A Tutorial on Machine Learning Methods

Varun Chandola

Computer Science and Engineering
Computational and Data-Enabled Science and Engineering
University at Buffalo, State University of New York

March 24, 2016

Tutorial Overview

Part I - Methods (Chandola)

- Today from 1.00 PM - 3.00 PM
- 330 Student Union
- Introduction and general overview of methods and concepts

Part II - Applications (Bauman and Hachmann)

- Tomorrow from 9.00 AM - 11.00 AM
- 280 Park Hall
- II.a - Bayesian Methods and Partial Differential Equations
- II.b - Machine Learning for Computational Chemistry

Setup

- Python and iPython Notebooks
- `scikit-learn` - ML library in Python

Follow Along

- Slides: `https://github.com/ubdsgroup/cdsedaystutorial/blob/master/talk.pdf`
- Notebook: `http://nbviewer.jupyter.org/github/ubdsgroup/cdsedaystutorial/blob/master/MachineLearningBasics.ipynb`
- Git Repo: `https://github.com/ubdsgroup/cdsedaystutorial.git`

What makes a machine intelligent?

1. Talk. See. Hear.
    - Natural Language Processing, Computer Vision, Speech Recognition
2. Store. Access. Represent. (*Knowledge*)
    - Ontologies. Semantic Networks. Information Retrieval.
3. Reason.
    - Mathematical Logic. Bayesian Inference.
4. **Learn.**
    - Improve with Experience

What makes a machine intelligent?

1. Talk. See. Hear.
    - Natural Language Processing, Computer Vision, Speech Recognition
2. Store. Access. Represent. (*Knowledge*)
    - Ontologies. Semantic Networks. Information Retrieval.
3. Reason.
    - Mathematical Logic. Bayesian Inference.
4. **Learn.**
    - Improve with Experience
        - Machine Learning

What is Machine Learning?

- Computers learn without being **explicitly programmed**.
    - Arthur Samuel (1959)
- A computer program learns from experience E with respect to some task T, if its performance P while performing task T improves over E.
    - Tom Mitchell (1989)

Why Machine Learning?

- Machines that know everything from the beginning?
  - Too bulky. Creator already knows everything. Fails with *new* experiences.
- Machines that *learn*?
  - Compact. Learn what is *necessary*.
  - Adapt.
  - Assumption: Future experiences are not too different from past experiences.
    - Have (structural) relationship.

Why is ML so popular?

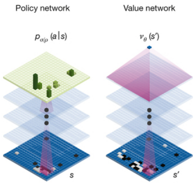
- Learn to do tasks that are "impossible" (or infeasible) for humans
- Automatically learn from past to operate in the unknown future
- In CS, we focus on issues such as scalability, usability, performance, error handling, complexity, convergence, *big data challenges*
- What is in it for non-AI and non-CS folks?
  - All of us are writing tiny "robots"
  - Almost all machine learning $\equiv$ learning from **data**

Topics in Machine Learning

- Tasks
  - **Classification**, **Regression**, Ranking, Latent Variable Modeling (**clustering**, **dimensionality reduction**, dictionary learning)
- Learning types
  - **Supervised**, **Unsupervised**, Semi-supervised, Reinforcement
- Learning strategies
  - **Error minimization**, Statistical, **Heuristic search**
- Inference models
  - Parametric: **Probabilistic, Linear, Non-linear**
  - Non-parametric
- Cross-cutting issues
  - Handling structured inputs/outputs, **Stability/Generalizability, Interpretability**, Incorporating domain knowledge, Scalability

What else will I not cover?

- What is Deep Learning?
- How does AlphaGo work?
  - It does use ML (neural networks to be exact)
  - `https://www.tastehit.com/blog/google-deepmind-alphago-how-it-works/`
- How to make money in the stock market with this?

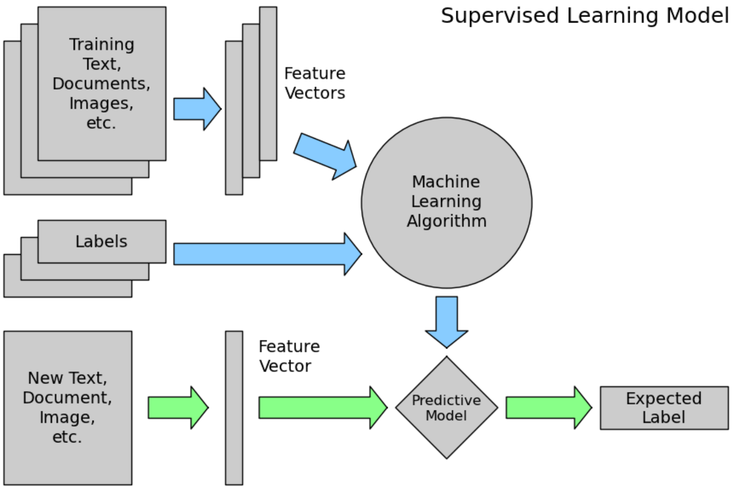


Src: Nature, 2016

Classification or Regression

- Given input $\mathbf{x}$, predict target $\mathbf{y}$
- Assumptions:
  1. $\mathbf{x} \in \Re^D$, i.e., $\mathbf{x}$ is represented as a vector of D real values – attributes or features
  2. Regression: $y \in \Re$
  3. Classification $y \in \{+1, -1\}$ or $y \in \{1, 2, \ldots, k\}$

A Data Science View



Parametric vs. Non-parametric Models

Parametric

- Model represented as a set of *parameters*
- Concise, easy to store
- Making assumptions (inductive bias)
- *Examples*: Linear regression, Neural networks

Non-parametric

- No (finite) set of parameters
- Use entire training data for inference on a new example
- No assumptions
- Complexity grows with data
- *Examples*: Gaussian process regression, Nearest neighbor classification

Supervised Learning for Parametric Models

- Given a training data set, learn (or estimate) the model parameters

Inverse Problem Theory - Physical Systems

- Forward modeling: Given system parameters one can compute the behavioral output of the system (apply physical laws)
- Inverse modeling: Given the observations, *infer* the values of the system parameters

Possible approaches

- Find parameters that minimize error on training data
  - Least squares regression, neural networks, support vector machines
- Find parameters that maximize the likelihood (or posterior) of the data distribution
  - Naive Bayes, Bayesian Regression
- Search for best solution using greedy heuristics
  - Decision trees, Random Forests

Error Minimization Methods

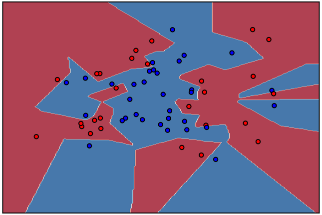
1. Define an error as a function of the model parameters ($\mathbf{\Theta}$) and the training data
2. Minimize the error to obtain optimal parameters

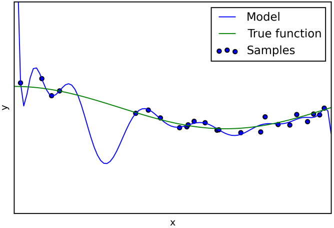
$$\begin{aligned}\widehat{\mathbf{\Theta}} &= \arg\min_{\mathbf{\Theta}} \quad J(\mathbf{\Theta}) \\ &= \arg\min_{\mathbf{\Theta}} \quad \sum_{i=1}^{N}(y_i - \hat{y}_i)^2\end{aligned}$$

- The squared error between true output (y_i) and predicted output ($\hat{y}_i$) is not the only choice

Perils of Unconstrained Error Minimization Approaches

- Favor complex solutions
- Do perfectly on training data (overfitting)
- No guarantee to work well on unseen data (generalizability)
- Learning theory: Complex models $\Rightarrow$ Poor generalizability





How to Control Overfitting?

- Use simpler models (linear instead of polynomial)
  - Might have poor results (underfitting)
- Use regularized complex models

$$\widehat{\mathbf{\Theta}} = \arg\min_{\mathbf{\Theta}} J(\mathbf{\Theta}) + \alpha R(\mathbf{\Theta})$$

- $R()$ corresponds to the penalty paid for complexity of the model

More about Regularization

- **Probabilistic methods**: impose regularization through priors on the parameters
- **Search methods**: avoid searching along paths that lead to complex results
- Some regularization methods have other benefits too:
  - Feature selection
  - Imposing sparsity constraints (interpretability)
  - Incorporating domain knowledge
  - Reducing impact of correlated inputs.

Regularization for Linear Models

- Linear regression (for regression) and logistic regression (for classification) are two popular linear models

$$y \sim \mathcal{N}(\mathbf{w}^\top \mathbf{x}, \sigma^2)$$

$$y \sim Bernoulli(sigmoid(\mathbf{w}^\top \mathbf{x}))$$

- where $sigmoid(a) = \frac{1}{1+exp(-a)}$
- One can use polynomial expansion to model non-linear dependencies (the model is still linear in the weights)
- How to control the complexity?

Examples of Regularization

Ridge Regression

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w}) + \alpha\|\mathbf{w}\|^2$$

- Also known as l_2 or *Tikhonov* regularization
- Helps in reducing impact of correlated inputs

LASSO

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w}) + \alpha|\mathbf{w}|$$

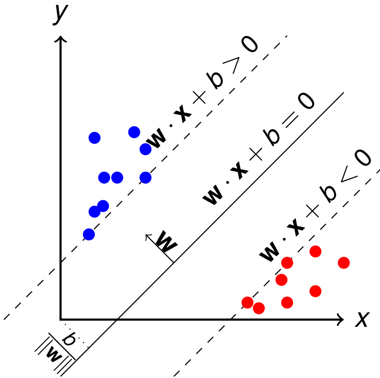
- Also known as l_1 regularization
- Helps in feature selection – favors sparse solutions

Regularization – A Research Area by Itself

- For linear models, many other forms exist
    1. Elastic net
    2. Group Lasso
    3. Tree Structured Lasso
    4. Network Lasso
- Each formulation requires a unique optimization problem to be solved

Line as a Decision Surface

- Decision boundary represented by the hyperplane **w**
- For binary classification, **w** points **towards** the positive class



Decision Rule

$$y = sign(\mathbf{w}^\top\mathbf{x} + b)$$

- $\mathbf{w}^\top\mathbf{x} + b > 0 \Rightarrow y = +1$
- $\mathbf{w}^\top\mathbf{x} + b < 0 \Rightarrow y = -1$

Support Vector Machines

- A hyperplane based classifier defined by $\mathbf{w}$ and b
- Find hyperplane with *maximum separation margin* on the training data

SVM Prediction Rule

$$y = sign(\mathbf{w}^\top \mathbf{x} + b)$$

SVM Learning

- **Input**: Training data $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
- **Objective**: Learn $\mathbf{w}$ and b that maximizes the margin

Going Non-Linear with SVMs

- SVMs are linear classifiers
- To model non-linear dependencies
  1. Basis vector expansion

  $$\Phi(x) = 1, x, x^2, \ldots, x^p$$

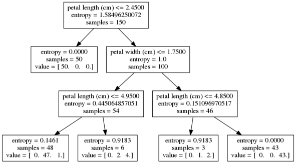
  still restrictive
  2. The **kernel trick**
     - Allows for (implicit) mapping of data into a new high dimensional space through kernel functions

     $$k(x_1, x_2) \equiv \Phi_1^\top \Phi_2$$

     - The classes are assumed to be linearly separable in that space
     - Most popular is Radial Basis Function (RBF) kernel that allows mapping data into infinite dimensional space!

Decision Trees

- Learning algorithm searches for a sequence of rules that organize a data set
- Learning is one variable at a time
    - Find boundaries between sections of data (a sub-decision)
- Which variable to choose and where to set the boundary?
    - Information Gain, Gini Impurity, Variance Reduction



Why Doesn’t Everyone Use Decision Trees?

Strengths

- Both regression and classification
- Handle all types of attributes, missing data, noise
- Validate through statistical tests
- Interpretable
- No need for normalization

Weaknesses

- No optimality guarantee (learning problem is NP-complete)
- Favor attributes with more levels or categories
- Become too complex for certain types of problems

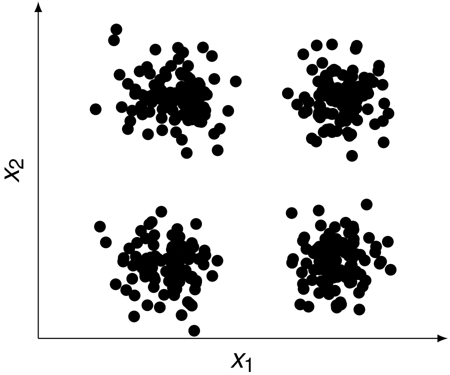
Extensions of Decision Trees

- Using conditional inference
- Regression trees
- Boosted Trees
- Random Forests

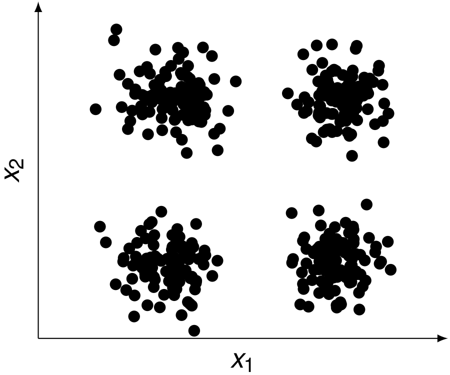
What is Clustering?

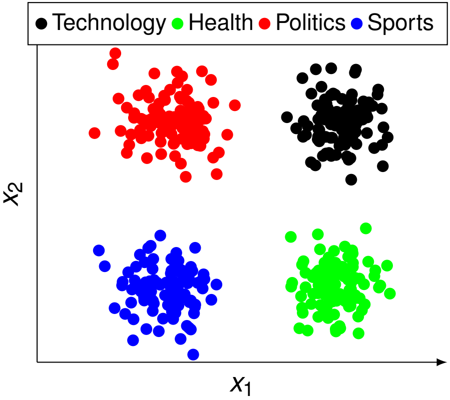
- Grouping similar things together
- A notion of a similarity or distance metric
- A type of **unsupervised learning**
    - Learning without any labels or target

Expected Outcome of Clustering



Expected Outcome of Clustering





K-Means Clustering

- **Objective**: Group a set of N points ($\in \Re^D$) into K clusters.

K-Means Clustering

- **Objective**: Group a set of N points ($\in \Re^D$) into K clusters.

1. **Start** with k *randomly initialized* points in D dimensional space
    - Denoted as $\{\mathbf{c}_k\}_{k=1}^K$
    - Also called *cluster centers*

K-Means Clustering

- **Objective**: Group a set of N points ($\in \Re^D$) into K clusters.

1. **Start** with k *randomly initialized* points in D dimensional space
   - Denoted as $\{\mathbf{c}_k\}_{k=1}^K$
   - Also called *cluster centers*
2. **Assign** each input point $\mathbf{x}_n$ ($\forall n \in [1, N]$) to cluster k, such that:

$$\min_k \text{dist}(\mathbf{x}_n, \mathbf{c}_k)$$

K-Means Clustering

- **Objective**: Group a set of N points ($\in \Re^D$) into K clusters.

1. **Start** with k *randomly initialized* points in D dimensional space
   - Denoted as $\{\mathbf{c}_k\}_{k=1}^{K}$
   - Also called *cluster centers*
2. **Assign** each input point $\mathbf{x}_n$ ($\forall n \in [1, N]$) to cluster k, such that:

$$\min_k \text{dist}(\mathbf{x}_n, \mathbf{c}_k)$$

3. **Revise** each cluster center $\mathbf{c}_k$ using all points assigned to cluster k

K-Means Clustering

- **Objective**: Group a set of N points ($\in \Re^D$) into K clusters.

1. **Start** with k *randomly initialized* points in D dimensional space
   - Denoted as $\{\mathbf{c}_k\}_{k=1}^K$
   - Also called *cluster centers*
2. **Assign** each input point $\mathbf{x}_n$ ($\forall n \in [1, N]$) to cluster k, such that:
$$\min_k \text{dist}(\mathbf{x}_n, \mathbf{c}_k)$$
3. **Revise** each cluster center $\mathbf{c}_k$ using all points assigned to cluster k
4. **Repeat** 2

Variants of K-Means

- Finding distance
  - Euclidean distance is popular
- Finding cluster centers
  - Mean for K-Means
  - Median for k-medoids

Choosing Parameters

1. Similarity/distance metric
   - Can use non-linear transformations
   - K-Means with Euclidean distance produces "circular" clusters
2. How to set k?
   - Trial and error
   - How to evaluate clustering?
   - K-Means objective function

$$J(\mathbf{c}, \mathbf{R}) = \sum_{n=1}^{N} \sum_{k=1}^{K} R_{nk} \|\mathbf{x}_n - \mathbf{c}_k\|^2$$

   - $\mathbf{R}$ is the cluster assignment matrix

$$R_{nk} = \begin{cases} 1 & \text{If } \mathbf{x}_n \in \text{ cluster } k \\ 0 & \text{Otherwise} \end{cases}$$

Initialization Issues

- Can lead to wrong clustering
- Better strategies
  1. Choose first centroid randomly, choose second farthest away from first, third farthest away from first and second, and so on.
  2. Make multiple runs and choose the best

Strengths and Limitations of K-Means

Strengths

- Simple
- Can be extended to other types of data
- Easy to parallelize

Weaknesses

- Circular clusters (not with kernelized versions)
- Choosing K is always an issue
- Not guaranteed to be optimal
- Works well if natural clusters are round and of equal densities
- **Hard Clustering**

Issues with K-Means

- "Hard clustering"
- Assign every data point to exactly one cluster
- **Probabilistic Clustering**
  - Each data point can belong to multiple clusters with varying probabilities
  - In general

$$P(\mathbf{x}_i \in C_j) > 0 \quad \forall j = 1 \ldots K$$

  - For hard clustering probability will be 1 for one cluster and 0 for all others

Latent Variable Models

- Consider a probability distribution parameterized by $\boldsymbol{\theta}$
- Generates samples ($\mathbf{x}$) with probability $p(\mathbf{x}|\boldsymbol{\theta})$

2-step generative process

Latent Variable Models

- Consider a probability distribution parameterized by $\boldsymbol{\theta}$
- Generates samples ($\mathbf{x}$) with probability $p(\mathbf{x}|\boldsymbol{\theta})$

2-step generative process

1. Distribution generates the hidden variable

Latent Variable Models

- Consider a probability distribution parameterized by $\boldsymbol{\theta}$
- Generates samples ($\mathbf{x}$) with probability $p(\mathbf{x}|\boldsymbol{\theta})$

2-step generative process

1. Distribution generates the hidden variable
2. Distribution generates the observation, given the hidden variable

More About Latent Variable Models

- The observed random variable $\mathbf{x}$ depends on a hidden random variable $\mathbf{z}$
- $\mathbf{z}$ is generated using a *prior* distribution - $p(\mathbf{z})$
- $\mathbf{x}$ is generated using $p(\mathbf{x}|\mathbf{z})$
- Different combinations of $p(\mathbf{z})$ and $p(\mathbf{x}|\mathbf{z})$ give different latent variable models
    1. Mixture Models
    2. Factor analysis
    3. Probabilistic Principal Component Analysis (PCA)
    4. Latent Dirichlet Allocation (LDA)
    5. Dictionary Learning, Sparse Coding

Other Resources

- My UB course - CSE474/574 (`http://www.cse.buffalo.edu/~chandola/machinelearning.html`)
- Beautiful ML (`http://www.r2d3.us/visual-intro-to-machine-learning-part-1/`)