CS340 Cheat Sheet
Tristan Rice
2016-12-12

Assignment 1

Decision Trees:
- Classify by walking tree
- Greedy recursive splitting
- Information gain as score, entropy
- O(mnd log n),

Decision Stump Rule Search:
- Ignore rules outside feature ranges.
- Sort examples O(n log n)
- O(n) score updates
- Total cost: O(nd log n), data size: O(nd)

Learning Theory:
- THE TEST DATA CANNOT INFLUENCE THE TRAINING PHASE IN ANY WAY.
- Fundamental Trade-Off:
1. How small you can make the training error.
2. How well training error approximates the test error.
- Simple models have good approximation of test error, but fit training data poorly.
- Cross validation, test on your training data, use each point as test once.
- No free lunch theorem: no best machine learning model for every problem

Naive Bayes:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\, p(y_i = \text{"spam"})}{p(x_i)}$$

- Compute p(y="spam"|xi) > p(y="not spam",xi)
- Need to estimate p(xi|y="spam")
- Runtime: Test: O(kd); Train: O(nkd+n)

Bayes Rule:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Descriptive Statistics:
- Mean, Median, Quantiles (v s.t. x% < v)
- Box Plot

Assignment 2

KNN:
- Non-parametric
- Distance fn: Euclidean, L1 Dist, Jaccard similarity, Cosine, Distance after dimension reduction, Metric learning

Jaccard similarity: D(x1,x2)=(x1 ∩ x2)/(x1 U x2)

Ensemble Methods: Boosting, Averaging

Random Forests:
- Train number of deep decision trees on bootstrapped samples from the main dataset + small random subset of features on each stump
- Very fast, good out of the box classifier
- Bayesian Model Averaging
- Train O(tmnd log n), t = #trees, m = depth, d = #features, can be sqrt(d)
- Classify one is O(tm)

Clustering: No best method, no test error

K-Means:
- k clusters, random center
- add one point and update mean based on assignment
- initialization important, need to know k
- Total cost: O(ndk), updating means O(nd)
- vector quantization: cluster colors
- K-Medians: Can also use L1-norm + median
- K-Means++, init clusters by furthest distance
- Convex clusters

Density based clustering:
- Non-convex clusters

DBSCAN:
- Radius, if at least minPoints of same type become "core"
- Merge clusters if reachable
- Can have outliers
- Sensitive to boundaries
- need lots of points in high dimensions
- finding cluster is expensive
- O(n^2d)

Lp-norms: L1, L2, L-inf

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}.$$

Elbow method: choose k in which the the sharpest elbow (biggest change in slope) for minimum error vs k.

CS340 Cheat Sheet
Tristan Rice
2016-12-12

Assignment 3

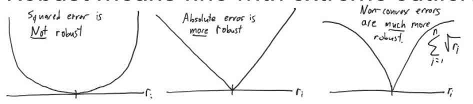
Linear regression (adding bias, change of basis, RBFs, regularization, cost of training/testing, effect of basis parameters).
- 1D $f(w) = \sum_{i=1}^{n} (wx_i - y_i)^2$
  - Minimize least squares $w = \frac{\sum_{i=1}^{n} y_i x_i}{\sum_{i=1}^{n} x_i^2}$
  - Solution
- Bias variable, add row with 1s
- Change of basis, linear least squares
  - each row: [1 x x^2 ... x^p]; [1 x sin(6x)]
- Radial basis functions: non-parametric

$$\begin{bmatrix} g(\|x_1 - x_1\|) & g(\|x_1 - x_2\|) & \dots & g(\|x_1 - x_n\|) \\ g(\|x_2 - x_1\|) & g(\|x_2 - x_2\|) & \dots & g(\|x_2 - x_n\|) \\ \vdots & \vdots & \ddots & \vdots \\ g(\|x_n - x_1\|) & g(\|x_n - x_2\|) & \dots & g(\|x_n - x_n\|) \end{bmatrix}$$

$$g(\alpha) = \exp(-\frac{\alpha^2}{2\sigma^2})$$

- Can combine both

Robust regression (weighted least squares, smooth approximations, computing gradients).
- Robust means fine with extreme outliers


Squared error is Not robust
r_i
Absolute error is more robust
r_i
Non-convex errors are much more robust.
$\sum_{i=1}^{n} \sqrt{r_i}$
r_i

- Absolute is most robust convex error function
- Weighted least squares: weight on each training example

Convexity:
– 1-variable, twice-differentiable function is convex iff f”(w) ≥ 0 for all ‘w’.
– A convex function multiplied by non-negative constant is convex.
– Norms and squared norms are convex.
– The sum of convex functions is a convex function.
– The max of convex functions is a convex function.
– Composition of a convex function and a linear function is convex.
- Not true: composition of convex and convex is convex

0-1 Loss: Number of classification errors
- Hinge: Convex, max{0, 1-y_iw^Tx_i}
  - SVM is hinge loss + L2-regularizer
- Logistic: smooth approximation to hinge, differentiable

Vectors/matrices/norms (summation and vector/matrix/norm notation, minimizing quadratic functions as linear systems)

Assignment 4:

Convex and MLE/MAP estimation (showing functions are convex, connection between probabilities and losses/regularizers)
- likelihood function: p(y|X,w)
- minimize negative log-likelihood

$$f(w) = -\sum_{i=1}^{n} \log(p(y_i|x_i, w))$$

Maximum a Posteriori (MAP) Estimation:
- $p(w|X, y) \propto p(y|X, w)p(w)$
- p(w|X,y) "posterior"
- p(y|X,w) "likelihood"
- p(w) "prior"

$$\boxed{f(w) = -\log(p(w|X, y)) = -\sum_{i=1}^{n} \log(p(y_i|x_i, w)) - \sum_{j=1}^{d} \log(p(w_j)) + const.}$$

- Loss = NLL
- Regularizer = negative log-prior

Sigmoid: h(z) = 1/(1+exp(-z))
Logistic Regression:

$$p(y_i|x_i, w) = \frac{1}{1+\exp(-y_i w^T x_i)}$$

$$f(w) = \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i)) + \frac{\lambda}{2}\|w\|^2$$

Regularizers:
- "robust" = less influenced by large values
- L0 sparsity, non-convex
- L1 makes things more sparse, non-convex
  - Huber loss:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$$

- L2
- L Infinite = max, log-sum-exp is smooth approx
  - LSE(x1,...,xn) = log(exp(x1)+...+exp(xn))
- Frobenius Norm: sqrt(each element squared)

Regularized logistic regression (loss functions for binary classification, effect of different regularizers on overfitting and sparsity)

CS340 Cheat Sheet
Tristan Rice
2016-12-12

Multi-class Logistic (one-vs-all, softmax loss derivatives and implementation)
- multi-class yi is row of d [+1/-1]
- multi-class classification, exactly one +1, encode as class k
- can take max of yi to get correct classification

$$\text{softmax/multinomial logistic regression}$$
$$p(y_i|x_i,w) = \frac{\exp(w_{y_i}^T x_i)}{\sum_{c=1}^k \exp(w_c^T x_i)}$$

- logistic regerssion for binary labels
- softmax for multi-class
- ordinal logistic regression, thresholds of sigmoid as params

Assignment 5:
Principle Compontent Analysis
PCA (computing 1st PC, scaling issue, PCA for visualization, PCA for compression)
- ||w||_2 = 1
- X=ZW
- f(Z,W) = sum_{i=1}^n(w*z_i-x_i)^2
- reduce dimensionality

$$f(W,Z) = \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^d (w_j^T z_i - x_{ij})^2 = \frac{1}{2}\|ZW - X\|_F^2$$

- Solve via:
  - Singular value decomposition (SVD) non-iterative
  - Alternate between updating W and Z
  - Stochastic gradient / Gradient Descent
- X: n by d, Z: n by k, W: k x d

Beyond PCA (different loss functions, effect of regularizers)
- PCA minimized approximation error
- PCA maximizes variance

$$var(X) = \frac{1}{n}\|X\|_F^2$$

- Regularizers can split into different components, sparsity
- Non-negative matrix factorization
  - Regularizes, negative can't cancel out large values

Multi-dimensional scaling (basic model, ISOMAP and geodesic distance)
- Directly optimize the location of zi values
- Gradient descent
- non-convex, sensitive to initialization

$$f(Z) = \sum_{i=1}^n \sum_{j=i+1}^n d_3(d_2(z_i,z_j) - d_1(x_i,x_j))$$

ISOMAP: Geodesic distance
- Use neighbors to compute edge weights
- Use dijkstra's to find shortest path distances

t-SNE: wayyy better, but ugly hack
- focus on small distances
- allow large variance in large distances
- not always better than PCA, can "twist" the plane

Assignment 6:
PageRank (random walk model)
- Probability of landing on page as t->inf
- Add small probability of going to random webpage
- At each step follow random link on page

Stochastic gradient and neural networks
- effect of step-size: too large = can't converge, too small = can't get to the solution
- effect of standardization: helps typically
- effect of initialization: random weights work better with non-convex losses
- effect of non-linearity: makes it a universal approximator
- effect of regularization: higher training error, better estimate of test error
- Dropout: randomly set some xi, zi = 0
- Convolutional
- Vanishing gradient, ReLU (hinge loss)

CS340 Cheat Sheet
Tristan Rice
2016-12-12

Inner product
$a^Tb = b^Ta$
$a^T(b+c) = a^Tb + a^Tc$
Orthogonal $a^Tb = 0$ Orthonormol $a^Tb = 1$
Matrix Multiplication
$A(BC) = (AB)C$
$A(B+C) = AB + AC$
$AB \neq BC$
$(AB)^T = B^TA^T$
$(AB)^2 = ABAB$
Matrix $*$ vector $=$ vector
$x^TAy = x^T(Ay) = (Ay)^Tx = y^TA^Tx$
Inverses
$A^{-1}A = I = AA^{-1}$
$(A^{-1})^T = (A^T)^{-1}$
$\gamma A)^{-1} = \gamma^{-1}A^{-1}$
$(AB)^{-1} = B^{-1}A^{-1}$

Assignment 1:
Summary statistics (like range, median, and quantiles).
Data visualization (histogram, scatterplot, and box plot).
Decision trees (how to fit stumps using classification error, how to classify a new example, runtime in O() notation, effect of depth).
Learning theory (training vs. test error, validation sets and cross-validation, fundamental trade-off)
Naive Bayes (conditional probability, fitting the probabilities, classifying a new example, runtime).

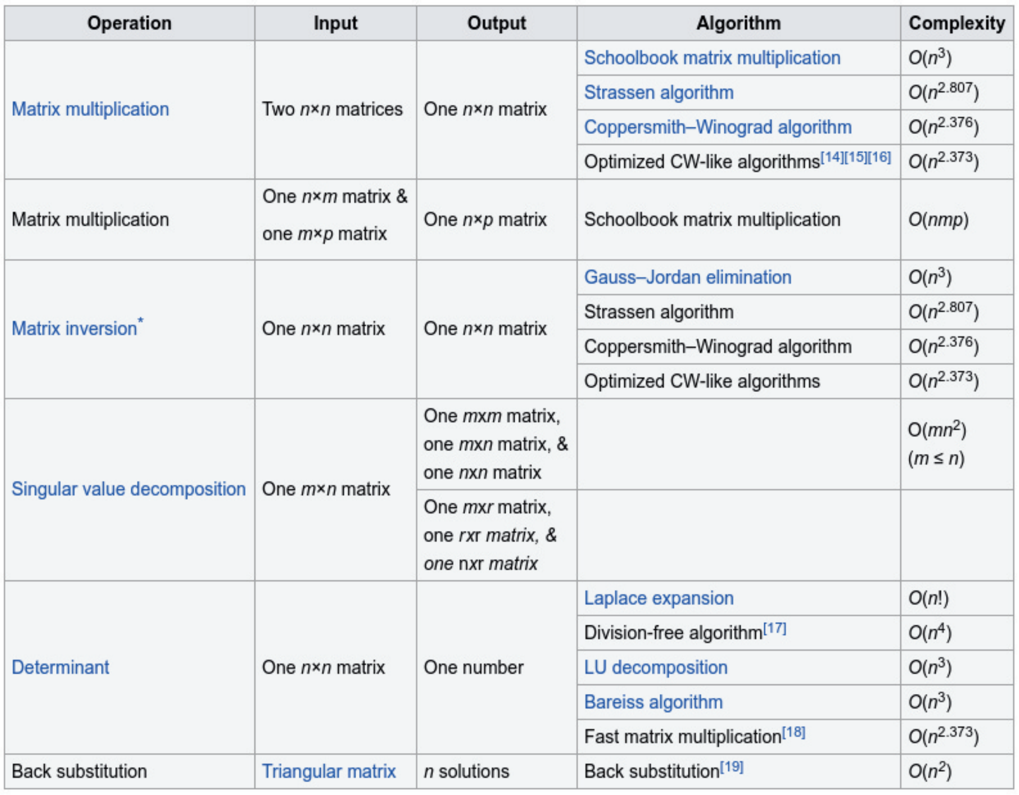
Assignment 2:
K-nearest neighbours (how to classify a new example, runtime in O() notation, effect of k, condensed version).
Random forests (what bootstrap does, how random trees works, effect of number of trees/random-features, how to classify new example).
K-Means (effect of initialization, error functions, runtime, how to cluster new example, elbow method, k-medians, vector quantization).
Density-based clustering (effect of parameters, shape of clusters).

Assignment 3:
Vectors/matrices/norms (summation and vector/matrix/norm notation, minimizing quadratic functions as linear systems)
Linear regression (adding bias, change of basis, RBFs, regularization, cost of training/testing, effect of basis parameters).
Robust regression (weighted least squares, smooth approximations, computing gradients).

Assignment 4:
Regularized logistic regression (loss functions for binary classification, effect of different regularizers on overfitting and sparsity)
Convex and MLE/MAP estimation (showing functions are convex, connection between probabilities and losses/regularizers)
Multi-class Logistic (one-vs-all, softmax loss derivatives and implementation)

Assignment 5:
PCA (computing 1st PC, scaling issue, PCA for visualization, PCA for compression)
Beyond PCA (different loss functions, effect of regularizers)
Multi-dimensional scaling (basic model, ISOMAP and geodesic distance)

Assignment 6:
PageRank (random walk model)
Stochastic gradient and neural networks (effect of step-size, effect of standardization, effect of initialization, effect of non-linearity, effect of regularization)



Operation	Input	Output	Algorithm	Complexity
Matrix multiplication	Two $n \times n$ matrices	One $n \times n$ matrix	Schoolbook matrix multiplication	$O(n^3)$
			Strassen algorithm	$O(n^{2.807})$
			Coppersmith–Winograd algorithm	$O(n^{2.376})$
			Optimized CW-like algorithms[14][15][16]	$O(n^{2.373})$
Matrix multiplication	One $n \times m$ matrix & one $m \times p$ matrix	One $n \times p$ matrix	Schoolbook matrix multiplication	$O(nmp)$
Matrix inversion*	One $n \times n$ matrix	One $n \times n$ matrix	Gauss–Jordan elimination	$O(n^3)$
			Strassen algorithm	$O(n^{2.807})$
			Coppersmith–Winograd algorithm	$O(n^{2.376})$
			Optimized CW-like algorithms	$O(n^{2.373})$
Singular value decomposition	One $m \times n$ matrix	One mxm matrix, one mxn matrix, & one nxn matrix		$O(mn^2)$ $(m \le n)$
		One mxr matrix, one rxr matrix, & one nxr matrix		
Determinant	One $n \times n$ matrix	One number	Laplace expansion	$O(n!)$
			Division-free algorithm[17]	$O(n^4)$
			LU decomposition	$O(n^3)$
			Bareiss algorithm	$O(n^3)$
			Fast matrix multiplication[18]	$O(n^{2.373})$
Back substitution	Triangular matrix	n solutions	Back substitution[19]	$O(n^2)$