Machine Learning

Basic Info

- Andrew Ng — lecturer
- mixed — level
- 7/15/2020 — Date
- By Chaolan Xu

Week 1 Intro

Introduction

1. what is Machine Learning/(un)supervised
   - 1.1 E,T,P /study gives computer ability to learn without explicitly programmed
   - 1.2 content (supervised, unsupervised, reinforcement learning, recommender system...

Model Representation

1. cost function
2. gradient descent
   - 2.1 intuition
   - 2.2 learning rate alpha (small, large and fixed)
   - 2.3 algorithm (update thetas simultaneously)

Linear Algebra Review

1. scala, sigular matrix, etc

Week 2 Linear Regression

Multivariate Linear Regression

1. cost function, gradient descent
   - 1.1 how to make sure gd is working correctly (declare convergence if cost function decreases by less than 0.001 in one iteration)
2. feature scaling
   - 2.1 about -3 < xi < 3
   - 2.2 mean normalization
3. polynomial regression
   - 3.1 create new features by yourself (x2x3, x2^2...) (pay attention to feature scaling)

Normal Equation (another way to solve for theta)

1. theta = (X'X)-1X'y
2. no need to feature scaling
3. compare with gradient descent
   - 3.1 no need alpha/iteration; slow when n is large
4. noninvertbility
   - 4.1 the 'pinv' function will give you a value of \thetaθ even if X^TX is not invertible.
   - 4.2 consider if there are redundant features
   - 4.3 may be too many features---delete some features or use regularization

Octave/Matlab

Week 3 Logistic Regression

Classification

1. hypothesis
   - 1.1 g(z)/interpretation of hypothesis output
2. decision boundary
   - 2.1 g(z) >= 0.5, z needs to > 0, i.e. transpose of theta * x > 0

Logistic Regression Model

1. cost function and its intuition/gradient descent
2. advanced optimization algorithms (Conjugate Gradient, BFGS, L-BFGS)

Multiclass Classification

1. one-vs-all method

Solving the problem of overfitting

1. reduce number of features
   - 1.1 manually select which features to keep
   - 1.2 model selection algo
2. regularization (reduce magnitude of theta)
   - 2.1 regularize linear regression
     - 2.1.1 gradient descent
     - 2.1.2 normal equation
   - 2.2 regularize logistic regression
     - 2.2.1 gradient descent
     - 2.2.2 advanced optimization

Week 4 Neural Networks 1

Motivation

1. 1. when there are lots of features
2. 2. car recognition (computer vision)
3. 3. back ground: neurons and brain

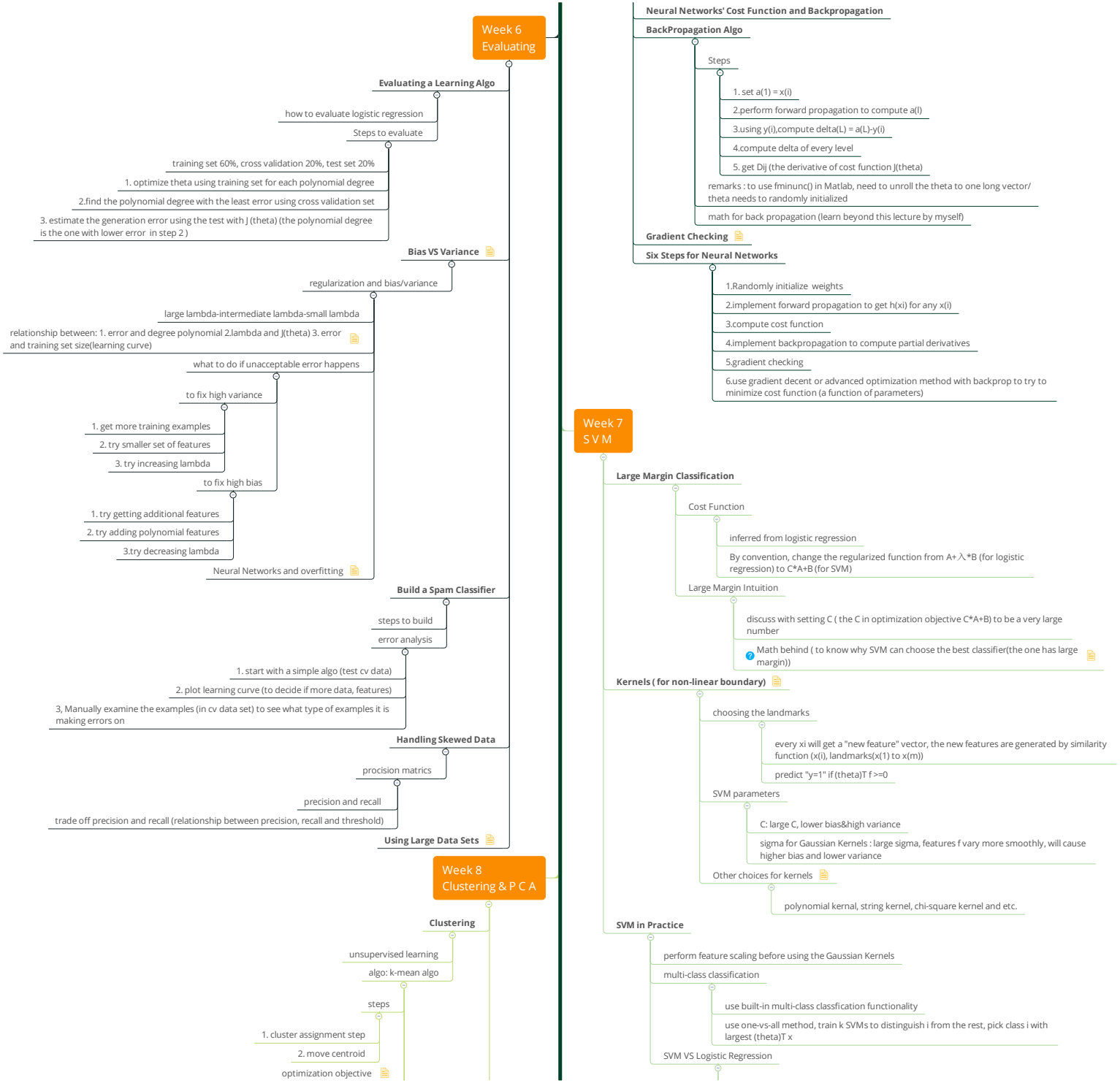
Neural Networks

1. Model Representation

Applications

1. x1 XOR(XNOR) X2
2. XI AND(OR) X2 (have example)
3. (NOT x1) AND (NOT x2) (=1 if and only if x1=0=x2)
4. Multiclass Classification (with y having more columns)

Week 5 Neural Networks 2


Week 6
Evaluating
Evaluating a Learning Algo
how to evaluate logistic regression
Steps to evaluate
training set 60%, cross validation 20%, test set 20%
1. optimize theta using training set for each polynomial degree
2.find the polynomial degree with the least error using cross validation set
3. estimate the generation error using the test with J (theta) (the polynomial degree is the one with lower error in step 2)
Bias VS Variance
regularization and bias/variance
large lambda-intermediate lambda-small lambda
relationship between: 1. error and degree polynomial 2.lambda and J(theta) 3. error and training set size(learning curve)
what to do if unacceptable error happens
to fix high variance
1. get more training examples
2. try smaller set of features
3. try increasing lambda
to fix high bias
1. try getting additional features
2. try adding polynomial features
3.try decreasing lambda
Neural Networks and overfitting
Build a Spam Classifier
steps to build
error analysis
1. start with a simple algo (test cv data)
2. plot learning curve (to decide if more data, features)
3, Manually examine the examples (in cv data set) to see what type of examples it is making errors on
Handling Skewed Data
procision matrics
precision and recall
trade off precision and recall (relationship between precision, recall and threshold)
Using Large Data Sets
Week 8
Clustering & P C A
Clustering
unsupervised learning
algo: k-mean algo
steps
1. cluster assignment step
2. move centroid
optimization objective
Neural Networks' Cost Function and Backpropagation
BackPropagation Algo
Steps
1. set a(1) = x(i)
2.perform forward propagation to compute a(l)
3.using y(i),compute delta(L) = a(L)-y(i)
4.compute delta of every level
5. get Dij (the derivative of cost function J(theta)
remarks : to use fminunc() in Matlab, need to unroll the theta to one long vector/ theta needs to randomly initialized
math for back propagation (learn beyond this lecture by myself)
Gradient Checking
Six Steps for Neural Networks
1.Randomly initialize weights
2.implement forward propagation to get h(xi) for any x(i)
3.compute cost function
4.implement backpropagation to compute partial derivatives
5.gradient checking
6.use gradient decent or advanced optimization method with backprop to try to minimize cost function (a function of parameters)
Week 7
S V M
Large Margin Classification
Cost Function
inferred from logistic regression
By convention, change the regularized function from A+入*B (for logistic regression) to C*A+B (for SVM)
Large Margin Intuition
discuss with setting C (the C in optimization objective C*A+B) to be a very large number
Math behind (to know why SVM can choose the best classifier(the one has large margin))
Kernels (for non-linear boundary)
choosing the landmarks
every xi will get a "new feature" vector, the new features are generated by similarity function (x(i), landmarks(x(1) to x(m))
predict "y=1" if (theta)T f >=0
SVM parameters
C: large C, lower bias&high variance
sigma for Gaussian Kernels : large sigma, features f vary more smoothly, will cause higher bias and lower variance
Other choices for kernels
polynomial kernal, string kernel, chi-square kernel and etc.
SVM in Practice
perform feature scaling before using the Gaussian Kernels
multi-class classification
use built-in multi-class classfication functionality
use one-vs-all method, train k SVMs to distinguish i from the rest, pick class i with largest (theta)T x
SVM VS Logistic Regression

- random initialization
  - randomly pick k training examples, set u1,...,uk equal to these k examples
  - need to randomly initiailize couple of times to avoid stucking in local optima
- how to choose the number of clusters
  - elbow method (relationship between J and k)
  - considering the purpose

PCA

- motivation
  - 1. Data Compression/Dimensionality Reduction
  - 2. Data Visualization
- PCA algo
  - 1. compute " covariance matrix" of x(i) --sigma
  - 2. compute "eigenvectors" of matrix sigma ([U,S,V] = svd(sigma))
  - 3. "compressed data" z = u(reduce)T*x (from n dimension to k dimension , U (reduce) here is n *k matrix)
- applying PCA
  - reconstruction from compressed representation (x(approximate) = U(reduce)*z
  - choose the number of principle components (k)
    - PCA tries to minimize average squared projection error a (about | |xi-x(appro| |**)
    - b is the total variation
    - k is the smallest value to make a/b <= 0.01 (in this case, we say 99% of variance is retained)
    - algo to choose k
- advice for applying PCA
  - supervised learning speedup (from data (x,y) to new training set (x,y)), the PCA is run on training set but the mapping can apply to CV data set and test set
  - application: compression & visualization
  - bad use of PCA: to prevent overfitting, instead, we can use regularization to prevent
  - before implementing PCA, first try whatever you want to do with the original data

Week 10
Large Scare Machine Learning

Gradient Descent with Large Datasets

- batch gradient descent (use m examples for every iteration of theta)
- stochastic gradient descent
  - 1. randomly reorder m training examples
  - 2. repeat couple of times (1 to 10) to get the theta (not to sum up m examples like batch gradient descent but use the example one by one to gradually optimize theta) (use only one example in every iteration)
- minibatch gradient descent
  - use b (minibatch gradient descent size) examples (sum up) in each iteration
  - is likely to outperform stochastic gradient descent only you can good vectorized implementation
  - it allows you to partially paralleize your computation over the b examples
- stochastic gradient descent convergence
  - to check: during learning, compute cost(x(i), y(i)) before updating theta using (x(i), y(i)); every 1000 iterations, plot cost(x(i), y(i)) averaged over the last 1000 examples processed by algo; learning rate a can be adjusted (a = const number1/(const 2+iteration number)
- online learning algorithm (has a continued stream of data)
  - use examples to learn and then discard the examples, use the new coming examples

- use SVM
  - n is small and m is intermediate (like n 1-1000, m 10-10000)
- use Logistic R
  - when n is large (relative to m) (equal to SVM without kernels)
  - n is small, m is large(like n 1-1000 and m 50000+), create more features, then SVM without a kernel or LR'

Week 9
Anomaly Detection&Predicting Rating (Collaborative Filtering)

Anomaly Detection

- one dimension Gaussian Distribution
  - algo steps
    - 1. choosing features xi that you think might be indicative of anomalous examples
    - 2. fit parameters (use data set to find the mean and sigma for every feature)
    - 3. given new example x, compute p(x), anomaly if p(x) < epsilon (mean and sigma use the ones fitted from data set)
  - build an anomaly detection system
    - 1. fit model on training set
    - 2. on cross validation/test examples, predict normal(1) or anomaly(0) based on p(x) < or >= epsilon (CV set can also be used to find epsilon)
    - 3. possible evaluation matrix (classfication accuracy is not a good indicator--why? -- skewed data, always predicting 0 will have high classification accuracy)
  - anomaly detection VS supervised learning
    - use anomaly detection when:
      - many normal examples
      - many different types of anomalies, hard for any algo to learn from positive examples (y=1) what the anomallies may look like
    - use supervised learning when:
      - large number of positive and negative examples
      - enough positive examples (y=1, i.e. anomalies) for algo to get a sense of what positive examples likely to be similar to ones in training set
  - choosing what feature to use
    - 1. make data more likes Gaussian Distri. e.g. use log()
    - 2.most common problem: p(x) is comparable for normal and anomalous examples------> create new feature to capture the anomalies
- multivate Gaussian Distribution
  - motivation and intuition
  - anomaly detection using the multivariate Gaussian Distribution
    - steps
      - 1. fit model with data set
      - 2. give a new example, compute p(x) ---flag anomaly if p(x) < epsilon
    - relationship to original model
      - 1. the original model is the special case of multivariate model (with a special sigma matrix)
      - 2. difference
        - original: need to manually create features to capture anomalies where x1, x2 take unusual combinations of values ; ok even if m is small; computationally cheaper
        - multivariate: automatically captures correlations between features; computationally more expensive; must have m>n(otherwise, sigma matrix will be non-inversable) (if sigma matrix is non-inversable, check redundant features)

Predicting Movie Ratings

- another example : to predict the click-through rate (product search)

Map-reduce and data parallelism

- split training set into different subset, assume using 4 machine to run in parallel (each machine takes 1/4 data) and finally combine the four results to speed up
- can be applicable on single computer with multi-core

- Problem formulation
  - theta is the weight for features of a movie, x is the "magtitude" of each feature , y is the rating.
  - algo (give x to learn theta) cost function, gradient decent
- collaborative filtering
  - given theta to learn x, given x to learn theta
  - algo
    - (1) optimization objective (minimize x and theta simultaneously)
    - (2) initialize x and theta to small random values -- minimize cost function using gradient decent or an advanced optimization algo -- for a user with new x and theta, predict rating y
- low rank matrix (X*transpose of theta) factorization algo
  - recommend related product
    - 1. for each product i, we learn a feature vector x(i)
    - 2. small ||x(i) - x(j)|| ---> movie j and i are similar
    - implementation details (mean normalization) (use an example in which character Eve did not rate for any movie and we want to predict her rating)

Week 11
Application Examples

- photo optical character recognition (OCR)
  - problem description and pipeline
    - image-->text detection-->character segmentation-->character recognition
  - sliding windows
  - getting lots of data and artificial data
    - can download a character with different font
    - add distortions to characters
  - what part of the pipeline should you spend the most time trying to improve?
    - Ceiling analysis (estimate the errors due to each component)