Machine Learning Tutorial

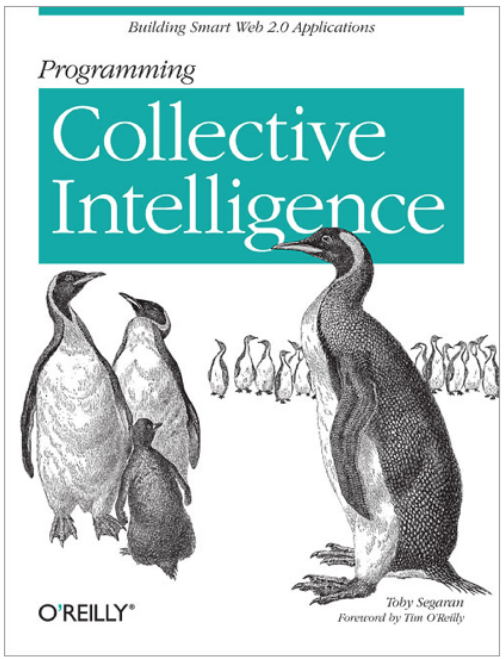
Wrap-up

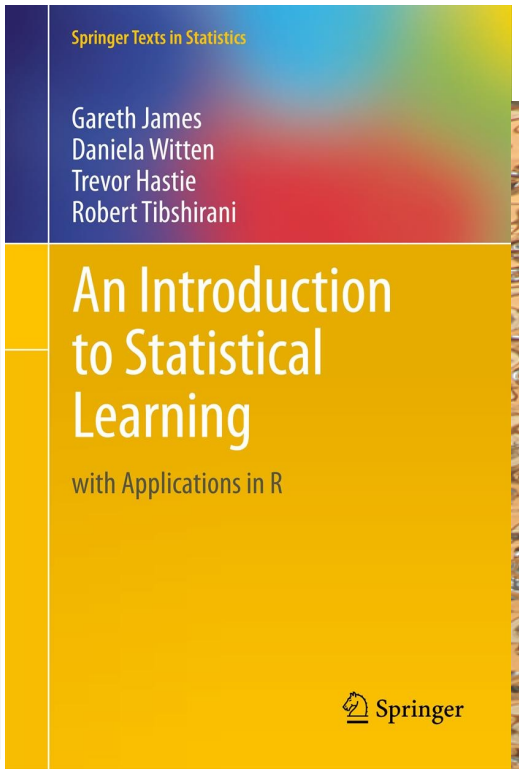
Jake Hofman

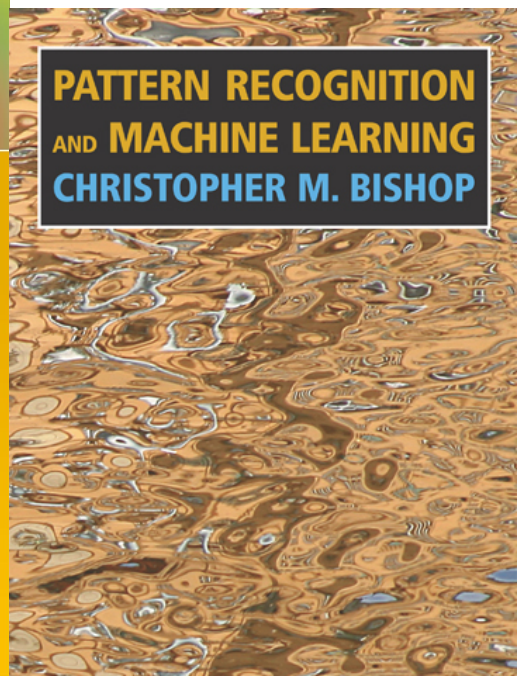
Microsoft Research

March 16, 2018

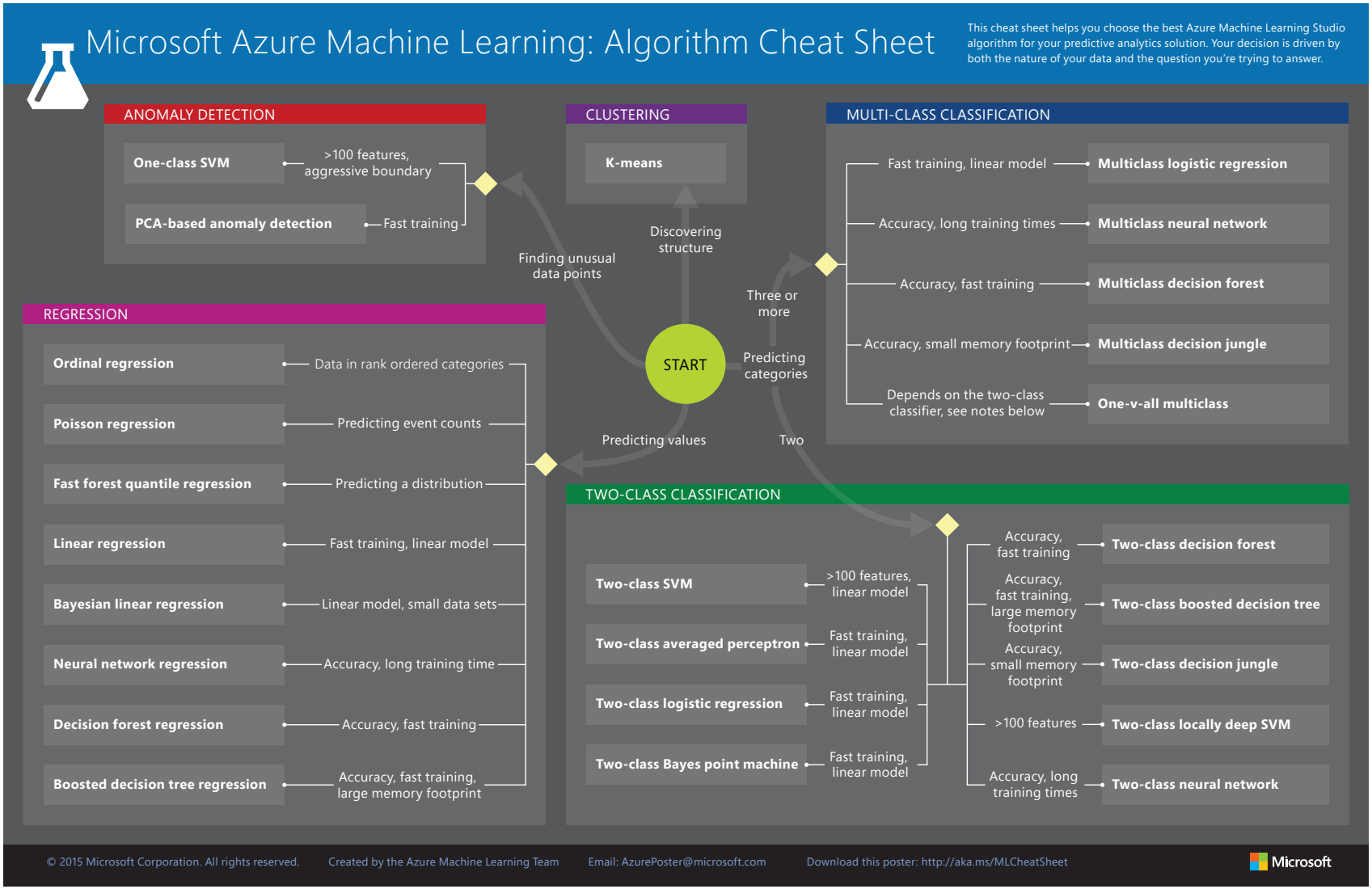
Books


Building Smart Web 2.0 Applications
Programming
Collective Intelligence
O'REILLY
Toby Segaran
Foreword by Tim O'Reilly


Springer Texts in Statistics
Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani
An Introduction to Statistical Learning
with Applications in R
Springer

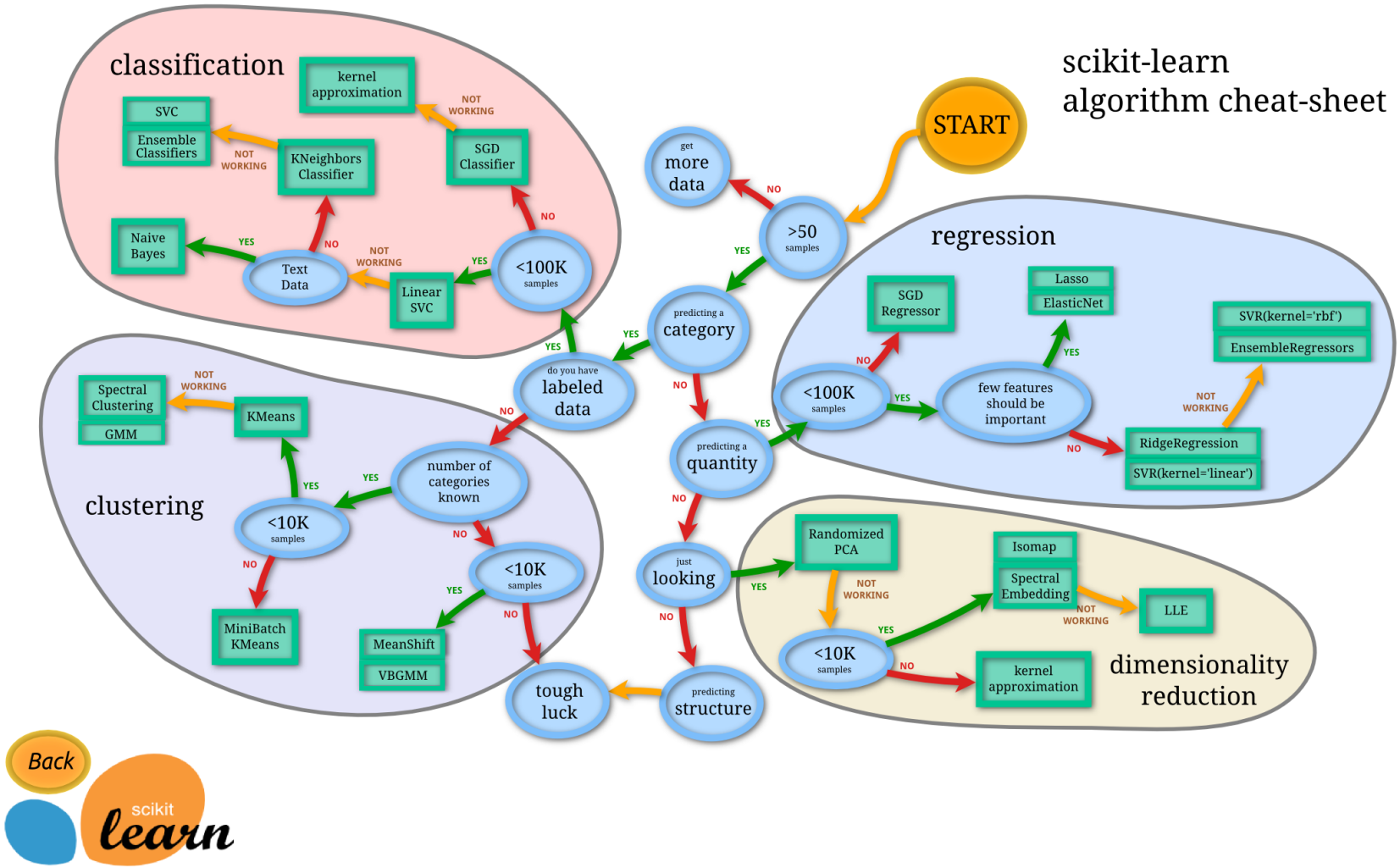

PATTERN RECOGNITION
AND MACHINE LEARNING
CHRISTOPHER M. BISHOP

Cheatsheets



http://bit.ly/howtochooseml

Cheatsheets



http://bit.ly/scikitcheat

Cheatsheets

Cheat Sheet: Algorithms for Supervised- and Unsupervised Learning [1]

Algorithm	Description	Model	Objective	Training	Regularisation	Complexity	Non-linear	Online learning
k-nearest neighbour	The label of a new point $\hat{x}$ is classified with the most frequent label $\hat{t}$ of the k nearest training instances.	$\hat{t} = \arg\max_c \sum_{i:x_i \in N_k(\boldsymbol{x},\hat{x})} \delta(t_i, C)$ • $N_k(\boldsymbol{x}, \hat{x}) \leftarrow k$ points in $\boldsymbol{x}$ closest to $\hat{x}$ • Euclidean distance formula: $\sqrt{\sum_{i=1}^D (x_i - \hat{x}_i)^2}$ • $\delta(a, b) \leftarrow 1$ iff $a = b$; 0 o/w	No optimisation needed.	Use cross-validation to learn the appropriate k; otherwise no training, classification based on existing points.	k acts as to regularise the classifier: as $k \to N$ the boundary becomes smoother.	$\mathcal{O}(NM)$ space complexity, since all training instances and all their features need to be kept in memory.	Natively finds non-linear boundaries.	To be added.
Naive Bayes	Learn $p(\mathcal{C}_k\|\boldsymbol{x})$ by modelling $p(\boldsymbol{x}\|\mathcal{C}_k)$ and $p(\mathcal{C}_k)$, using Bayes' rule to infer the class conditional probability. Assumes each feature independent of all others, ergo 'Naive.'	$y(\boldsymbol{x}) = \arg\max_k p(\mathcal{C}_k\|\boldsymbol{x})$ $= \arg\max_k p(\boldsymbol{x}\|\mathcal{C}_k) \times p(\mathcal{C}_k)$ $= \arg\max_k \prod_{i=1}^D p(x_i\|\mathcal{C}_k) \times p(\mathcal{C}_k)$ $= \arg\max_k \sum_{i=1}^D \log p(x_i\|\mathcal{C}_k) + \log p(\mathcal{C}_k)$	No optimisation needed.	**Multivariate likelihood** $p(\boldsymbol{x}\|\mathcal{C}_k) = \sum_{i=1}^D \log p(x_i\|\mathcal{C}_k)$ $p_{\text{MLE}}(x_i = v\|\mathcal{C}_k) = \frac{\sum_{j=1}^N \delta(t_j = \mathcal{C}_k \wedge x_{ji} = v)}{\sum_{j=1}^N \delta(t_j = \mathcal{C}_k)}$ **Multinomial likelihood** $p(\boldsymbol{x}\|\mathcal{C}_k) = \prod_{i=1}^D p(\text{word}_i\|\mathcal{C}_k)^{x_i}$ $p_{\text{MLE}}(\text{word}_i = v\|\mathcal{C}_k) = \frac{\sum_{j=1}^N \delta(t_j = \mathcal{C}_k) \times x_{ji}}{\sum_{j=1}^N \sum_{d=1}^D \delta(t_j = \mathcal{C}_k) \times x_{di}}$ …where: • x_{ji} is the count of word i in test example j; • x_{di} is the count of feature d in test example j. **Gaussian likelihood** $p(\boldsymbol{x}\|\mathcal{C}_k) = \prod_{i=1}^D \mathcal{N}(v; \mu_{ik}, \sigma_{ik})$	Use a Dirichlet prior on the parameters to obtain a MAP estimate. **Multivariate likelihood** $p_{\text{MAP}}(x_i = v\|\mathcal{C}_k) = \frac{(\beta_i - 1) + \sum_{j=1}^N \delta(t_j = \mathcal{C}_k \wedge x_{ji} = v)}{\|x_i\|(\beta_i - 1) + \sum_{j=1}^N \delta(t_j = \mathcal{C}_k)}$ **Multinomial likelihood** $p_{\text{MAP}}(\text{word}_i = v\|\mathcal{C}_k) = \frac{(\alpha_i - 1) + \sum_{j=1}^N \delta(t_j = \mathcal{C}_k) \times x_{ji}}{\sum_{j=1}^N \sum_{d=1}^D (\delta(t_j = \mathcal{C}_k) \times x_{di}) - D + \sum_{d=1}^D \alpha_d}$	$\mathcal{O}(NM)$, each training instance must be visited and each of its features counted.	Can only learn linear boundaries for multivariate/multinomial attributes. With Gaussian attributes, quadratic boundaries can be learned with uni-modal distributions.	To be added.
Log-linear	Estimate $p(\mathcal{C}_k\|\boldsymbol{x})$ directly, by assuming a maximum entropy distribution and optimising an objective function over the conditional entropy distribution.	$y(\boldsymbol{x}) = \arg\max_k p(\mathcal{C}_k\|\boldsymbol{x})$ $= \arg\max_k \sum_m \lambda_m \phi_m(\boldsymbol{x}, \mathcal{C}_k)$ …where: $p(\mathcal{C}_k\|\boldsymbol{x}) = \frac{1}{Z_\lambda(\boldsymbol{x})} e^{\sum_m \lambda_m \phi_m(\boldsymbol{x}, \mathcal{C}_k)}$ $Z_\lambda(\boldsymbol{x}) = \sum_k e^{\sum_m \lambda_m \phi_m(\boldsymbol{x}, \mathcal{C}_k)}$	Minimise the negative log-likelihood: $\mathcal{L}_{\text{MLE}}(\lambda, \mathcal{D}) = \prod_{(x,t)\in\mathcal{D}} p(t\|x) = -\sum_{(x,t)\in\mathcal{D}} \log p(t\|x)$ $= \sum_{(x,t)\in\mathcal{D}} \left(\log Z_\lambda(x) - \sum_m \lambda_m \phi_m(x,t) \right)$ $= \sum_{(x,t)\in\mathcal{D}} \left(\log \sum_k e^{\sum_m \lambda_m \phi_m(x,\mathcal{C}_k)} - \sum_m \lambda_m \phi_m(x,t) \right)$	Gradient descent (or gradient ascent if maximising objective): $\lambda^{n+1} = \lambda^n - \eta \Delta\mathcal{L}$ …where η is the step parameter. $\Delta\mathcal{L}_{\text{MLE}}(\lambda, \mathcal{D}) = \sum_{(x,t)\in\mathcal{D}} \mathbb{E}[\phi(x,\cdot)] - \phi(x,t)$ $\Delta\mathcal{L}_{\text{MAP}}(\lambda, \mathcal{D}, \sigma) = \frac{\lambda}{\sigma^2} + \sum_{(x,t)\in\mathcal{D}} \mathbb{E}[\phi(x,\cdot)] - \sum_{(x,t)\in\mathcal{D}} \phi(x,t)$ …where $\sum_{(x,t)\in\mathcal{D}} \phi(x,t)$ are the empirical counts. For each class $\mathcal{C}_k$: $\sum_{(x,t)\in\mathcal{D}} \mathbb{E}[\phi(x,\cdot)] = \sum_{(x,t)\in\mathcal{D}} \phi(x,\cdot) p(\mathcal{C}_k\|x)$	Penalise large values for the λ parameters, by introducing a prior distribution over them (typically a Gaussian). **Objective function** $\mathcal{L}_{\text{MAP}}(\lambda, \mathcal{D}, \sigma) = \arg\min_\lambda \left(-\log p(\lambda) - \sum_{(x,t)\in\mathcal{D}} \log p(t\|x) \right)$ $= \arg\min_\lambda \left(-\log \frac{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(0-\lambda)^2}{2\sigma^2}}}{} - \sum_{(x,t)\in\mathcal{D}} \log p(t\|x) \right)$ $= \arg\min_\lambda \left(\frac{\sum_m \lambda_m^2}{2\sigma^2} - \sum_{(x,t)\in\mathcal{D}} \log p(t\|x) \right)$	$\mathcal{O}(INMK)$, since each training instance must be visited and each combination of class and features must be calculated for the appropriate feature mapping.	Reformulate the class conditional distribution in terms of a kernel $K(x, x')$, and use a non-linear kernel (for example $K(x, x') = (1 + \mathbf{w}^T x)^2$). By the Representer Theorem: $p(\mathcal{C}_k\|x) = \frac{1}{Z_\lambda(x)} e^{\lambda^T \phi(x,\mathcal{C}_k)}$ $= \frac{1}{Z_\lambda(x)} e^{\sum_{n=1}^N \sum_{i=1}^K \alpha_{nk} \phi(x_n, C_i)^T \phi(x,\mathcal{C}_k)}$ $= \frac{1}{Z_\lambda(x)} e^{\sum_{n=1}^N \sum_{i=1}^K \alpha_{nk} K((x_n, C_i), (x,\mathcal{C}_k))}$ $= \frac{1}{Z_\lambda(x)} e^{\sum_{n=1}^N \alpha_{nk} K(x_n, x)}$	Online Gradient Descent: Update the parameters using GD after seeing each training instance.
Perceptron	Directly estimate the linear function $y(x)$ by iteratively updating the weight vector when incorrectly classifying a training instance.	Binary, linear classifier: $y(x) = \text{sign}(\mathbf{w}^T x)$ …where: $\text{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$ Multiclass perceptron: $y(x) = \arg\max_{\mathcal{C}_k} \mathbf{w}^T \phi(x, \mathcal{C}_k)$	Tries to minimise the Error function; the number of incorrectly classified input vectors: $\arg\min_{\mathbf{w}} E_P(\mathbf{w}) = \arg\min_{\mathbf{w}} -\sum_{n\in\mathcal{M}} \mathbf{w}^T x_n t_n$ …where $\mathcal{M}$ is the set of misclassified training vectors.	Iterate over each training example x_n, and update the weight vector if misclassification: $\mathbf{w}^{i+1} = \mathbf{w}^i + \eta \Delta E_P(\mathbf{w})$ $= \mathbf{w}^i + \eta x_n t_n$ …where typically $\eta = 1$. For the multiclass perceptron: $\mathbf{w}^{i+1} = \mathbf{w}^i + \phi(x,t) - \phi(x, y(x))$	The Voted Perceptron: run the perceptron i times and store each iteration's weight vector. Then: $y(x) = \text{sign}\left(\sum_i c_i \times \text{sign}(\mathbf{w}_i^T x) \right)$ …where c_i is the number of correctly classified training instances for $\mathbf{w}_i$.	$\mathcal{O}(INML)$, since each combination of instance, class and features must be calculated (see log-linear).	Use a kernel $K(x, x')$, and 1 weight per training instance: $y(x) = \text{sign}\left(\sum_{n=1}^N w_n t_n K(x, x_n) \right)$ …and the update: $w_n^{i+1} = w_n^i + 1$	The perceptron is an online algorithm per default.
Support vector machines	A maximum margin classifier: finds the separating hyperplane with the maximum margin to its closest data points.	$y(x) = \sum_{n=1}^N \lambda_n t_n x^T x_n + w_0$	**Primal** $\arg\min_{\mathbf{w},w_0} \frac{1}{2}\|\|\mathbf{w}\|\|^2$ s.t. $t_n(\mathbf{w}^T x_n + w_0) \geq 1 \quad \forall n$ **Dual** $\tilde{\mathcal{L}}(\wedge) = \sum_{n=1}^N \lambda_n - \sum_{n=1}^N \sum_{m=1}^N \lambda_n \lambda_m t_n t_m x_n^T x_m$ s.t. $\lambda_n \geq 0, \quad \sum_{n=1}^N \lambda_n t_n = 0, \quad \forall n$	• Quadratic Programming (QP) • SMO, Sequential Minimal Optimisation (chunking).	The soft margin SVM: penalise a hyperplane by the number and distance of misclassified points. **Primal** $\arg\min_{\mathbf{w},w_0} \frac{1}{2}\|\|\mathbf{w}\|\|^2 + C\sum_{n=1}^N \xi_n$ s.t. $t_n(\mathbf{w}^T x_n + w_0) \geq 1 - \xi_n, \quad \xi_n > 0 \quad \forall n$ **Dual** $\tilde{\mathcal{L}}(\wedge) = \sum_{n=1}^N \lambda_n - \sum_{n=1}^N \sum_{m=1}^N \lambda_n \lambda_m t_n t_m x_n^T x_m$ s.t. $0 \leq \lambda_n \leq C, \quad \sum_{n=1}^N \lambda_n t_n = 0, \quad \forall n$	• QP: $\mathcal{O}(n^3)$; • SMO: much more efficient than QP, since computation based only on support vectors.	Use a non-linear kernel $K(x, x')$: $y(x) = \sum_{n=1}^N \lambda_n t_n x^T x_n + w_0$ $= \sum_{n=1}^N \lambda_n t_n K(x, x_n) + w_0$ $\tilde{\mathcal{L}}(\wedge) = \sum_{n=1}^N \lambda_n - \sum_{n=1}^N \sum_{m=1}^N \lambda_n \lambda_m t_n t_m x_n^T x_m$ $= \sum_{n=1}^N \lambda_n - \sum_{n=1}^N \sum_{m=1}^N \lambda_n \lambda_m t_n t_m K(x_n, x_m)$	Online SVM. See, for example: • *The Huller: A Simple and Efficient Online SVM*, Bordes & Bottou (2005) • *Pegasos: Primal Estimated sub-Gradient Solver for SVM*, Shalev-Shwartz et al. (2007)
k-means	A hard-margin, geometric clustering algorithm, where each data point is assigned to its closest centroid.	Hard assignments $r_{nk} \in \{0, 1\}$ s.t. $\forall n \sum_k r_{nk} = 1$, i.e. each data point is assigned to one cluster k. Geometric distance: The Euclidean distance, l^2 norm: $\|\|x_n - \mu_k\|\|_2 = \sqrt{\sum_{i=1}^D (x_{ni} - \mu_{ki})^2}$	$\arg\min_{r,\mu} \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\|x_n - \mu_k\|\|_2^2$ …i.e. minimise the distance from each cluster centre to each of its points.	Expectation: $r_{nk} = \begin{cases} 1 & \text{if } \|\|x_n - \mu_k\|\|^2 \text{ minimal for } k \\ 0 & \text{o/w} \end{cases}$ Maximisation: $\mu_{\text{MLE}}^{(k)} = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$ …where $\mu^{(k)}$ is the centroid of cluster k.	Only hard-margin assignment to clusters.	To be added.	For non-linearly separable data, use kernel k-means as suggested in: *Kernel k-means, Spectral Clustering and Normalized Cuts*, Dhillon et al. (2004).	Sequential k-means: update the centroids after processing one point at a time.
Mixture of Gaussians	A probabilistic clustering algorithm, where clusters are modelled as latent Guassians and each data point is assigned the probability of being drawn from a particular Gaussian.	Assignments to clusters by specifying probabilities $p(z^{(i)}, z^{(i)}) = p(x^{(i)}\|z^{(i)}) p(z^{(i)})$ …with $z^{(i)} \sim$ Multinomial(γ), and $\gamma_{nk} \equiv p(k\|x_n)$ s.t. $\sum_{j=1}^k \gamma_{nj} = 1$. I.e. want to maximise the probability of the observed data $\boldsymbol{x}$.	$\mathcal{L}(\boldsymbol{x}, \pi, \mu, \Sigma) = \log p(\boldsymbol{x}\|\pi, \mu, \Sigma)$ $= \sum_{n=1}^N \log \left(\sum_{k=1}^K \pi_k \mathcal{N}_k(x_n\|\mu_k, \Sigma_k) \right)$	Expectation: For each n, k set: $\gamma_{nk} = p(z^{(i)} = k\|x^{(i)}, \gamma, \mu, \Sigma) \quad (= p(k\|x_n))$ $= \frac{p(x^{(i)}\|z^{(i)} = k; \mu, \Sigma) p(z^{(i)} = k; \pi)}{\sum_{j=1}^K p(x^{(i)}\|z^{(i)} = l; \mu, \Sigma) p(z^{(i)} = l; \pi)}$ $= \frac{\pi_k \mathcal{N}(x_n\|\mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(x_n\|\mu_j, \Sigma_j)}$ Maximisation: $\pi_k = \frac{1}{N} \sum_{n=1}^N \gamma_{nk}$ $\Sigma_k = \frac{\sum_{n=1}^N \gamma_{nk} (x_n - \mu_k)(x_n - \mu_k)^T}{\sum_{n=1}^N \gamma_{nk}}$ $\mu_k = \frac{\sum_{n=1}^N \gamma_{nk} x_n}{\sum_{n=1}^N \gamma_{nk}}$	The mixture of Gaussians assigns probabilities for each cluster to each data point, and as such is capable of capturing ambiguities in the data set.	To be added.	Not applicable.	Online Gaussian Mixture Models. A good start is: *A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants*, Neal & Hinton (1998).

[1] Created by Emanuel Ferm, HT2011, for semi-procrastinational reasons while studying for a Machine Learning exam. Last updated May 5, 2011.

http://bit.ly/efermcheat

soulmachine@gmail.com

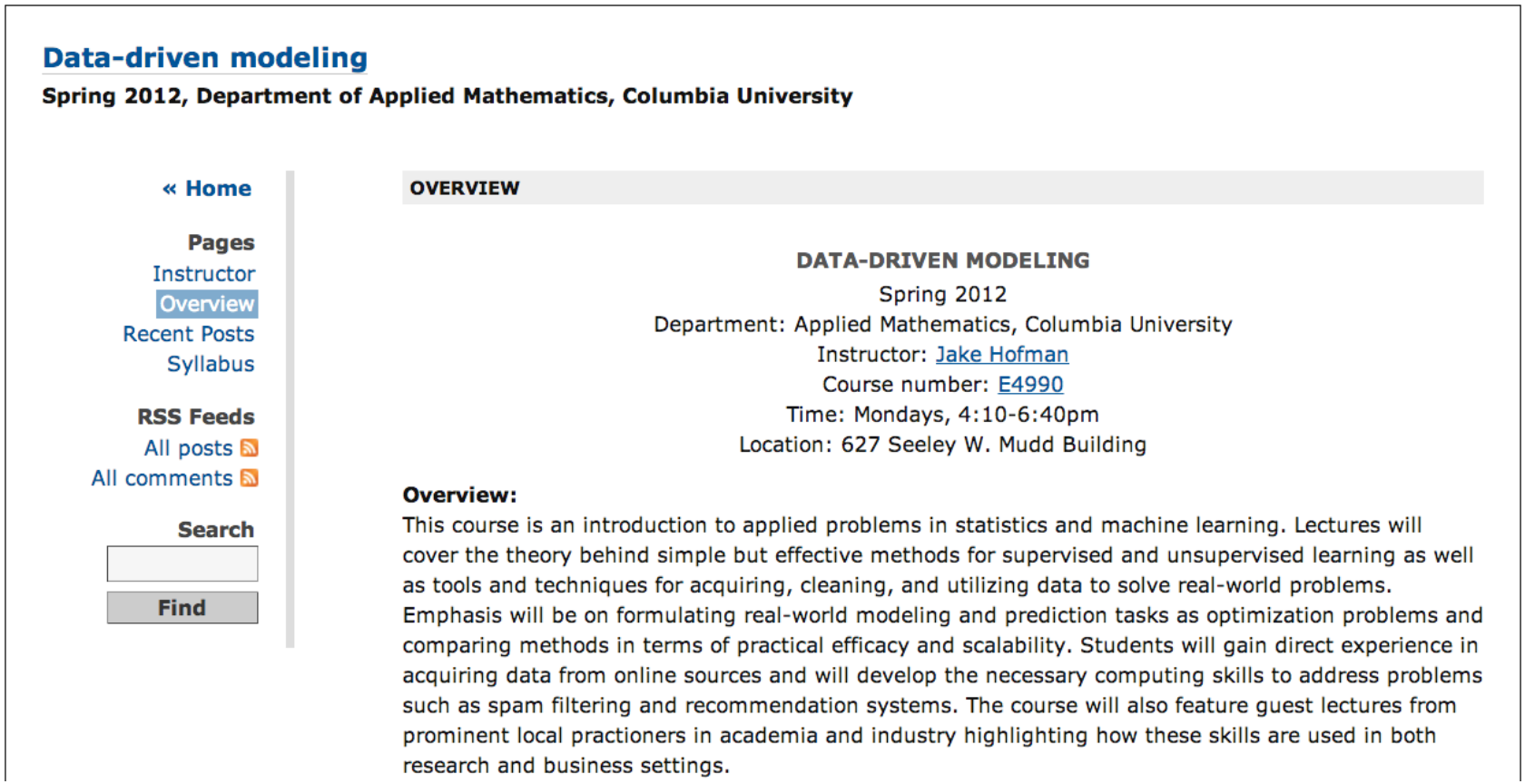
Machine Learning Cheat Sheet

Classical equations, diagrams and tricks in machine learning

February 12, 2015

`http://bit.ly/mlcheatbook`

Applied ML Course



Data-driven modeling

Spring 2012, Department of Applied Mathematics, Columbia University

« Home

Pages
Instructor
Overview
Recent Posts
Syllabus

RSS Feeds
All posts
All comments

Search
Find

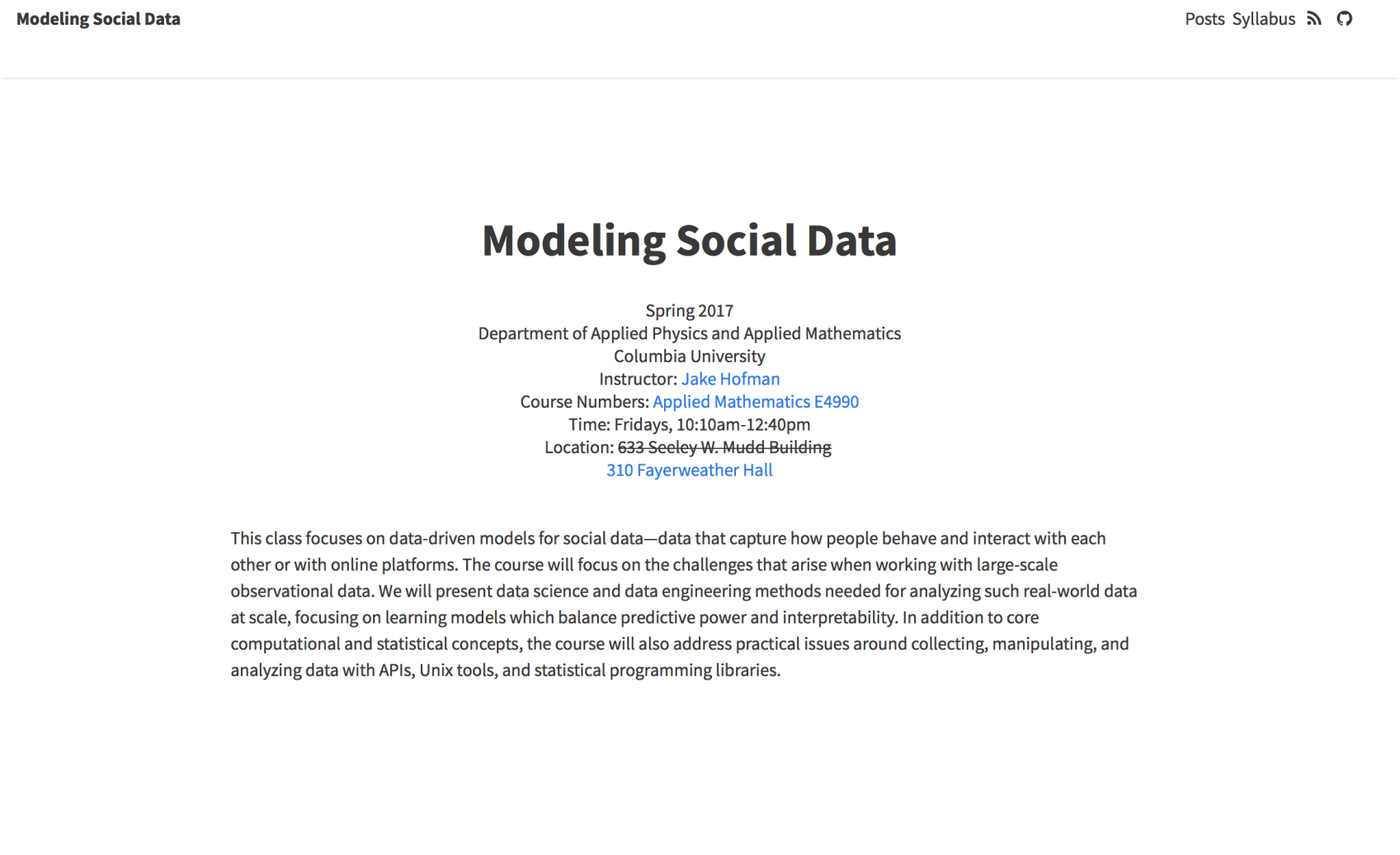
OVERVIEW

DATA-DRIVEN MODELING
Spring 2012
Department: Applied Mathematics, Columbia University
Instructor: Jake Hofman
Course number: E4990
Time: Mondays, 4:10-6:40pm
Location: 627 Seeley W. Mudd Building

Overview:
This course is an introduction to applied problems in statistics and machine learning. Lectures will cover the theory behind simple but effective methods for supervised and unsupervised learning as well as tools and techniques for acquiring, cleaning, and utilizing data to solve real-world problems. Emphasis will be on formulating real-world modeling and prediction tasks as optimization problems and comparing methods in terms of practical efficacy and scalability. Students will gain direct experience in acquiring data from online sources and will develop the necessary computing skills to address problems such as spam filtering and recommendation systems. The course will also feature guest lectures from prominent local practioners in academia and industry highlighting how these skills are used in both research and business settings.

`http://jakehofman.com/ddm`

CSS Course



Modeling Social Data

Posts Syllabus

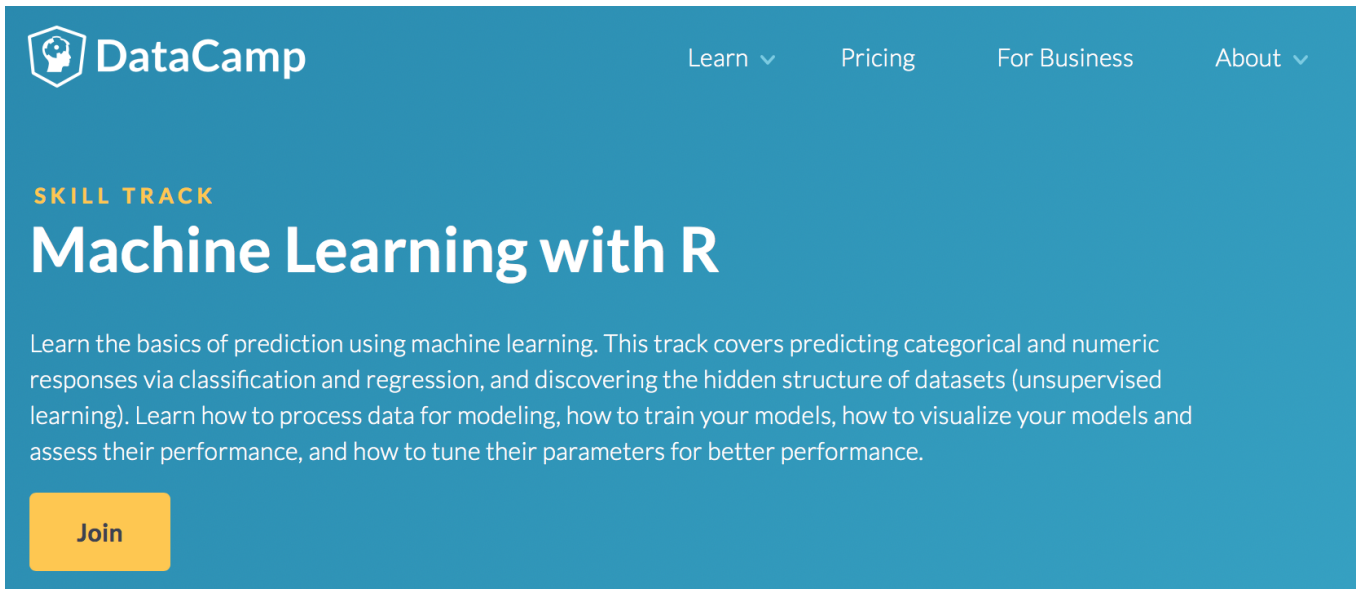
Modeling Social Data

Spring 2017
Department of Applied Physics and Applied Mathematics
Columbia University
Instructor: Jake Hofman
Course Numbers: Applied Mathematics E4990
Time: Fridays, 10:10am-12:40pm
Location: ~~633 Seeley W. Mudd Building~~
310 Fayerweather Hall

This class focuses on data-driven models for social data—data that capture how people behave and interact with each other or with online platforms. The course will focus on the challenges that arise when working with large-scale observational data. We will present data science and data engineering methods needed for analyzing such real-world data at scale, focusing on learning models which balance predictive power and interpretability. In addition to core computational and statistical concepts, the course will also address practical issues around collecting, manipulating, and analyzing data with APIs, Unix tools, and statistical programming libraries.

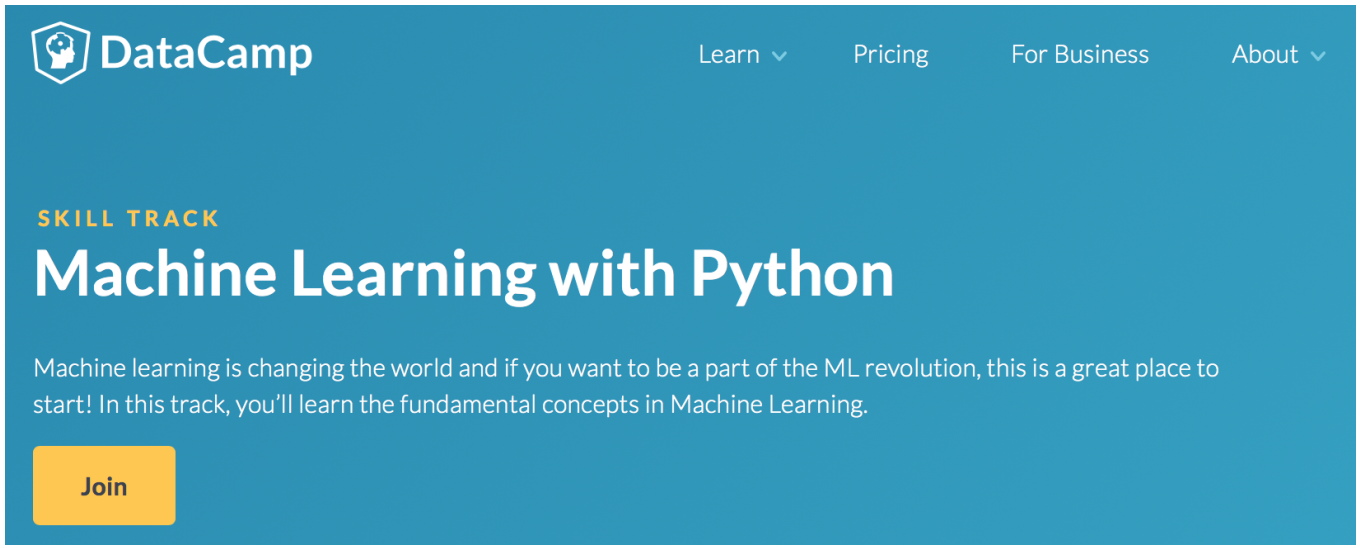
http://modelingsocialdata.org

MOOCs



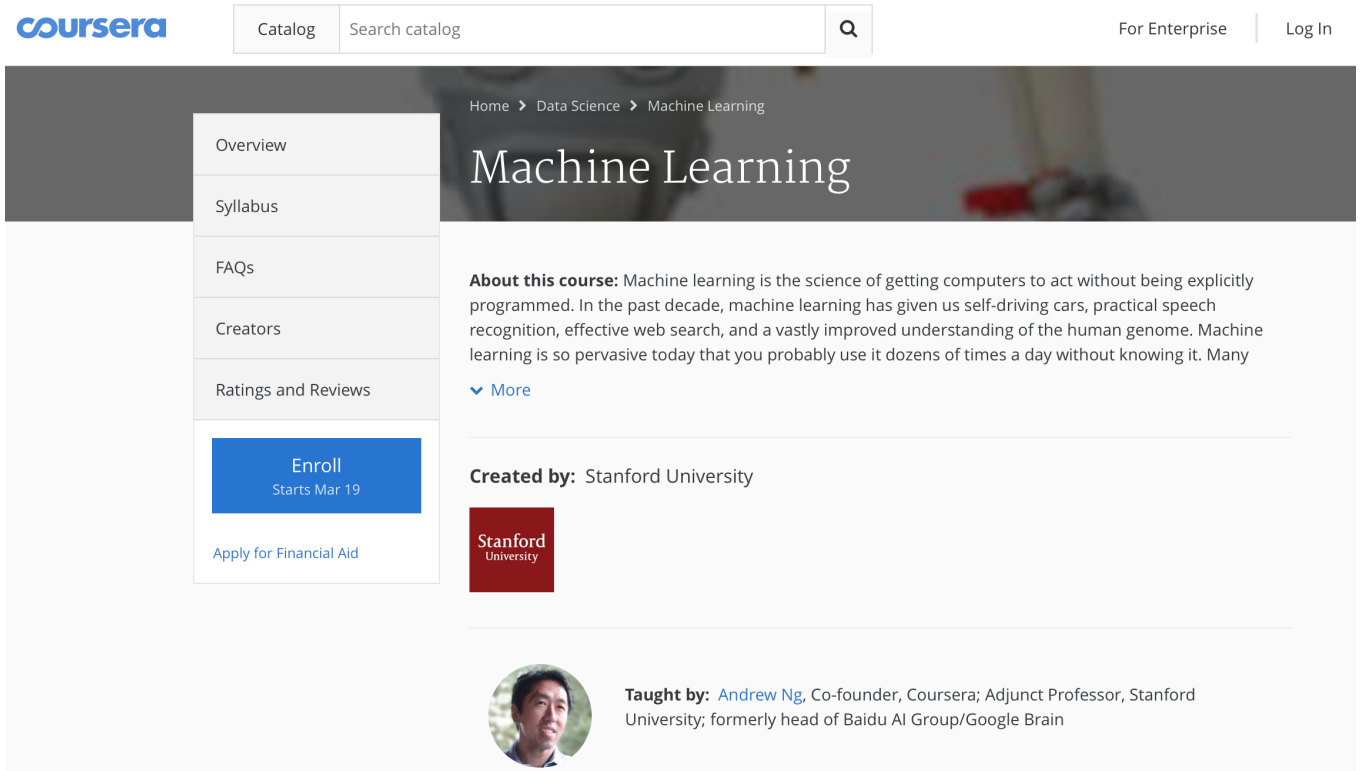
`https://www.datacamp.com/tracks/machine-learning`

MOOCs



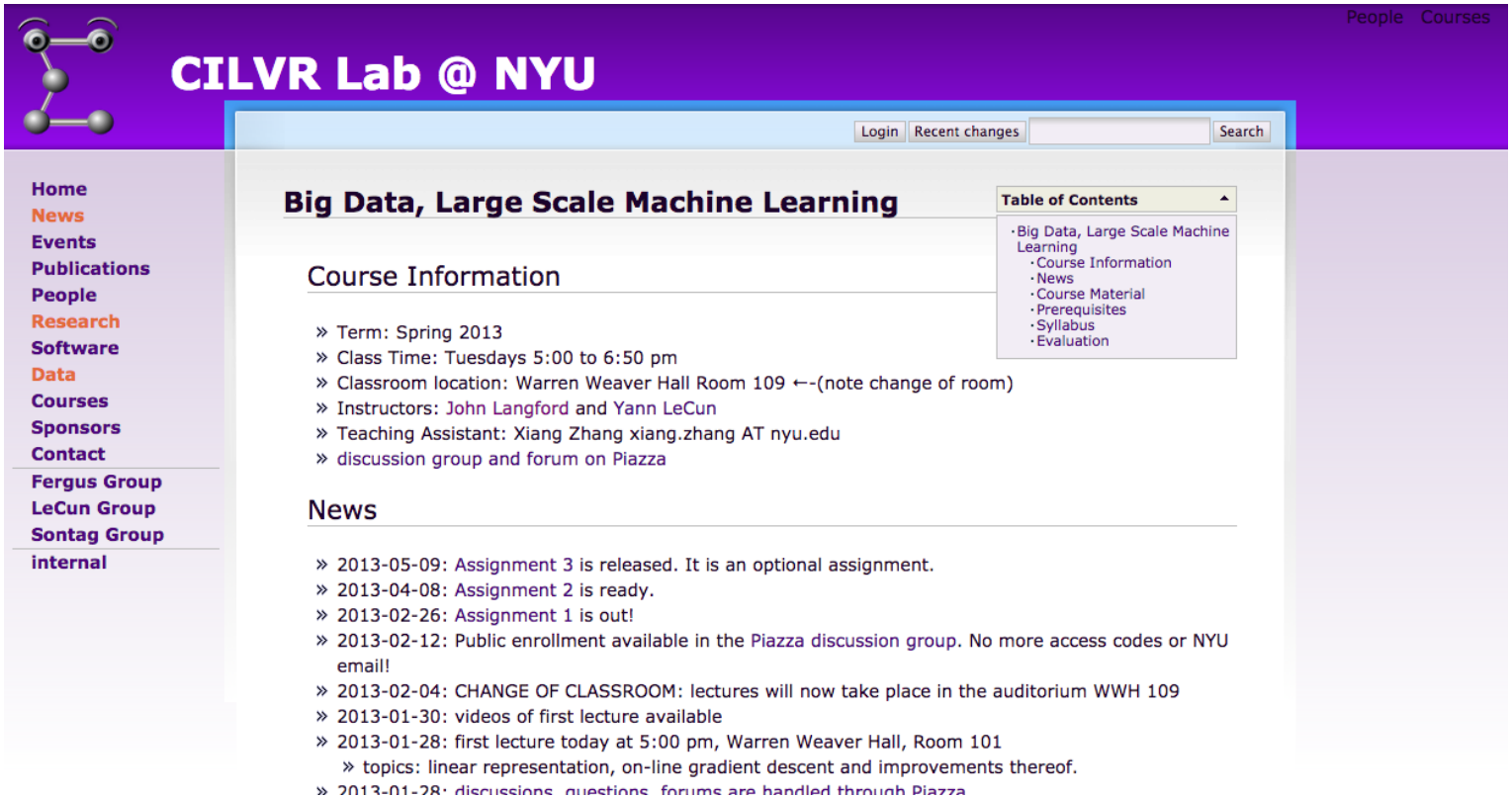
`http://www.datacamp.com/tracks/machine-learning-with-python`

MOOCs



`https://www.coursera.org/learn/machine-learning`

Course++



http://bit.ly/nyubigdata

Course++

Machine Learning the Future

This is class 1pm-2:15pm Mondays in Spring 2017 at Cornell Tech and Cornell about the frontier of machine learning. Each class is lecture and discussion around a chosen topic. I would like each student to leave with an understanding of the topic deep enough to think about the next step.

The class will cover topics broadly related to reinforcement learning including contextual bandits, imitation learning, and exploration as well as other select topics drawn from online learning, optimization, boosting, parallelization, logarithmic prediction, active learning, representation. Logistically, we'll use zoom in addition to a video link.

January 30: cancelled (My voice is not working)
Feburary 6: ML the Future slides, Generalization slides with source, background reading, with a Youtube recording and a backup recording.
February 13: cancelled (travel)
Feburary 20: No class (February break)
Feburary 27: Online Linear Learning slides with source Youtube recording and a backup recording papers: Importance Invariant, Adagrad, Normalized Dataset: RCV1 CCAT-or-not
March 6: Contextual Bandit Eval and Optimization slides with source. A Youtube recording and a backup recording papers: Double robust policies Dataset: RCV1 CCAT-or-not in contextual bandit format (uniform exploration)
March 13: Contextual Bandit Exploration slides with source. A Youtube recording and a backup recording, Papers Epoch Greedy, Cover, Bootstrap Dataset: RCV1 CCAT-or-not in multiclass format
March 20: Nonstationary evaluation slides with source Decision Service slides Contextual Decision Process slides Youtube recording and a backup recording Nonstationary evaluation, Decision Service, and Bellman Rank papers Decision Service site
March 27: Joint prediction as learning to search. Youtube recording and a backup recording DAgger paper AggreVaTe paper
April 3: no class(Spring break)
April 10: Joint prediction results, analysis, programming slides with source. A Youtube recording and a backup recording. LOLS paper Credit Assignment Compiler paper
April 17: Cancelled (healing up)
April 24: Logarithmic time prediction with source Youtube recording and a backup recording covering Error-Correcting Tournaments, Label embedding trees Dynamic Trees I, and Dynamic Trees II. Not covered: Static contextual bandits, Dynamic log time class probability Efficient Multilabel
May 1: Active Learning with source a Youtube recording and a backup recording Iwal papers search papers Cost sensitive active learning
May 8: Parallel Learning with source a Youtube recording and a backup recording allreduce paper distbelief paper 1-bit SGD Ring allreduce

http://hunch.net/~mltf

Thanks. Questions?

jmh@microsoft.com
@jakehofman