Outline

• Week 3 Recap
• The Linear Model
  – Linear Classification
  – Linear Regression
  – Logistic Regression
  – Nonlinear Transformation
• Theory of the Learning Problem
  – The Feasibility of the Learning Problem
  – Error and Noise
  – Theory of Generalization
  – VC Dimension
  – Training and Testing
  – Bias-Variance Tradeoff
Outline

• Week 3 Recap
• Linear Model
  – Linear Classification
  – Linear Regression
  – Logistic Regression
  – Nonlinear Transformation
• Theory of the Learning Problem
  – The Feasibility of the Learning Problem
  – Error and Noise
  – Theory of Generalization
  – VC Dimension
  – Training and Testing
  – Bias-Variance Tradeoff
Recap – Learning Styles

- In terms of inputs and outputs of data

Supervisory | Unsupervisory | Semi-supervisory | Reinforcement
Recap – PLA and Pocket Algorithm

- PLA: $h(x) = \text{sign}(w^T x)$, $w_{t+1} = w_t + y_n(t)x_n(t)$
- Pocket Algorithm: keep the best one in the pocket
Outline

• Week 3 Recap
• Linear Model
  – Linear Classification
  – Linear Regression
  – Logistic Regression
  – Nonlinear Transformation
• Theory of the Learning Problem
  – The Feasibility of the Learning Problem
  – Theory of Generalization
  – VC Dimension
  – Error and Noise
  – Training and Testing
  – Bias-Variance Tradeoff
The Overview of The Linear Model

• Linear Model
  – Classification
  – Regression
  – **Probability Estimate**: Logistic Regression

• The Linear Model should be tried first in general as the ML solution
Linear Classification

• Classification Problem
  – PLA is a binary classifier: \( y(x) \in [+1, -1] \)
  – Output will take two values

• What if an output has > 2 values (multi-class)?
• Basic idea:
  – Transform multi-class into binary class
Example - Handwritten Digital Recognition

- Each digit is represented by a 16x16 pixel image
- Output is one of 10 values
Example - Cont.

- Original Problem:
  - Raw inputs: \((x_0, x_1, x_2, \ldots, x_{256})\)
  - Linear model: \((w_0, w_1, w_2, \ldots, w_{256})\)

- Transformed Problem:
  - Main features: Intensity and Symmetry
  - Inputs: \((x_0, x_1, x_2)\)
  - Linear model: \((w_0, w_1, w_2)\)
Example - Cont.

- Inputs: \((x_0, x_1, x_2)\)
- \(x_1\): Intensity
- \(x_2\): Symmetry
Example - Results

PLA:

Pocket:
Outline

• Week 3 Recap
• Linear Model
  – Linear Classification
  – Linear Regression
  – Logistic Regression
  – Nonlinear Transformation
• Theory of the Learning Problem
  – The Feasibility of the Learning Problem
  – Theory of Generalization
  – VC Dimension
  – Error and Noise
  – Training and Testing
  – Bias-Variance Tradeoff
Outline Linear Regression

• Math Concepts and Theories Involved
• ML Algorithm in theory
• ML algorithm in practice
Math Concepts Involved

• Linear Algebra
  – Matrix Transpose
  – Vector Norm
  – Matrix Inverse
  – Pseudo-Inverse

• Calculus
  – Partial Differentiation

• Probability and Statistics
  – Squared Errors
Linear Regression in Statistics

- In statistics, *linear regression* is a popular method to modeling the relationship between a dependent variable and one or more independent variables.

\[ y = ax + b \]

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>2.00</td>
<td>2.00</td>
</tr>
<tr>
<td>3.00</td>
<td>1.30</td>
</tr>
<tr>
<td>4.00</td>
<td>3.75</td>
</tr>
<tr>
<td>5.00</td>
<td>2.25</td>
</tr>
</tbody>
</table>
• What a hypothesis $h$ looks like for linear regression?
Credit Problem Revisit

- Input $x$

<table>
<thead>
<tr>
<th>Age</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
<td>Male</td>
</tr>
<tr>
<td>Annual Income</td>
<td>$95,000</td>
</tr>
<tr>
<td>Year in Residence</td>
<td>2</td>
</tr>
<tr>
<td>Year in Job</td>
<td>3</td>
</tr>
<tr>
<td>Outstanding Debt</td>
<td>$120,000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Classification: Credit Approval (Yes/No)
- Regression: Credit Line: (dollar amount)

- Output $h(x)$
  - Classification: $h(x) = \text{sign}(w^T x)$
  - Regression: $h(x) = \sum_{i=0}^{d} w_i x_i = w^T x$
The Data Set

• Credit officiers decide on credit lines

\[ (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N) \]

• where \( y_n \in \mathbb{R} \) is the credit line for a customer \( x_n \)

• Linear regression tries to replicate it
How To Measure The Error

• How well does \( h(x) \) approximates \( f(x) \)?
• In linear regression, we used squared error \((h(x) - f(x))^2\)

\[
E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} (h(x_n) - y_n)^2
\]
Geometric View of Linear Regression
Anatomy of Error $E$

\[ E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} (w^T x_n - y_n)^2 \]

\[ = \frac{1}{N} \|Xw - y\|^2 \]

where

\[ X = \begin{bmatrix}
  x_1^T \\
  x_2^T \\
  \vdots \\
  x_N^T \\
\end{bmatrix}, \quad y = \begin{bmatrix}
  y_1 \\
  y_2 \\
  \vdots \\
  y_N \\
\end{bmatrix} \]
Minimization of Error $E$

$$E_{in}(w) = \frac{1}{N} \|Xw - y\|^2$$

$$\nabla E_{in}(w) = \frac{2}{N} X^\top (Xw - y) = 0$$

$$X^\top Xw = X^\top y$$

$$w = X^\dagger y \text{ where } X^\dagger = (X^\top X)^{-1} X^\top$$

$X^\dagger$ is the ‘pseudo-inverse’ of $X$
The Pseudo-Inverse

\[
X^\dagger = (X^\top X)^{-1}X^\top
\]
The Linear Regression Algorithm

1. Construct the matrix $X$ and the vector $y$ from the data set $(x_1, y_1), \cdots, (x_N, y_N)$ as follows

$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}. $$

2. Compute the pseudo-inverse $X^\dagger = (X^T X)^{-1} X^T$.

3. Return $w = X^\dagger y$. 

Linear Regression and Classification

- Linear regression learns a real-valued function $y = f(x) \in \mathbb{R}$
- Binary-valued functions are also real-valued! $\pm 1 \in \mathbb{R}$
- Use linear regression to get $w$ where $w^T x_n \approx y_n = \pm 1$
- In this case, $\text{sign}(w^T x_n)$ is likely to agree with $y_n = \pm 1$
- It can be used for good initial weights for classification
Linear Regression in Action

• Dataset
  – The Diabetes Dataset from UCI Machine Learning Repo

• Python Code
  – Will be posted on the CCLE class website

• Result
Outline

• Week 3 Recap
• Linear Model
  – Linear Classification
  – Linear Regression
  – Logistic Regression
  – Nonlinear Transformation
• Theory of the Learning Problem
  – The Feasibility of the Learning Problem
  – Theory of Generalization
  – VC Dimension
  – Error and Noise
  – Training and Testing
  – Bias-Variance Tradeoff
Outline of Logistic Regression

• Math Concepts and Theories Involved
• Logistic regression in theory
  – The model
  – Error measure
  – Learning algorithm
• Logistic regression in practice
Math Concepts Involved

• Linear Algebra
  – Matrix Transpose
  – Vector Norm

• Calculus
  – Partial Differentiation
  – Gradient

• Probability and Statistics
  – Conditional Probability
  – Logistic/Sigmoid Function
Logistic Regression in ML

- What a hypothesis $h$ looks like for Logistic regression?
The Linear Model Overview

\[ s = \sum_{i=0}^{d} w_i x_i \]

**Linear Classification**
\[ h(\mathbf{x}) = \text{sign}(s) \]

**Linear Regression**
\[ h(\mathbf{x}) = s \]

**Logistic Regression**
\[ h(\mathbf{x}) = \theta(s) \]
Example: Prediction of Heart Attacks

• **Input** $\mathbf{x}$: $(x_1, x_2, \ldots, x_d)$
  – Cholesterol level, age, weight, gender, ...

• **Output** $\mathbf{y}$: $(y_1, y_2, \ldots, y_d)$
  – PLA: Yes/No
  – Linear regression: Risk Score ($s = \mathbf{w}^\top \mathbf{x}$)
Basic Idea of Logistic Regression

- **Input** $\mathbf{x}$: $(x_1, x_2, \ldots, x_d)$
  - Cholesterol level, age, weight, gender, ...
- **Output** $\mathbf{y}$: $(y_1, y_2, \ldots, y_d)$
- Logistic regression:
  - Probability of a heart attack
  - $\theta(s): X \rightarrow [0, 1]$

Logistic Regression

$h(\mathbf{x}) = \theta(s)$
The Logistic Function $\theta$

- The Formula

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

- aka *soft threshold*, or the *sigmoid* function
The Problem Formulation

- The target function to be learned
  \[ f(x) = P[y = +1 | x] \]

- Data is generated by a noisy target \( P[y | x] \)

\[
P(y | x) = \begin{cases} 
  f(x) & \text{for } y = +1; \\
  1 - f(x) & \text{for } y = -1.
\end{cases}
\]

The target function \( f : \mathbb{R}^d \rightarrow [0, 1] \) is the probability

Learn \( g(x) = \theta(w^T x) \approx f(x) \)
Error Measure - Likelihood

- For each \((x, y)\), \(y\) is generated by probability \(f(x)\)
- The standard error measure is based on likelihood
- if \(h = f\), how likely it is to get \(y\) from \(x\)?

\[
P(y \mid x) = \begin{cases} 
  h(x) & \text{for } y = +1; \\
  1 - h(x) & \text{for } y = -1.
\end{cases}
\]
\[ P(y \mid x) = \begin{cases} h(x) & \text{for } y = +1; \\ 1 - h(x) & \text{for } y = -1. \end{cases} \]

Substitute \( h(x) = \theta(w^T x) \), noting \( \theta(-s) = 1 - \theta(s) \)

\[ P(y \mid x) = \theta(y \, w^T x) \]

Likelihood of \( \mathcal{D} = (x_1, y_1), \ldots, (x_N, y_N) \) is

\[ \prod_{n=1}^{N} P(y_n \mid x_n) = \prod_{n=1}^{N} \theta(y_n w^T x_n) \]
Maximize the Likelihood

\[ \prod_{n=1}^{N} \theta(y_n w^T x_n) \]

Minimize

\[- \frac{1}{N} \ln \left( \prod_{n=1}^{N} \theta(y_n w^T x_n) \right) \]

\[ = \frac{1}{N} \sum_{n=1}^{N} \ln \left( \frac{1}{\theta(y_n w^T x_n)} \right) \]

\[ \left[ \theta(s) = \frac{1}{1 + e^{-s}} \right] \]

\[ E_{\text{in}}(w) = \frac{1}{N} \sum_{n=1}^{N} \ln \left( 1 + e^{-y_n w^T x_n} \right) \]

“Cross-Entropy” error, or Log-Loss error
How to Minimize $E_{\text{in}}$

• For Linear regression

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{w}^T \mathbf{x}_n - y_n)^2$$

← closed-form solution

• For logistic regression

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

← iterative solution
Gradient Descent

- Is a general method for nonlinear optimization
- Minimize a function via a greedy local search
- Scale well to large datasets

\[ f(x) \]

**Figure 1**: A univariate objective function

Minimizing \( f \) using gradient descent may yield a different local minimum depending on the start position.

(a) If \( f'(x) > 0 \) — so \( f \) is increasing — then move \( x \) a little to the left;

(b) If \( f'(x) < 0 \) then move \( x \) a little to the right.

Note that at each step, the derivative of \( f \) is used to decide which direction to move in.

Already with \( n = 1 \) and in Figure 1, it is clear that the outcome of gradient descent depends on the starting point. This example also shows how, with a non-convex function \( f \), gradient descent can compute a local minimum — meaning there's no way to improve \( f \) by moving a little bit in either direction — that is worse (i.e., larger) than a global minimum.

3. The converse also holds: if \( f \) is convex, then gradient descent can only terminate at a global minimum.

2.3 Warm-Up #2: Linear Functions

In almost all of the real applications of gradient descent, the number \( n \) of dimensions is much larger than 1. Already with \( n = 2 \) we see an immediate complication: from a point \( x \in \mathbb{R}^n \), there's an infinite number of directions in which we could move, not just 2.

3. Recall from last lecture that a function is convex if the region above its graph is a convex set. Equivalently, given two points on or above its graph, the entire line segment between the two points should also be on or above its graph. It's clear that the function shown in Figure 1 does not have this property.

4. Modulo any approximation error from stopping before the derivative is exactly zero; see Section 2.5 for details.

5. The same argument works for any number \( n \) of dimensions.
Gradient Descent in Logistic Regression

• Start at $\mathbf{w}(0)$, take a step along steepest slope
• Fixed step size $\eta$:

$$\mathbf{w}(1) = \mathbf{w}(0) + \eta \hat{\mathbf{v}}$$

What is the direction $\hat{\mathbf{v}}$?
Formula for the Direction

$$\Delta E_{in} = E_{in}(w(0) + \eta \hat{v}) - E_{in}(w(0))$$

$$= \eta \nabla E_{in}(w(0))^T \hat{v} + O(\eta^2)$$

$$\geq -\eta \Vert \nabla E_{in}(w(0)) \Vert$$

Since $\hat{v}$ is a unit vector,

$$\hat{v} = -\frac{\nabla E_{in}(w(0))}{\Vert \nabla E_{in}(w(0)) \Vert}$$
Step Size

• How large a step $\eta$ should take at each iteration?

$\eta$ too small

$\eta$ too large

variable $\eta$ – just right

$\eta$ should increase with the slope
Instead of

\[ \Delta w = \eta \hat{v} \]

\[ = -\eta \frac{\nabla E_{in}(w(0))}{\|\nabla E_{in}(w(0))\|} \]

Have

\[ \Delta w = -\eta \nabla E_{in}(w(0)) \]

Fixed learning rate \( \eta \)
The Logistic Regression Algorithm

1. Initialize the weights at \( t = 0 \) to \( \mathbf{w}(0) \)
2. \textbf{for} \( t = 0, 1, 2, \ldots \) \textbf{do}
3. \hspace{1em} Compute the gradient

\[
\nabla E_{\text{in}} = - \frac{1}{N} \sum_{n=1}^{N} \frac{y_n x_n}{1 + e^{y_n \mathbf{w}^T(t) x_n}}
\]

4. Update the weights: \( \mathbf{w}(t + 1) = \mathbf{w}(t) - \eta \nabla E_{\text{in}} \)
5. Iterate to the next step until it is time to stop
6. Return the final weights \( \mathbf{w} \)
Types of Gradient Descent

• Batch gradient descent
  • Use all training data points to calculate an error
• Stochastic gradient descent (SGD)
  • Use only one training data point to calculate an error at each iteration
  • It is much more efficient and scalable
Logistic Regression in Action

- **Dataset**
  - The built-in digits dataset from scikit-learn
- **Python Code**
  - Will be posted on the CCLE class website