CSE 417T
Introduction to Machine Learning

Lecture 8

Instructor: Chien-Ju (CJ) Ho

Logistics: Homework 1

- Please follow the (long) instructions
  - Make sure your submissions are **readable** (especially for scanned submissions)
  - Double check whether you correctly **specify pages for each problem**
    - You won't get points if they are not correctly specified
- Gradescope
  - Check whether you can get access to Gradescope
  - Reserve time for submissions
    - Generally won't grant extensions for technical reasons

Logistics: Homework 2

- Homework 2 will be announced on Friday (tomorrow)
  - One implementation question (in Matlab)
  - Several math questions

- Planned due date: Feb 24, Monday

- Remember that March 3 is the date of the first exam
  - Considering the two late days, we can talk about HW2 the lecture before the exam if needed

Recap

Linear Models

This is why it's called linear models

- H contains hypothesis $h(\vec{x})$ as **some function of $\vec{w}^T\vec{x}$**

	Domain	Model
Linear Classification	$y \in \{-1, +1\}$	$H = \{h(\vec{x}) = sign(\vec{w}^T\vec{x})\}$
Linear Regression	$y \in \mathbb{R}$	$H = \{h(\vec{x}) = \vec{w}^T\vec{x}\}$
Logistic Regression	$y \in [0,1]$	$H = \{h(\vec{x}) = \theta(\vec{w}^T\vec{x})\}$

$$\theta(s) = \frac{e^s}{1 + e^s}$$

- Algorithm:
  - Focus on $g = argmin_{h \in H} E_{in}(h)$

Linear Classification

- Formulation
  - Hypothesis set $H = \{h(\vec{x}) = sign(\vec{w}^T\vec{x})\}$
  - Error measure: binary error $e(h(\vec{x}), y) = \mathbb{I}[h(\vec{x}) \neq y]$
- Data is linearly separable
  - Run PLA => $E_{in} = 0$ => Low E_{out}
- Data is not linearly separable
  - Engineering the features to make data closer to be separable
  - Pocket algorithm

Linear Regression

- Formulation
  - Hypothesis set $H = \{h(\vec{x}) = \vec{w}^T \vec{x}\}$
  - Squared error $e(h(\vec{x}), y) = (h(\vec{x}) - y)^2$
- Given dataset $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$
  - $E_{in}(\vec{w}) = \frac{1}{N} \sum_{n=1}^{N} (\vec{w}^T \vec{x}_n - y_n)^2$
- Goal: find $\vec{w}_{lin} = argmin_{\vec{w}}\, E_{in}(\vec{w})$

Linear Regression Algorithm

- Given $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$
- Construct $X = \begin{bmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{1,0} & x_{1,1} & \cdots & x_{1,d} \\ x_{2,0} & x_{2,1} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{2,0} & x_{N,1} & \cdots & x_{N,d} \end{bmatrix}$ and $\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$
- Output $\vec{w}_{lin} = (X^T X)^{-1} X^T \vec{y}$ (Assume $X^T X$ is invertible)
- Done

Brief Lecture Notes Today

The notes are not intended to be comprehensive. They should be accompanied by lectures and/or textbook.

Let me know if you spot errors.

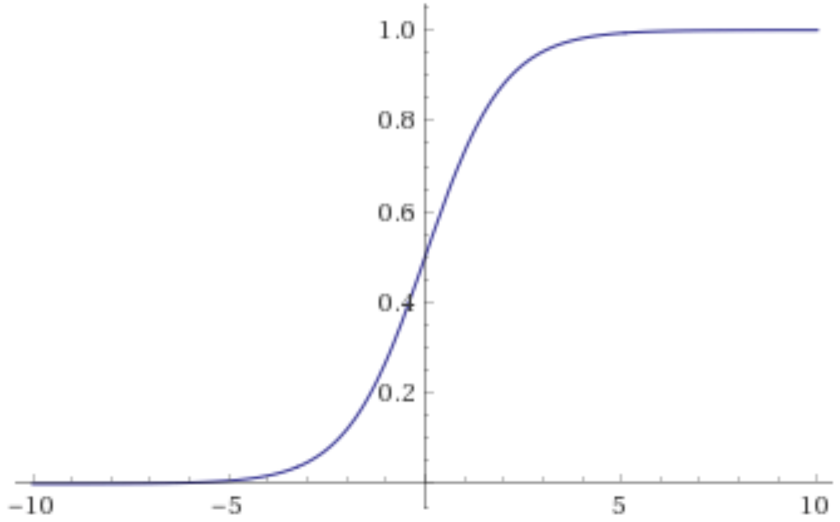
Logistic Regression

Logistic Regression

- Hypothesis set $H = \{h(\vec{x}) = \theta(\vec{w}^T\vec{x})\}$

- $\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$
  - A sigmoid function (”S”-shaped function)
  - $\theta(s) = \begin{cases} 1 & \text{when } s \to \infty \\ 0.5 & \text{when } s = 0 \\ 0 & \text{when } s \to -\infty \end{cases}$

- Useful property
  - $1 - \theta(s) = \frac{1+e^s}{1+e^s} - \frac{e^s}{1+e^s} = \frac{1}{1+e^s} = \theta(-s)$



Logistic Regression: Predicting a Probability

- Will this patient have a heart attack within the next year?

age	62 years
gender	male
blood sugar	120 mg/dL40,000
HDL	50
LDL	120
Mass	190 lbs
Height	$5'\ 10''$
...	...

Classification: Yes/No

Logistic regression: Probability of Yes

- A hypothesis $h(\vec{x})$ output a value in $[0,1]$
  - Interpreting it as the probability of yes

What kind of Dataset do We Get?

- Dataset $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$

- What are the values of y_n?
    - Ideally, we want to have y_n to be the probability value
    - In practice, we cannot measure a probability
    - We can only see the occurrence of an event and infer the probability

- Need to address the case when $y_n \in \{-1, +1\}$ in the given dataset D

How to Quantify $g \approx f$

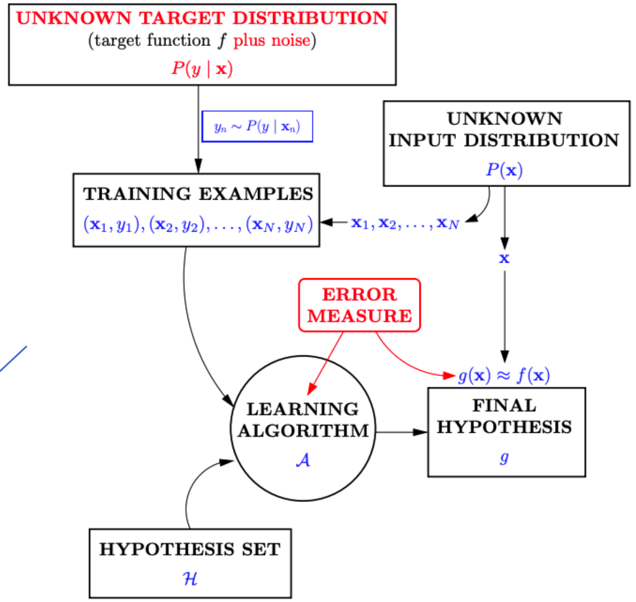
- Target function $f(\vec{x}) = \Pr(y = +1|\vec{x})$
  - $\Pr(y|\vec{x}) = \begin{cases} f(\vec{x}) & \text{for } y = +1 \\ 1 - f(\vec{x}) & \text{for } y = -1 \end{cases}$
- How do define the error measure to quantify $g \approx f$
  - Ideally, we want it to be meaningful
    - Binary error for classification: tell us the number of mistakes we make
    - Squared error for regression: the error minimizer is the "mean (average)"
  - We also want it to be easy to optimize

Cross Entropy Error

$$E_{in}(\vec{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln(1 + e^{-y_n \vec{w}^T \vec{x}_n})$$

- It looks complicated, but
  - It has nice interpretations (min error => max likelihood)
  - It is easy to optimize (continuous, differentiable, convex)

Likelihood



- How are $D = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N)\}$ generated?
  - $(\vec{x}_1, \ldots, \vec{x}_N)$ are i.i.d. drawn from a distribution
  - $(y_1, \ldots, y_N)$ are labeled according to target function $f(\vec{x})$

- Likelihood $\Pr(D|h)$
  - The probability of seeing dataset D if we believe y is generated according to h
  - $\Pr(D|h) = \Pr((\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N)|h)$

$$= \Pr(\vec{x}_1, \ldots, \vec{x}_N) \Pr\big((y_1, \ldots, y_N)|(\vec{x}_1, \ldots, \vec{x}_N), \mathrm{h}\big)$$

$$= \prod_{n=1}^{N} \Pr(\vec{x}_n) \prod_{n=1}^{N} \Pr(y_n|\vec{x}_n, h)$$

Assumption of independent data

Maximum Likelihood

- Choosing the hypothesis that maximizes the likelihood
  - $g = argmax_{h \in H} \Pr(D|h)$
    $= argmax_{h \in H} \prod_{n=1}^{N} \Pr(\vec{x}_n) \ \prod_{n=1}^{N} \Pr(y_n|\vec{x}_n, h)$
    $= argmax_{h \in H} \prod_{n=1}^{N} \Pr(y_n|\vec{x}_n, h)$

$\prod_{n=1}^{N} \Pr(\vec{x}_n)$ doesn't depend on h

- We interpret $h(\vec{x})$ as the probability
  - $\Pr(y|\vec{x}, h) = \begin{cases} h(\vec{x}) = \theta(\vec{w}^T \vec{x}) & \text{for } y = +1 \\ 1 - h(\vec{x}) = 1 - \theta(\vec{w}^T \vec{x}) & \text{for } y = -1 \end{cases}$

- Since $1 - \theta(s) = \theta(-s)$, we have $\Pr(y|\vec{x}, h) = \theta(y\, \vec{w}^T \vec{x})$

Maximum Likelihood

- Choosing the hypothesis that maximizes the likelihood
  - $g = argmax_{h \in H} \Pr(D|h)$
    $= argmax_{h \in H} \prod_{n=1}^{N} \Pr(y_n | \vec{x}_n, h)$

  - $\vec{w}^* = argmax_{\vec{w}} \prod_{n=1}^{N} \theta(y_n \vec{w}^T \vec{x}_n)$
    $= argmax_{\vec{w}} \ln(\prod_{n=1}^{N} \theta(y_n \vec{w}^T \vec{x}_n)\ )$
    $= argmax_{\vec{w}} \sum_{n=1}^{N} \ln(\theta(y_n \vec{w}^T \vec{x}_n))$
    $= argmin_{\vec{w}} - \sum_{n=1}^{N} \ln(\theta(y_n \vec{w}^T \vec{x}_n))$
    $= argmin_{\vec{w}} \sum_{n=1}^{N} \ln \frac{1}{\theta(y_n \vec{w}^T \vec{x}_n)}$
    $= argmin_{\vec{w}} \sum_{n=1}^{N} \ln \frac{1}{\theta(y_n \vec{w}^T \vec{x}_n)}$
    $= argmin_{\vec{w}} \boxed{\sum_{n=1}^{N} \ln(1 + e^{-y_n \vec{w}^T \vec{x}_n})}$

$$E_{in}(\vec{w}) = \frac{1}{N} \sum_{n=1}^{N} \ln(1 + e^{-y_n \vec{w}^T \vec{x}_n})$$

Cross Entropy Error

$$E_{in}(\vec{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln(1 + e^{-y_n \vec{w}^T \vec{x}_n})$$

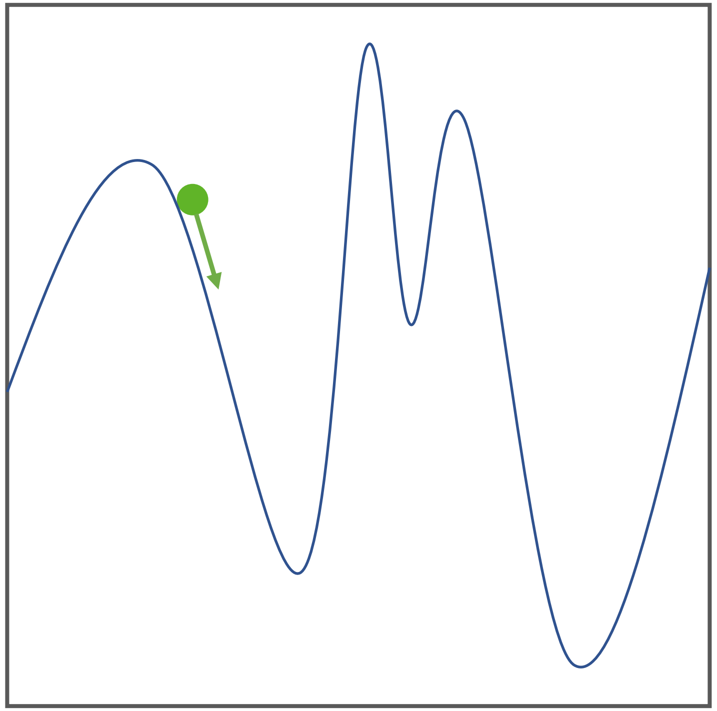
- Minimizing $E_{in}(\vec{w})$ is the same as maximizing likelihood
- Next question: How to solve $\vec{w}^* = argmin_{\vec{w}} E_{in}(\vec{w})$
  - Answer: Solve for $\nabla_{\vec{w}} E_{in}(\vec{w}) = 0$
  - No single-step solution like we have in linear regression

Gradient Descent

A general optimization technique

Gradient Descent

- An technique for optimizing functions that gradients exist everywhere.



- An iterative method that converges to local optimum.
- Converge to global optimum if the function is convex (since there is only one local optimum).

Gradient Descent: Optimizing $E_{in}(\vec{w})$

- An iterative method of the form:

$$\vec{w}(t+1) \leftarrow \vec{w}(t) + \eta_t \vec{v}_t$$

- $\vec{v}_t$: an unit vector, determining the direction of the update
- η_t: an scalar, determining how much to update

- How to choose $\vec{v}_t$ and η_t?

Choosing $\vec{v}_t$ in $\vec{w}(t+1) \leftarrow \vec{w}(t) + \eta_t \vec{v}_t$

- Choose $\vec{v}_t$ that moves towards the "steepest" direction
  - Approaching the minimum faster

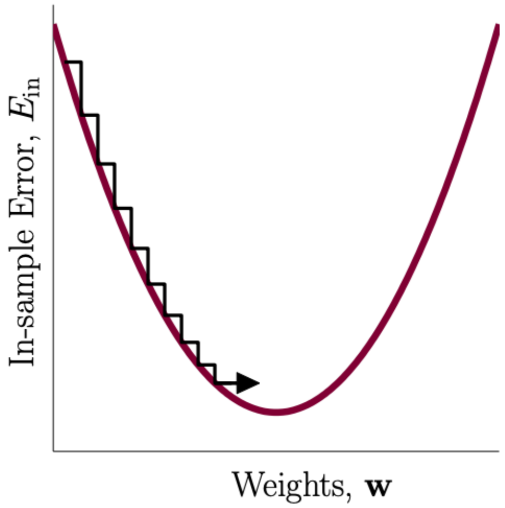
η_t is usually small, so ignoring this term

- Taylor's approximation:
  - $E_{in}(\vec{w}(t) + \eta_t \vec{v}_t) = E_{in}(\vec{w}(t)) + \eta_t \nabla_{\vec{w}} E_{in}(\vec{w}(t))^T \vec{v}_t + O(\eta_t^2)$
  - $E_{in}(\vec{w}(t+1)) - E_{in}(\vec{w}(t)) \approx \eta_t \nabla_{\vec{w}} E_{in}(\vec{w}(t))^T \vec{v}_t$

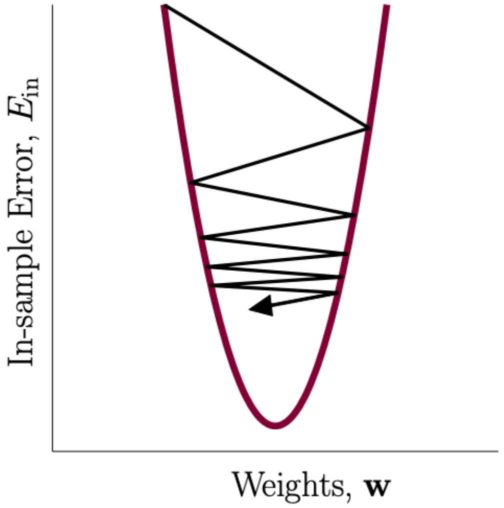
- Choose unit vector $\vec{v}_t$ that minimizes $\nabla_{\vec{w}} E_{in}(\vec{w}(t))^T \vec{v}_t$
  - $\vec{v}_t$ should be in the opposite direction of $\nabla_{\vec{w}} E_{in}(\vec{w}(t))$
  - $\vec{v}_t = \frac{-\nabla_{\vec{w}} E_{in}(\vec{w}(t))}{\|\nabla_{\vec{w}} E_{in}(\vec{w}(t))\|}$

Choosing η_t in $\vec{w}(t+1) \leftarrow \vec{w}(t) + \eta_t \vec{v}_t$

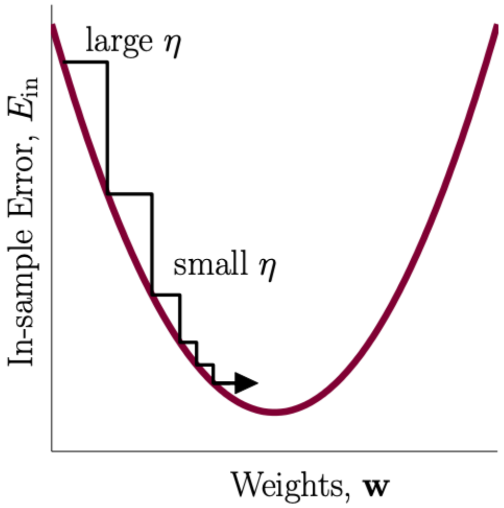
η too small



η too large



variable η_t – just right



Choosing η_t in $\vec{w}(t+1) \leftarrow \vec{w}(t) + \eta_t \vec{v}_t$

- Intuition (for convex E_{in})
  - When E_{in} is closer to the minimum,
    - $\nabla_{\vec{w}} E_{in}\left(\vec{w}(t)\right)$ is smaller
    - We should set η_t smaller

- Therefore, set $\eta_t = \eta \|\nabla_{\vec{w}} E_{in}(\vec{w}(t))\|$

Putting Them Together

- Iterative update rule: $\vec{w}(t+1) \leftarrow \vec{w}(t) + \eta_t \vec{v}_t$
  - $\vec{v}_t = \frac{-\nabla_{\vec{w}} E_{in}(\vec{w}(t))}{\|\nabla_{\vec{w}} E_{in}(\vec{w}(t))\|}$
  - $\eta_t = \eta \|\nabla_{\vec{w}} E_{in}(\vec{w}(t))\|$

- Gradient calculations
  - $E_{in}(\vec{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln(1 + e^{-y_n \vec{w}^T \vec{x}_n})$
  - $\nabla_{\vec{w}} E_{in}(\vec{w}) = \frac{1}{N}\sum_{n=1}^{N} \frac{-y_n \vec{x}\, e^{-y_n \vec{w}^T \vec{x}_n}}{1+e^{-y_n \vec{w}^T \vec{x}_n}} = -\frac{1}{N}\sum_{n=1}^{N} \frac{y_n \vec{x}_n}{1+e^{y_n \vec{w}^T \vec{x}_n}}$

Gradient Descent for Logistic Regression

- Initialize $\vec{w}(0)$
- For $t = 0, \ldots$
  - Compute $\nabla_{\vec{w}} E_{in}(\vec{w}(t)) = -\frac{1}{N}\sum_{n=1}^{N} \frac{y_n \vec{x}_n}{1+e^{y_n \vec{w}(t)^T \vec{x}_n}}$
  - $\vec{w}(t+1) \leftarrow \vec{w}(t) - \eta \nabla_{\vec{w}} E_{in}\left(\vec{w}(t)\right)$
  - Terminate if the stop conditions are met
- Return the final weights

η: learning rate.
A parameter the learner can choose.

Gradient Descent for Logistic Regression

- Initialization
  - Random initialization is a good idea and common approach
- Stop conditions (a mix of the following criteria)
  - When the number of iteration exceeds the pre-set threshold
  - When the improvement on E_{in} (e.g., check $\nabla_{\vec{w}} E_{in}$) is too small
  - When E_{in} is small enough