# Machine Learning Practice and Theory

## Day 5 - Supervised Learning - Logistic Regression

Govind Gopakumar

IIT Kanpur

# Prelude

# Announcements

- New project groups : Meet after class for short discussion
- Old project groups : Meeting tomorrow
- Programming tutorials to be put up tonight / tomorrow
- Webpage - govg.github.io/acass

# Recap

## Our first Regression model

- How to fit a line through our model
- How is this formed?
- Analytical solution for 1D
- Problems with this for more than 1D

## Matrix factorization as regression

- Reduction of "complicated" problem to simple problems.
- "Random" method to optimize - alternating optimization
- Works for non-convex loss functions

# Probabilistic Classification

# Logistic Regression - I

**Why do we need this?**

- Wish to predict "probability" of a label
- Useful to quantify "confidence" about prediction

**Idea from linear regression**

- $\langle w, x \rangle$ : Similarity between parameter and point
- How do we extend this to classification?
- Very simple model : sum of all features!

# Logistic Regression - II

## Model overview

- Learn a parameter $w$
- $p(y_i = 1) = \mu_i$
- $\mu_i = \frac{1}{1+\exp(-w^T x_i)}$
- Computes a "score" : $\langle w, x \rangle$
- Squashes it between (0,1)

## Interpretation?

- Very high "scores" - ?
- Very low "scores" - ?
- When are we not "confident"?

# Logistic Regression - III

## Learning

- We need to find out this $w$ parameter.
- What does the decision rule look like?
- $\log \frac{p(y_i=1)}{p(y_i=0)} = ?$
- Intuitve explanation of this?

## Geometry of the solution

- Still learning a line!
- How does this differ from other "lines"?
- Why is this useful then?

# Logistic Regression - IV

**Learning the parameter**

- Can we come up with a loss function?
- Why will this be easy or hard?
- How can we optimize this?

**Problems with the squared loss**

- Can we differentiate this easily?
- Is this convex?

# Logistic Regression - V

## Constructing a loss

- How do we choose a loss?
- Loss should be high when predicted and actual are different.
- Loss should be low when predicted is same as actual.

## Two way loss

- If $y_i = 1$, loss $l(w) = -\log(\mu_i)$
- If $y_i = 0$, loss $l(w) = -\log(1 - \mu_i)$
- Why does this seem right?

# Logistic Regression - VI

**Final cross-entropy loss**

- $l(w) = -y_i \log(\mu_i) - (1 - y_i) \log(1 - \mu_i)$
- "Cross" entropy : related to earlier entropy
- How do we write this in terms of $w$?

**Loss function**

- Setting $\mu_i = \frac{\exp(w^T x_i)}{1+\exp(w^T x_i)}$
- $L(w) = -\sum(y_i w^T x_i - log(1 + \exp(w^T x_i)))$
- How do we impose control on solution?

# Logistic Regression - VII

## Optimizing this loss

- $L(w) = -\sum(y_i w^T x_i - log(1 + \exp(w^T x_i)))$
- $g = -\sum \left( y_i x_i - \frac{\exp(w^T x_i)}{1+\exp(w^T x_i)} \right)$
- Is there a simple form? Yes!

## Final expression

- $g = -\sum(y_i - \mu_i)x_i$
- Can we set it to zero?
- What do we do now?

# Logistic Regression - VIII

## Gradient descent

- Update using $w^{t+1} = w^t - \eta g_t$
- $w^{t+1} = w^t - \eta \sum(\mu_i^t - y_i)x_i$

## Analyzing the update step

- What $x_i$ is added to $w^t$ more?
- Does this sort of update make sense now?
- How much time do we require to compute this?

# Gradient Descent - I

## Improving gradient descent

- Choice of $\eta$ is crucial!
- Can add a momentum term $w^{t+1} = w^t - \eta g_t + \alpha^t(w^t - w^{t-1})$
- Can also use "second-order" methods (beyond the scope of this class)

## Speeding up gradient descent

- We need to compute gradient across entire data
- Is there a naive solution to this?

# Gradient Descent - II

## Mini-batch Gradient Descent

- Approximate the loss function using a subset
- Gradient becomes faster to compute
- Why should this work?

## Stochastic Gradient Descent

- Let's take it to the extreme - use just one point!
- Extremely fast gradient descent
- Why would this work at all?

# Logistic Regression - via Probability - I

## Choosing a likelihood

- What is appropriate?
- Can we relate this to something we know?
- How do we write down entire likelihood?

## Doing "Maximum" probability

- $p(y_i) = \mu_i^{y_i}(1-\mu_i)^{1-y_i}$
- What will we get? Any guesses?

# Logistic Regression - IX

## Multiclass

- Naturally extend this to multiclass - how?
- Can think of it both in loss function sense and probability sense
- Same methods will apply, with some tweaks

## Comments

- Probability estimate of class, instead of decision
- Gradient descent can be done fast
- Widely used, in different fields as well
- Used as modules in neural networks!

# Yet another Classifier

# Perceptron - I

## Extending the Logistic model

- $w^{t+1} = w^t - \eta_t(\mu_i^t - y_i)x_i$
- Replace with a cutoff for $\mu_i$
- $w^{t+1} = w^t - \eta_t(\hat{y}_i - y_i)x_i$

## Analyzing the new update

- When does this update actually take place?
- What is this update when it does take place?
- For ease, let us assume labels $y_i \in \{-1, 1\}$.

# Perceptron - II

## Mistake driven learning

- Update upon mistake : $w^{t+1} = w^t + 2\eta_t y_i x_i$
- What does this update look like?
- Why does the update work?

## Geometry of the classifier

- What will the loss surface be?
- Learns a linear surface!
- Why is it useful then? - Extremely fast way to construct it

# Perceptron - III

## Significance of Perceptrons

- Almost the first ever "classifier" built
- Can be thought of as a model for a brain
- Led to AI "winter" : ML research stalled for a while
- Actual theoretical proof on number of mistakes!

## Usage of perceptrons

- Multilayer Perceptrons : Starting point for neural networks
- Almost every "deep neural network" is an MLP
- Non-linear methods : do a transformation! (when we discuss kernels)

# Halfway round up

# General techniques

## Loss functions

- Why choice of a loss function matters
- Common loss functions : squared loss!
- How some loss functions can be bad.

## Probability method

- Maximize the experiment happening!
- How to choose a likelihood model
- How it (possibly) leads to same answer as above

# Methods discussed

## Classification

- K - nearest neighbors
- Decision Trees
- Random Forests
- Logistic Regression
- Perceptron

## Regression

- Adaptation of KNN
- Adaptation of Decision Tree?
- Linear Regression

# Agenda for next week

**Unsupervised Learning and Advanced methods**

- Cover some unsupervised learning methods
- Cover some "advanced" material (SVM, Neural Networks, Kernels)

**Greater focus on programming**

- Every class will have a programming assignment
- (Hopefully) deal with "realistic" datasets
- Two classes on feature "extraction" and modelling
- One class purely on best practices for experiments

# Conclusion

# Concluding Remarks

## Takeaways

- Another classification technique : Logistic Regression
- Gradient descent and stochastic gradient descent
- The perceptron algorithm

## Announcements

- Extra class : Monday 3 - 4 pm (purely a Python tutorial)
- Quiz 1 : Automatically graded
- Assignment 2 : Working on the MNIST dataset

# References


- Lecture 7, CS 771 IIT Kanpur
- Lecture 6, CS 771 IIT Kanpur