Deep Learning Theory and Practice

Lecture 3
Real learning is feasible

Dr. Ted Willke
willke@pdx.edu

Monday, April 8, 2019
Review of Lecture 2

- **Is learning feasible?** Yes, in a probabilistic sense.

- **Can we use Hoeffding to assess real learning?**
  - Yes, if we account for “selection bias”
  - Recall coin example

- If we select \( h \in H \) with the smallest \( E_{in} \), can we expect \( E_{out} \) to be small?

---

\[
E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} [h(x) \neq f(x)]
\]

\[
E_{out}(h) = \mathbb{P}_x[h(x) \neq f(x)]
\]

**Hoeffding:** \( E_{out}(h) \approx E_{in}(h) \)

\[
\mathbb{P}[|E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2e^{-2\epsilon^2 N}, \quad \epsilon > 0.
\]
Today’s Lecture

• Real learning versus verification

• The two step solution to learning

• Linear regression

• Dealing with error and noise

(Many slides adapted from Yaser Abu-Mostafa and Malik Magdon-Ismail, with permission of the authors. Thanks guys!)
**Interpreting Hoeffding Bound for finite $|H|$**

\[
\mathbb{P}[ |E_{in}(h_j) - E_{out}(h_j) | > \epsilon ] \leq 2e^{-2\epsilon^2 N}, \quad \text{for any } \epsilon > 0.
\]

This is for one fixed hypothesis $h_j$. What about the relationship for all $h \in H$, given $|H| = M$?

$A_j = \text{event that } |E_{out}(h_j) - E_{in}(h_j) | > \epsilon$, \quad $\mathbb{P}(A_j) \leq 2e^{-2\epsilon^2 N}$

\[
\mathbb{P}(\exists h_j \in H \mid |E_{out}(h_j) - E_{in}(h_j) | > \epsilon) = \mathbb{P}(A_1 \cup A_2 \cup \ldots \cup A_M)
\]
\[
\leq \sum_{i=1}^{M} \mathbb{P}(A_i)
\]
\[
\leq \sum_{i=1}^{M} 2e^{-2\epsilon^2 N} = 2Me^{-2\epsilon^2 N}.
\]
The ‘uniform convergence’ result

\[ \mathbb{P}(\exists h_j \in H \mid |E_{out}(h_j) - E_{in}(h_j)| > \epsilon) \leq 2Me^{-2\epsilon^2N} \]

\[ \mathbb{P}(\nexists h_j \in H \mid |E_{out}(h_j) - E_{in}(h_j)| > \epsilon) = \mathbb{P}(\forall h_j \in H \mid |E_{out}(h_j) - E_{in}(h_j)| \leq \epsilon) \quad (\text{taking } 1 - \text{both sides}) \]
\[ \geq 1 - 2Me^{-2\epsilon^2N}. \]

So, with probability \( 1 - 2Me^{-2\epsilon^2N} \), \( E_{in} \) will be within \( \epsilon \) of \( E_{out} \) simultaneously for all \( h \in H \).

‘uniform convergence’ result

- \( E_{in} \) converges to \( E_{out} \) simultaneously for all \( h \in H \) as \( N \) gets larger
- Bound is \( M \) looser than one hypothesis bound
Other equivalent forms of the bound

1. Given \( \delta = \mathbb{P}(\cdot) \) and \( \epsilon \), how large a training set is needed (i.e., solve for \( N \))?

Solution: \( \delta = 2M e^{-2 \epsilon^2 N} \). Solve for \( N \).

So long as \( N \geq \frac{1}{2e^2} \log \frac{2M}{\delta} \), then with probability \( 1 - \delta \), we have \( |E_{out}(h) - E_{in}| \leq \epsilon \) for all \( h \in H \).

‘Sample complexity’ bound

[Note: \( N \) grows as \( \log M \), so very slow growing.]
Other equivalent forms of the bound

2. Holding $N, \delta$ fixed, solve for $\epsilon$.

With probability $1 - \delta$, we have that $\forall h \in H, |E_{in}(h) - E_{out}(h)| \leq \sqrt{\frac{1}{2N} \log \frac{2M}{\delta}}$.

'Sorry bound'

Solve for $\epsilon$ and plug in.

Next question: Assuming uniform convergence holds, i.e.,

$\forall h \in H, |E_{in}(h) - E_{out}(h)| \leq \epsilon$, can we prove something about $E_{out}(g)$?
Bounding the generalization error

\[ \mathbb{P}[ |E_{\text{in}}(h_j) - E_{\text{out}}(h_j)| > \epsilon] \leq 2e^{-2\epsilon^2N}, \quad \epsilon > 0 \]  
(Hoeffding’s)  

(1)

\[ g = \arg \min_{h \in H} E_{\text{in}}(h) \]  
(hypothesis chosen by E.R.M)  

(2)

\[ h^* = \arg \min_{h \in H} E_{\text{out}}(h) \]  
(hypothesis with smallest generalization error)  

(3)

where \( h^* \) is the hypothesis in H with the smallest generalization error.

Then it follows that:

\[ E_{\text{out}}(g) \leq E_{\text{in}}(g) + \epsilon \]  
by Hoeffding’s for \( g \)

\[ \leq E_{\text{in}}(h^*) + \epsilon \]  
by (2), since \( h^* \) can’t beat \( g \) on \( E_{\text{in}} \)

\[ \leq E_{\text{out}}(h^*) + 2\epsilon \]  
by Hoeffding’s for \( h^* \)
Dataset size scaling requirement

Let $|H| = M$. Let any $\delta, \epsilon$ be fixed as well.

Then in order to guarantee that $E_{out}(g) \leq \min_{h \in H} E_{out}(h) + 2\epsilon$

with probability $1 - \delta$, it suffices that:

$$N \geq \frac{1}{2e^2} \log \frac{2M}{\delta} = \mathcal{O}\left(\frac{1}{\epsilon^2 \log \frac{M}{\delta}}\right)$$

**Takeaway:** If $N \gg \ln |H|$, then $E_{out}(g) \approx E_{in}(g)$.

- Doesn’t depend on $X$, $\mathbb{P}(x)$, $f$ or how $g$ is found.
- Only requires $\mathbb{P}(x)$ to generate the points independently and also the test point.
The 2-step approach to learning

\begin{align*}
(1) & \quad E_{out}(g) \approx E_{in}(g). \\
(2) & \quad E_{in}(g) \approx 0.
\end{align*}

- Together, these ensure \( E_{out} \approx 0 \).
- How to verify (1), since we don’t know \( E_{out} \)?
- We can ensure (2) (or know when we can’t!)

Caution: There is a tradeoff:

- Small \( |H| \) \( \implies \) \( E_{in} \approx E_{out} \)
- Large \( |H| \) \( \implies \) \( E_{in} \approx 0 \) is more likely.

Why??
The tradeoff

Recall $E_{out}(g) \leq E_{in}(g) + \epsilon$ and $\epsilon = \sqrt{\frac{1}{2N} \log \frac{2|H|}{\delta}}$

$h \in$ linear hypotheses

$h \in$ quadratics

$h \in n^{th}$-order polynomials

$H_{linear} \subseteq H_{quad} \subseteq H_{n polys}$

What about infinite $H$, like the perceptron?
‘Complex’ target functions are harder to learn

What happened to the ‘difficulty’ (complexity) of $f$?

- Simple $f \implies$ can use small $|H|$ to get $E_{in} \approx 0$ (requires less data)
- Complex $f \implies$ need large $|H|$ to get $E_{in} \approx 0$ (requires more data)
Revisiting the learning problem

unknown target function
\( f : X \mapsto Y \)

training examples
\((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\)

unknown input distribution
\( P(x) \)

learning algorithm
\( A \)

final hypothesis
\( g \approx f \)
Linear regression: credit example

**Classification:** Credit approval (yes or no)

**Regression:** Credit line (dollar amount)

Input: \( x = \)

<table>
<thead>
<tr>
<th>Age (years)</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salary (dollars)</td>
<td>80,000</td>
</tr>
<tr>
<td>Debt (dollars)</td>
<td>26,000</td>
</tr>
<tr>
<td>Employed (years)</td>
<td>3</td>
</tr>
<tr>
<td>In residence (years)</td>
<td>2.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Linear regression output: \( h(x) = \sum_{i=0}^{d} w_i x_i = w^T x \)
Linear regression: the dataset

Credit officers decide on credit lines:

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

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

Linear regression attempts to replicate this.
Linear regression: measuring the error

How well does $h(x) = w^T x$ approximate $f(x)$?

In linear regression, we use the squared error $(h(x) - f(x))^2$.

in-sample error: $E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} (h(x_n) - y_n)^2$

(compare with in-sample error for classification)
Linear regression: illustration
Expression for $E_{in}$

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

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

where

\[
X = \begin{bmatrix}
-x_1^T \\
-x_2^T \\
\vdots \\
-x_N^T
\end{bmatrix}, \quad y = \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N
\end{bmatrix}
\]
Minimizing $E_{in}$

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

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

$$X^T Xw = X^T y$$

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

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

\[ X^\dagger = (X^T X)^{-1} X^T \]
The linear regression algorithm

Linear regression algorithm:

1: Construct the matrix $X$ and the vector $y$ from the dataset $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ as follows:

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

$$x_i^T$$ denotes the transpose of $x_i$.

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

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

$w$ is the weight vector.
Error measures

What does "$h \approx f$" mean?

Error measure: $E(h, f)$

Almost always a pointwise definition: $e(h(x), f(x))$

Examples:

Squared error: $e(h(x), f(x)) = (h(x) - f(x))^2$ (regression)

Binary error: $e(h(x), f(x)) = [[h(x) \neq f(x)]]$ (classification)
From pointwise to overall

Overall error $E(h,f)$ is the average of pointwise errors $e(h(x), f(x))$.

In-sample error:

$$E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} e(h(x_n), f(x_n))$$

Out-of-sample error:

$$E_{out}(h) = \mathbb{E}_x [e(h(x_n), f(x_n))]$$
How to select the error measure

Fingerprint verification:

Two types of error:

- **false accept**
- **false reject**

How do we penalize each type?

<table>
<thead>
<tr>
<th>$h$</th>
<th>$+1$</th>
<th>$-1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td>no error</td>
<td>false accept</td>
</tr>
<tr>
<td>-1</td>
<td>false reject</td>
<td>no error</td>
</tr>
</tbody>
</table>

$\{ +1 \quad \text{you} \}
\{ -1 \quad \text{intruder} \}$
Supermarket verifies fingerprint for discounts

False reject is costly; customer gets annoyed!

False accept is minor; gave away a discount and intruder left their fingerprint :-)
The error measure - for the CIA

CIA verifies fingerprint for security

False accept is a disaster!

False reject can be tolerated
Try again; you are an employee :-)

\[
\begin{array}{c|cc}
   h & f & \{ & +1 \quad \text{you} \\
   & +1 & -1 & \\
   & 0 & 1000 & \\
   & 1 & 0 & \\
\end{array}
\]
Take-home lesson

The error measure should be specified by the user.

Not always possible. Alternatives:

- **Plausible measures:** squared error \(\equiv\) Gaussian noise
- **Friendly measures:** closed-form solution, convex optimization
The learning problem - with error measure

**UNKNOWN TARGET FUNCTION**

\[ f : X \mapsto Y \]

**TRAINING EXAMPLES**

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

**LEARNING ALGORITHM**

\[ A \]

**HYPOTHESIS SET**

\[ H \]

**FINAL HYPOTHESIS**

\[ g \approx f \]

**UNKNOWN INPUT DISTRIBUTION**

\[ P(x) \]

**ERROR MEASURE**

\[ e(\ ) \]

\[ x_1, x_2, \ldots, x_N \]

\[ x \]
Noisy targets

The ‘target function’ is not always a function.

Consider credit approval problem:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age (years)</td>
<td>27</td>
</tr>
<tr>
<td>salary (dollars)</td>
<td>80,000</td>
</tr>
<tr>
<td>debt (dollars)</td>
<td>26,000</td>
</tr>
<tr>
<td>employed (years)</td>
<td>3</td>
</tr>
<tr>
<td>in residence (years)</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Two ‘identical’ customers ➔ two different behaviors.
Target ‘distribution’

Instead of \( y = f(x) \), we use a target distribution:

\[
P(y | x)
\]

\((x, y)\) is now generated by the joint distribution:

\[
P(x)P(y | x)
\]

Noisy target = deterministic target \( f(x) = \mathbb{E}(y | x) \) plus noise \( y - f(x) \)

Deterministic target is a special case of noisy target:

\[
P(y | x) \text{ is zero except for } y = f(x)
\]
The learning problem - including a noisy target

**UNKNOWN TARGET DISTRIBUTION**

\[ P(y | x) \]

target function \( f : X \mapsto Y \) plus noise

**TRAINING EXAMPLES**

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

**ERROR MEASURE**

\( e(\ ) \)

**LEARNING ALGORITHM**

\( A \)

**FINAL HYPOTHESIS**

\( g \approx f \)

**UNKNOWN INPUT DISTRIBUTION**

\[ P(x) \]
**Distinction between** $P(y \mid x)$ **and** $P(x)$

Both convey probabilistic aspects of $x$ and $y$.

The target distribution $P(y \mid x)$ is what we are trying to learn.

The input distribution $P(x)$ quantifies the relative importance of $x$.

Merging $P(x)P(y \mid x)$ as $P(x, y)$ mixes the two concepts.
Further reading