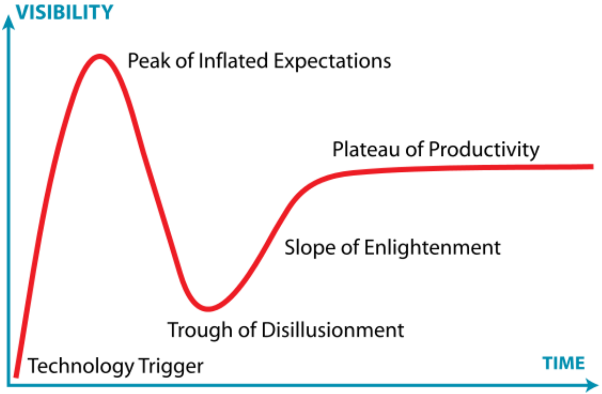
germany-will-win-world-cup-2014-baidu-predicts/

How popular is ML today?

- Largest Class at Stanford in Fall 2013
  CS 229 (Machine Learning by Andrew Ng), 760 Students

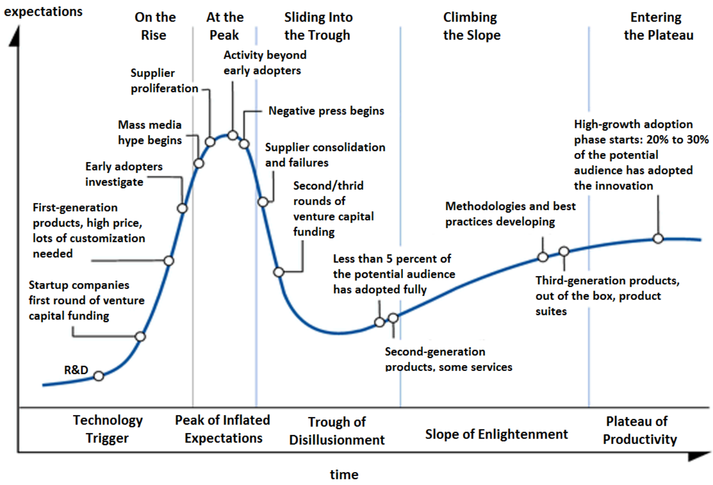
- Largest First Succesful MOOC in Fall 2011
  Introduction to AI (S. Thrun, P. Norvig), 100.000+ Students

ML through the Technology Hype Cycle



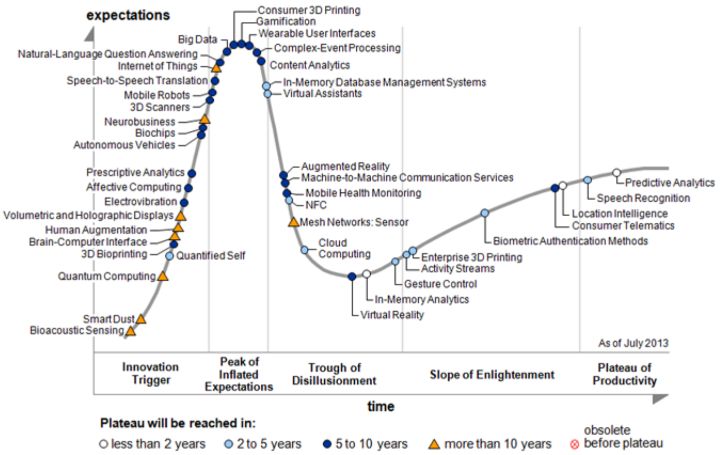
http://en.wikipedia.org/wiki/Hype_cycle

ML through the Technology Hype Cycle



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

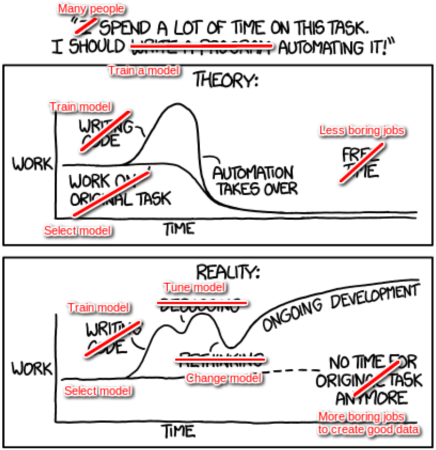
ML through the Technology Hype Cycle



Gartner 2013's Hype Cycle for Emerging Technologies

http://www.gartner.com/

Machine Learning: The Original Motivation



inspired from `http://xkcd.com/1319/`

Fields of Machine Learning

- Supervised
- Semi-supervised
- Unsupervised
- Reinforcement Learning
- Active Learning and Adaptive Control

- ... and a couple of others
- our focus: **supervised learning**

- Due to availability of data, **rule-based approaches** have been replaced by **statistical approaches**

- Due to availability of data, **rule-based approaches** have been replaced by **statistical approaches**
- Rule-based systems remain used when training data is poor or scarce, or simply precise expert knowledge is sufficient to solve the problem

Methodologies in Machine Learning

- Due to availability of data, **rule-based approaches** have been replaced by **statistical approaches**
- Rule-based systems remain used when training data is poor or scarce, or simply precise expert knowledge is sufficient to solve the problem
- Think of sub-parts of an NLP system

The ML Process in Supervised Learning

- Collect input-output examples from experts
- Learn a function that is able to map input to output

The ML Process in Supervised Learning

- Collect input-output examples from experts
- Learn a function that is able to map input to output

- Given a set of training examples $\{(x_i, f(x_i))\}$
- Learn a good approximation to f

Formal Setting of Supervised Learning

- Do we have any statistics involved already?

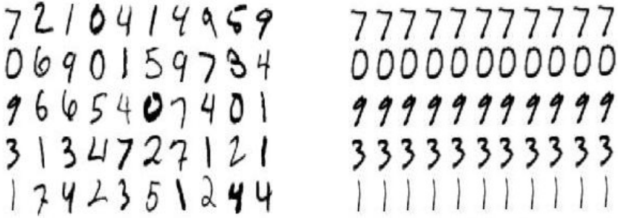
Formal Setting of Supervised Learning

- Do we have any statistics involved already?
- How are the labeled data points obtained?

Formal Setting of Supervised Learning

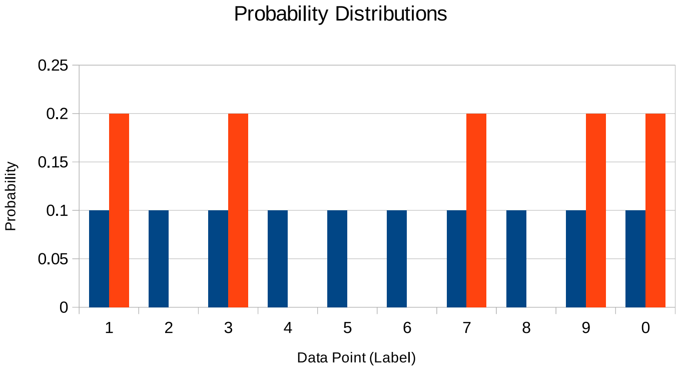
- Do we have any statistics involved already?
- How are the labeled data points obtained?

- What differs between these datasets?



Formal Setting of Supervised Learning

- Data points are drawn **independently and identically distributed** according to a probability distribution P(x, y)



Formal Setting of Supervised Learning

- The **i.i.d.** assumption is a very strong one
- The distribution is unknown, but it is assumed that all data encountered is generated by it, e.g. train/test set

- The **i.i.d.** assumption is a very strong one
- The distribution is unknown, but it is assumed that all data encountered is generated by it, e.g. train/test set

- *Note*: statistical learning theory is the branch that formally proves that, under these assumptions, learning can occur

Supervised Learning Workflow

- **Training set** generated i.i.d. from P(x, y)
- A **learning algorithm** builds a classifier f, by going through the examples in the training set
- **Test point** (x, y) drawn from P

Supervised Learning Workflow

- **Training set** generated i.i.d. from P(x, y)
- A **learning algorithm** builds a classifier f, by going through the examples in the training set
- **Test point** (x, y) drawn from P

- The built classifier gets x and predicts $\hat{y} = f(x)$
- Compute the loss, according to a **loss function** $\mathcal{L}(y, \hat{y})$

Supervised Learning Workflow

- **Training set** generated i.i.d. from P(x, y)
- A **learning algorithm** builds a classifier f, by going through the examples in the training set
- **Test point** (x, y) drawn from P

- The built classifier gets x and predicts $\hat{y} = f(x)$
- Compute the loss, according to a **loss function** $\mathcal{L}(y, \hat{y})$

- Goal: find f that minimizes the *expected* loss

Supervised Learning Workflow: Spam Detection

- **P(x, y)**: probability distribution of email message x and their labels y ("spam" or "not spam")
- **Train set**: email message x manually labeled by the user
- **Learning algorithm**: SVMs, Logistic Regression, etc.
- f: classifier output by the learning algorithm
- **Test point**: email message without its label (hidden)

- What about the **loss function**?

Supervised Learning Workflow: Spam Detection Loss Function

	true label	
predicted label	spam	not spam
spam	0	5 (*false positive*)
not spam	1 (*false negative*)	0

Supervised Learning Workflow: Spam Detection Loss Function

	true label	
predicted label	spam	not spam
spam	0	5 (*false positive*)
not spam	1 (*false negative*)	0

- Penalize more when a message is classified as spam, but it's actually not: the user suffers by losing emails

Supervised Learning Terminology: Classification / Regression

- Tasks which require discrete output, binary or fixed number of classes, are referred to as **classification tasks**

Supervised Learning Terminology: Classification / Regression

- Tasks which require discrete output, binary or fixed number of classes, are referred to as **classification tasks**
- Tasks which require continous output, are referred to as **regression tasks**

Quick Refresh and Terminology

- So why is P(x, y) relevant?

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

Quick Refresh and Terminology

- So why is P(x, y) relevant?

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

- Bayes Rule

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)}$$

Quick Refresh and Terminology

- So why is P(x, y) relevant?

$$P(x,y) = P(x|y)P(y) = P(y|x)P(x)$$

- Bayes Rule

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)}$$

- Terminology

$$\text{Posterior} = \frac{\text{Prior x Likelihood}}{\text{Evidence}}$$

rosedu SUMMER WORKSHOPS

- Learn a classifier: a function f

Machine Learning: Three Approaches

- Learn a classifier: a function f
- Learn a conditional distribution $P(y|x)$ [**discriminative**]

Machine Learning: Three Approaches

- Learn a classifier: a function f
- Learn a conditional distribution $P(y|x)$ [**discriminative**]
- Learn the joint distribution $P(x, y)$ [**generative**]

Machine Learning: Discriminative Model

- Estimates directly the posterior probability $P(y|x)$
- Does not model the joint probability distribution $P(x, y)$
- Focused - learns only how to discrimate between classes

Machine Learning: Discriminative Model

- Estimates directly the posterior probability $P(y|x)$
- Does not model the joint probability distribution $P(x, y)$
- Focused - learns only how to discrimate between classes

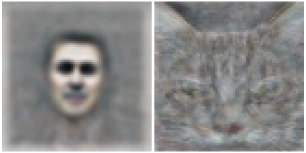
- Models: Logistic Regression, SVMs, (Traditional) Neural Networks and others

- Models class-conditionals and prior probabilities
- Generative because one can use the model to generate synthetic data points, as you can explicitly compute P(x, y)
- Requires more knowledge or assumptions

Machine Learning: Generative Model

- Models class-conditionals and prior probabilities
- Generative because one can use the model to generate synthetic data points, as you can explicitly compute P(x, y)
- Requires more knowledge or assumptions

- Models: Naive Bayes, Gaussian Mixture Models, Deep Belief Networks



Synthetic human and cat face from deep network [Google, Inc.]

Machine Learning: Learn a classifier

- To some extent this method can be seen as part of discriminative methods, but it does not estimate $P(y|x)$
- It simply learns a threshold and differentiates between classes

- The popular '60s simple Perceptron algorithm

Linear Models: Motivation

- Easy to understand and usually trivial to implement

Linear Models: Motivation

- Easy to understand and usually trivial to implement
- Strong performance, despite their simplicity

Linear Models: Motivation

- Easy to understand and usually trivial to implement
- Strong performance, despite their simplicity
- Non-linear models (e.g. multilayer nn, naive bayes), sometimes have an extra top-layer linear model (softmax)

Linear Models: Motivation

- Easy to understand and usually trivial to implement
- Strong performance, despite their simplicity
- Non-linear models (e.g. multilayer nn, naive bayes), sometimes have an extra top-layer linear model (softmax)
- Transform the input into a higher dimension, and use the same linear model for non-linearly separable data

Linear Models: Motivation

- Easy to understand and usually trivial to implement
- Strong performance, despite their simplicity
- Non-linear models (e.g. multilayer nn, naive bayes), sometimes have an extra top-layer linear model (softmax)
- Transform the input into a higher dimension, and use the same linear model for non-linearly separable data

- Definitely worth studying them!
- Models: Perceptron, Logistic Regression, SVMs, Linear Discriminant Analysis, etc.

Linear Models: Linear Threshold Unit

Point: $x =< x_1, x_2, ..., x_n >$ (n features, $x_i \in \mathbb{R}$)
Weights: $w =< w_0, w_1, w_2, ..., w_n >$ (n weights, $w_i \in \mathbb{R}$)

$$h(x) = \begin{cases} +1 & w_1 x_1 + ... + w_n x_n \geq w_0 \\ -1 & otherwise \end{cases}$$

We simply transform $x =< -1, x_1, ..., x_n >$ and use $w^T x \geq 0$

Linear Models: Batch Perceptron

Input: training examples $(\mathbf{x}_i, y_i)$
$w = (0, ..., 0)$
Repeat until convergence

  g = (0, ..., 0) // gradient vector

  for i = 1 to N do

    if $(y_i w^T x_i < 0)$ // x_i misclassified

      for j = 1 to n do

        $g_j = g_j - y_i x_{ij}$

  $g = g/N$ // N = total input points

  $w = w - \eta g$ // move in the right direction

There's math to be understood behind g (gradient descent method), η (learning rate properties to ensure convergence)

Linear Models: Batch Perceptron

Also the fundamental LTU unit in a Neural Network

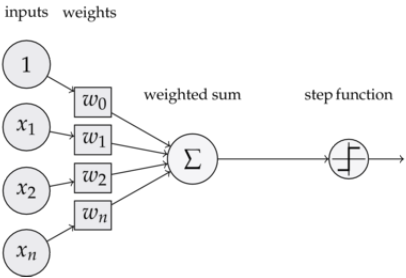


image from `http://blog.dbrgn.ch`

Perceptron: Linear Separability

- Linear Separability Requirement

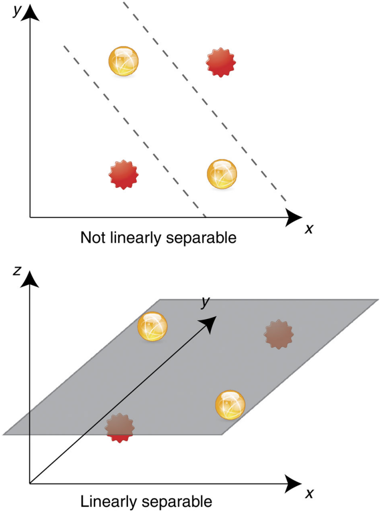


image from `http://nature.com`

Perceptron: Model Complexity

- What is the measure of model complexity for the perceptron?

Perceptron: Model Complexity

- What is the measure of model complexity for the perceptron?

- Number of weights (number of features)
- Numerical value of the weights: the larger, the more complexity they add

Perceptron: Model Complexity

- What is the measure of model complexity for the perceptron?

- Number of weights (number of features)
- Numerical value of the weights: the larger, the more complexity they add

- Linear models are less complex than quadratic ones

Types of Errors

- **Training Error** [fraction of training examples misclassified]
- **Validation Error** [fraction of test examples misclassified]
- **Generalization Error** [probability of misclassifiying new examples]

Types of Errors

- **Training Error** [fraction of training examples misclassified]
- **Validation Error** [fraction of test examples misclassified]
- **Generalization Error** [probability of misclassifiying new examples]

- Competitions and benchmarks generally split data: training set, validation set and test set

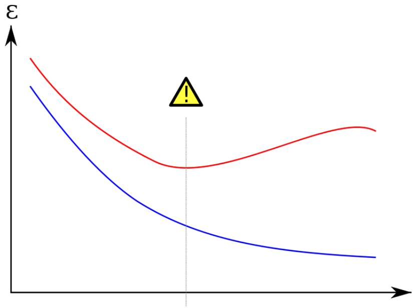
Model Complexity: Overfitting

- How long should we train?
- Should we perfectly learn the training data?

Model Complexity: Overfitting

- How long should we train?
- Should we perfectly learn the training data?

- Validation Error [red] vs Training Error [blue]



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

Model Complexity: What is the tradeoff?

- Amount of data we have

  3 samples, 5 samples, 10^k samples?

Model Complexity: What is the tradeoff?

- Amount of data we have

  3 samples, 5 samples, 10^k samples?

- Complexity of our model

  Linear, Quadratic, Weight Values, etc.

Model Complexity: What is the tradeoff?

- Amount of data we have

  3 samples, 5 samples, 10^k samples?

- Complexity of our model

  Linear, Quadratic, Weight Values, etc.

- Accuracy of the model on new data points

  We want to make sure we do a good prediction job,
  not that we explain the nature of the problem

Model Complexity: Guiding Principles

- When you have two competing theories making exactly the same predictions, the simpler one is the better. [Occam's Razor]

- When solving a problem of interest, do not solve a more general problem as an intermediate step. [Vapnik's Principle]

Model Complexity: Regularization

- **Regularization**: penalize the magnitude of the weights

- **Regularization**: penalize the magnitude of the weights
- $L' = \sum_i \frac{1}{N}\mathcal{L}(f_w(x_i), y_i) + \lambda \sum_j w_j^2$
- After following the math behind it, the gradient step is:

$$g_j = g_j - y_i x_{ij} - 2\lambda \sum_j w_j$$

Model Complexity: Regularization

- **Regularization**: penalize the magnitude of the weights
- $L' = \sum_i \frac{1}{N}\mathcal{L}(f_w(x_i), y_i) + \lambda \sum_j w_j^2$
- After following the math behind it, the gradient step is:

$$g_j = g_j - y_i x_{ij} - 2\lambda \sum_j w_j$$

- This is the l2 penalty, but l1 and elastic net are used too

Model Complexity: What Remains

- Hooray! Less complexity... but new parameter λ.
  How do we pick it?

Model Complexity: What Remains

- Hooray! Less complexity... but new parameter λ.
  How do we pick it?
- We didn't talk yet about the bias-variance tradeoff.
  How do we measure and reduce them?

Model Complexity: What Remains

- Hooray! Less complexity... but new parameter λ. How do we pick it?
- We didn't talk yet about the bias-variance tradeoff. How do we measure and reduce them?
- Let's end the day with a scikit-learn tutorial and see how much is implemented already for us

Quick tutorial with scikit-learn

- Scikit-learn started as GSoC project and is getting increasingly popular, having many algorithms implemented
- Quick introductory tutorial

  `http://scikit-learn.org/stable/tutorial/basic/tutorial.html`
- Goals after the introductory tutorial
  1. Load the iris training dataset
  2. Use Perceptron to **fit** and **predict**
  3. Fit samples [1:99] (only two classes)
  4. Predict samples 1 and 100
  5. Can you figure out how to use the l2 penalty?