Logistics: Exam 1

• Exam 1 Date: March 3, 2020 (Tuesday)
  • In-class exam (the same time/location as the lecture)
  • Exam duration: 75 minutes
  • Planned exam content: LFD Chapter 1 to 5
    • Everything in textbook/lectures are included, except for parts labeled as “safe to skip”.

• 2 sections
  • 5~6 long questions (written response questions with explanations required)
  • 10 multiple choice questions (no explanations needed)

• Closed-book exam. You can bring two cheat-sheets
  • Up to letter size, front and back (up to 4 pages)
  • No format limitations (it can be typed, written, or a combination)

• No calculators (you don’t need them)
Logistics: Exam Policies

• I plan to arrange random seat assignments
  • Will be announced on Piazza the night before the exam
Logistics: Exam Policies

• Please arrive on time. No extensions will be given if you arrive late.

• During the exam, if you have a question or if you finish before time is up:
  • **Do not get up**
  • Raise your hand and I will come to you
  • I most likely will not answer questions to individual students
    • But I’ll give clarifications to everyone if multiple students ask the same question

• When time is called:
  • **Stop writing**
  • **Do not get up**
  • Proctors will come around and collect your exam
Plans for Today

• A summary of the content so far.

• Discussion of the practice questions.

• Discussion of any other questions you might have.
Review for Exam 1

Brief overview on the content.
Not comprehensive and not covering everything that could appear in the exam.
Please make sure you still study for LFD Chapter 1-5.
Let me know if you find mistakes in lecture notes.
Whenever you have doubts on the lecture notes, please resort to the textbook for the confirmation.
• Chap 1: Setting up the learning problem
  • Problem setup
  • probability assumptions/inferences
  • error and noise

• Chap 2: Theory of generalization (training v.s. testing)
  • Hoeffding’s inequality
  • VC theory
  • Bias-variance decomposition

• Chap 3: Linear models
  • Linear classification/regression,
  • logistic regression, gradient descent,
  • nonlinear transformations

• Chap 4: Overfitting
  • Overfitting,
  • Regularization and validation

• Chap 5: Three learning principles
  • Occam’s razor, sampling bias, data snooping
Setup of the Learning Problem

- Key assumption:
  - Training/testing data from the same distribution

- Define (point-wise) error measure:
  - Binary error \( e(h(\vec{x}), y) = \mathbb{I}[h(\vec{x}) \neq y] \)
  - Squared error \( e(h(\vec{x}), y) = (h(\vec{x}) - y)^2 \)
  - Cost matrix

<table>
<thead>
<tr>
<th></th>
<th>( f )</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h = +1 )</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( h = -1 )</td>
<td></td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>( f )</th>
<th>1</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h = +1 )</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( h = -1 )</td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Hoeffding’s Inequality

• Fix a hypothesis $h$
  
  • $E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} e(h(\hat{x}_n), y_n) = $ In-sample error of $h$
  
  • $E_{out}(h) = \mathbb{E}_{\hat{x}}[e(h(\hat{x}), y)] = $ Out-of-sample error of $h$
  
  • Hoeffding’s inequality: $\Pr[|E_{out}(h) - E_{in}(h)| > \epsilon] \leq 2e^{-2\epsilon^2 N}$

• Learn a $g$ from a finite hypothesis set $H = \{h_1, ..., h_M\}$
  
  • $\Pr[|E_{out}(g) - E_{in}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$
Dealing with Infinite Hypothesis Set: $M \rightarrow \infty$

- Instead of # hypothesis, counting “effective” # hypothesis

  **Dichotomy**
  - Informally, consider it as “data-dependent” hypothesis
  - Characterized by both $H$ and $N$ data points $(\tilde{x}_1, ..., \tilde{x}_N)$
    \[ H(\tilde{x}_1, ..., \tilde{x}_N) = \{ h(\tilde{x}_1), ..., h(\tilde{x}_N) | h \in H \} \]
  - The set of possible prediction combinations $h \in H$ can induce on $\tilde{x}_1, ..., \tilde{x}_N$

  **Growth function**
  - Largest number of dichotomies $H$ can induce across all possible data sets of size $N$
    \[ m_H(N) = \max \limits_{(\tilde{x}_1, ..., \tilde{x}_N)} |H(\tilde{x}_1, ..., \tilde{x}_N)| \]
Why Growth Function?

• Finite-hypothesis Bound
  With prob at least $1 - \delta$,
  \[ E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}} \]

• VC Generalization Bound (VC Inequality, 1971)
  With prob at least $1 - \delta$
  \[ E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} \ln \frac{4m_H(2N)}{\delta}} \]

If we know the growth function $m_H(N)$ of $H$, we can obtain the learning guarantee for algorithms operating on $H$. 
Bounding Growth Functions

• More definitions....
  • Shatter
    • \( H \) **shatters** \((\vec{x}_1, ... , \vec{x}_N)\) if \(|H(\vec{x}_1, ... , \vec{x}_N)| = 2^N\)
    • \( H \) can induce all label combinations for \((\vec{x}_1, ... , \vec{x}_N)\)
  • Break point
    • \( k \) is a **break point** for \( H \) if no data set of size \( k \) can be shattered by \( H \)
    • \( k \) is a break point for \( H \leftrightarrow m_H(k) < 2^k \)

• VC Dimension: \( d_{vc}(H) \) or \( d_{vc} \)
  • The VC dimension of \( H \) is the largest \( N \) such that \( m_H(N) = 2^N \)
  • Equivalently, if \( k^* \) is the smallest break point for \( H \), \( d_{vc}(H) = k^* - 1 \)
# Examples

<table>
<thead>
<tr>
<th></th>
<th>(m_H(N))</th>
<th>(N=1)</th>
<th>(N=2)</th>
<th>(N=3)</th>
<th>(N=4)</th>
<th>(N=5)</th>
<th>Break Points</th>
<th>VC Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive Rays</td>
<td>(k = 2,3,4, \ldots)</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>(k = 2,3,4, \ldots)</td>
<td>1</td>
</tr>
<tr>
<td>Positive Intervals</td>
<td>(k = 3,4,5, \ldots)</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>11</td>
<td>16</td>
<td>(k = 3,4,5, \ldots)</td>
<td>2</td>
</tr>
<tr>
<td>Convex Sets</td>
<td>None</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>None</td>
<td>(\infty)</td>
</tr>
<tr>
<td>2D Perceptron</td>
<td>(k = 4,5,6, \ldots)</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>14</td>
<td>?</td>
<td>(k = 4,5,6, \ldots)</td>
<td>3</td>
</tr>
</tbody>
</table>

- **Positive Rays**: Predict +1 or -1 based on the position relative to the decision boundary.
- **Positive Intervals**: The boundaries are defined by the points where the function changes its prediction from +1 to -1 or vice versa.
- **Convex Sets**: The convex sets are defined by the region where the function is positive.
- **2D Perceptron**: The perceptron decision boundary separates the positive and negative examples.
Bounding Growth Functions using Break Points

• Theorem statement:
  • If there is no break point for $H$, then $m_H(N) = 2^N$ for all $N$.
  • If $k$ is a break point for $H$, i.e., if $m_H(k) < 2^k$ for some value $k$, then
    \[ m_H(N) \leq \sum_{i=0}^{k-1} \binom{N}{i} \]

• Rephrase the above theorem
  • If $k$ is a break point for $H$, the following statements are true
    • $m_H(N) \leq N^{k-1} + 1$ [Can be proven using induction from above. See LFD Problem 2.5]
    • $m_H(N) = O(N^{k-1})$
    • $m_H(N)$ is polynomial in $N$

• If $d_{vc}$ is the VC dimension of $H$, then
  • $m_H(N) \leq \sum_{i=0}^{d_{vc}} \binom{N}{i}$
  • $m_H(N) \leq N^{d_{vc}} + 1$
  • $m_H(N) = O(N^{d_{vc}})$

If $d_{vc}$ is the VC dimension of $H$, $d_{vc} + 1$ is a break point for $H$
Vapnik–Chervonenkis (VC) Bound

• VC Generalization Bound

With prob at least $1 - \delta$

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} \ln \frac{4m_H(2N)}{\delta}}$$

• Let $d_{vc}$ be the VC dimension of $H$, we have $m_H(N) \leq N^{d_{vc}} + 1$. Therefore,

With prob at least $1 - \delta$

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} \ln \frac{4((2N)^{d_{vc}}+1)}{\delta}}$$

• If we treat $\delta$ as a constant, then we can say, with high probability

$$E_{out}(g) \leq E_{in}(g) + O\left(\sqrt{d_{vc} \frac{\ln N}{N}}\right)$$
Approximation-Generalization Tradeoff

- **VC Dimension**: A single parameter to characterize the complexity of $H$

$$E_{out}(g) \leq E_{in}(g) + O\left(\sqrt{d_{vc} \frac{\ln N}{N}}\right)$$
Bias-Variance Decomposition

\[
\mathbb{E}_D[E_{out}(g^{(D)})] = \mathbb{E}_{\tilde{x}} \left[ (\bar{g}(\tilde{x}) - f(\tilde{x}))^2 \right] + \mathbb{E}_{\tilde{x}} \left[ \mathbb{E}_D \left[ (g^{(D)}(\tilde{x}) - \bar{g}(\tilde{x}))^2 \right] \right]
\]

• The performance of your learning, i.e., \( \mathbb{E}_D[E_{out}(g^{(D)})] \), depends on
  • How well you can fit your data using your hypothesis set (bias)
  • How close to the best fit you can get for a given dataset (variance)

Very small model

Very large model
Learning Curves

Simple Model:
- $E_{out}$
- $E_{in}$

Expected Error vs. Number of Data Points, $N$

Complex Model:
- $E_{out}$
- $E_{in}$

Expected Error vs. Number of Data Points, $N$

VC Analysis:
- Expected Error vs. Number of Data Points, $N$
- Generalization error
- In-sample error

Bias-Variance Analysis:
- Expected Error vs. Number of Data Points, $N$
- Variance
- Bias
Linear Models

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

<table>
<thead>
<tr>
<th>Domain</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Classification</td>
<td>$y \in {-1,+1}$ $H = {h(\vec{x}) = \text{sign}(\vec{w}^T \vec{x})}$</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>$y \in \mathbb{R}$ $H = {h(\vec{x}) = \vec{w}^T \vec{x}}$</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>$y \in [0,1]$ $H = {h(\vec{x}) = \theta(\vec{w}^T \vec{x})}$</td>
</tr>
</tbody>
</table>

This is why it’s called linear models

- Algorithm:
  - Focus on $g = \arg\min_{h \in H} E_{in}(h)$
Linear Classification

• Formulation
  • Hypothesis set $H = \{ h(\hat{x}) = \text{sign}(\hat{w}^T \hat{x}) \}$
  • Error measure: binary error $e(h(\hat{x}), y) = \mathbb{I}[h(\hat{x}) \neq y]$

• Data is linearly separable
  • Run PLA => $E_{in} = 0$ => Low $E_{out}$

• Data is not linearly separable
  • Engineering the features
  • Pocket algorithm

---

**Perceptron Learning Algorithm (PLA)**

Initialize $\hat{w}(0) = \vec{0}$

For $t = 0, \ldots$

Find a misclassified example $(\hat{x}(t), y(t))$ in $D$
  that is, $\text{sign}(\hat{w}(t)^T \hat{x}(t)) \neq y(t)$

If no such sample exists
  Return $\hat{w}(t)$

Else
  $\hat{w}(t + 1) \leftarrow \hat{w}(t) + y(t) \hat{x}(t)$
Linear Regression

• Formulation
  • Hypothesis set \( H = \{ h(\hat{x}) = \vec{w}^T \hat{x} \} \)
  • Squared error \( e(h(\hat{x}), y) = (h(\hat{x}) - y)^2 \)

• Linear regression algorithm (one-step learning for solving \( \nabla_{\vec{w}} E_{in}(\vec{w}_{lin}) = 0 \))
  • Given \( D = \{(\hat{x}_1, y_1), \ldots, (\hat{x}_N, y_N)\} \)
  • Construct \( X = \begin{bmatrix} \hat{x}_1^T \\ \vdots \\ \hat{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{1,0} & x_{1,1} & \cdots & x_{1,d} \\ x_{2,0} & x_{2,1} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{2,0} & x_{N,1} & \cdots & x_{N,d} \end{bmatrix} \) 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)
Logistic Regression

- Hypothesis set \( H = \{ h(\vec{x}) = \theta(\vec{w}^T \vec{x}) \} \)
  \( \theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}} \)

- Predict a probability
  - Interpreting \( h(\vec{x}) \) as the prob for \( y = +1 \) given \( \vec{x} \) when \( h \) is the target function

- Algorithm
  - Find \( g = \text{argmin}_{h \in H} E_{in}(h) \)

- Two key questions
  - How to define \( E_{in}(h) \)?
  - How to perform the optimization (minimizing \( E_{in} \))?
Define \( E_{in}(\vec{w}) \): 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 cross entropy error is the same as maximizing likelihood

• Likelihood: \( \Pr(D|\vec{w}) \)
  
  • \( \vec{w}^* = \text{argmax}_{\vec{w}} \Pr(D|\vec{w}) \) (maximizing likelihood)
  
  • \( \vec{w}^* = \text{argmin}_{\vec{w}} E_{in}(\vec{w}) \) (minimizing cross-entropy error)
Optimizing $E_{in}(\vec{w})$: Gradient Descent

• Gradient descent algorithm
  • Initialize $\vec{w}(0)$
  • For $t = 0, ...$
    • $\vec{w}(t + 1) \leftarrow \vec{w}(t) - \eta \nabla_w E_{in}(\vec{w}(t))$
    • Terminate if the stop conditions are met
  • Return the final weights

• Stochastic gradient decent
  • Replace the update step:
    • Randomly choose $n$ from \{1, ..., $N$\}
    • $\vec{w}(t + 1) \leftarrow \vec{w}(t) - \eta \nabla_w e_n(\vec{w}(t))$

Works for functions where gradient exists everywhere
Nonlinear Transformation

\[
\hat{z} = \Phi(\hat{x})
\]

\[
g^{(z)}(\hat{z}) = \text{sign}(\tilde{w}^{(z)^T} \hat{z})
\]

\[
g(\hat{x}) = g^{(z)}(\Phi(\hat{x})) = \text{sign}(\tilde{w}^{(z)^T} \Phi(\hat{x}))
\]
MUST Choose $\Phi$ BEFORE Looking at the Data

- Rely on domain knowledge (feature engineering)
  - Handwriting digit recognition example

- Use common sets of feature transformation
  - Polynomial transformation
  - E.g., 2nd order Polynomial transformation
    - $\tilde{x} = (1, x_1, x_2)$, $\Phi_2(\tilde{x}) = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$
    - Plus: more powerful (contains circle, ellipse, hyperbola, etc)
    - Minus:
      - More computation/storage
      - Worse generalization error

The VC dimension of d-dim perceptron is $d+1$
Q-th Order Polynomial Transform

• \( \tilde{x} = (1, x_1, ..., x_d) \)
• \( \Phi_1(\tilde{x}) = \tilde{x} \)
• \( \Phi_Q(\tilde{x}) = (\Phi_{Q-1}(\tilde{x}), x_1^Q, x_1^{Q-1}x_2, ..., x_d^Q) \)

• Each element in \( \Phi_Q(\tilde{x}) \) is in the form of \( \sum_{i=1}^{d} x_i^{a_i} \)
  • where \( \sum_{i=1}^{d} a_i \leq Q \), and \( a_i \) is a non-negative integer

• Number of elements in \( \Phi_Q(\tilde{x}) \): \( \binom{Q + d}{Q} \) (including the initial 1)
Overfitting and Its Cures

- **Overfitting**
  - Fitting the data more than is warranted
  - Fitting the noise instead of the pattern of the data
  - Decreasing $E_{\text{in}}$ but getting larger $E_{\text{out}}$
  - When $H$ is too strong, but $N$ is not large enough

- **Regularization**
  - Intuition: Constraining $H$ to make overfitting less likely to happen

- **Validation**
  - Intuition: Reserve data to estimate $E_{\text{out}}$
Regularization

• Constraining $H$
  • Example: Weight decay $H(C) = \{h \in H_Q \text{ and } \overrightarrow{w}^T \overrightarrow{w} \leq C\}$
  • Finding $g$ => Constrained optimization

• Defining augmented error
  • $E_{aug}(h, \lambda, \Omega) = E_{in}(\overrightarrow{w}) + \frac{\lambda}{N} \Omega(h)$
  • Finding $g$ => Unconstrained optimization

• The two interpretations are conceptually equivalent in a lot of cases.

• Understand the impacts of choosing $\Omega$ and $\lambda$
Validations

- Reserving data to estimate $E_{\text{out}}$

### Model Selection

<table>
<thead>
<tr>
<th>Disjoint Data Splits</th>
<th>Outlook</th>
<th>Relationship to $E_{\text{out}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{\text{train}}$</td>
<td>$E_{\text{in}}$</td>
<td>Incredibly optimistic</td>
</tr>
<tr>
<td>$D_{\text{val}}$</td>
<td>$E_{\text{val}}$</td>
<td>Slightly optimistic</td>
</tr>
<tr>
<td>$D$</td>
<td>$E_{\text{test}}$</td>
<td>Unbiased</td>
</tr>
</tbody>
</table>
Cross Validation

• Split $D$ into $V$ equally sized data sets: $D_1, D_2, \ldots, D_V$
  
  • Let $g_i$ be the hypothesis learned using all data sets except $D_i$
  
  • Let $e_i = E_{val}(g_i)$ where the validation uses data set $D_i$

• The $V$-fold cross validation error is $\frac{1}{V} \sum_{i=1}^{V} e_i$

• Leave-One-Out Cross Validation (LOOCV): $V = N$

![Diagram of cross validation process]

$E_{cv} = \frac{1}{3} (d_1^2 + d_2^2 + d_3^2)$
Three Learning Principles

• Occam’s Razor
  • The **simplest** model that fits the data is also the most **plausible**

• Sampling Bias
  • If the data is sampled in a biased way, learning will produce a similarly biased outcome.

• Data Snooping
  • If a data set has affected any step in the learning process, its ability to assess the outcome has been compromised.
Practice Questions