$E_{out}(g)$ and $E_{in}(g)$

- Previously, we proved that $E_{out}(g) \approx E_{in}(g)$ given certain conditions.

- But how can we find $g$ s.t. $E_{in}(g) \approx 0$?
### 3 Learning Problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>$y = {-1, +1}$</td>
</tr>
<tr>
<td>Predicting Probabilities</td>
<td>$y = [0, 1]$</td>
</tr>
<tr>
<td>Regression</td>
<td>$y = \mathbb{R}$</td>
</tr>
</tbody>
</table>
Our hypotheses will all be some function of a linear combination of the input

Don’t forget to pad each input with a leading 1

The values to be learned are the $d+1$ coefficients, $w_0$ to $w_d$
# Learning Solutions

<table>
<thead>
<tr>
<th>Problem</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Classification</td>
<td>$y = {-1, +1}$</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>$y = [0, 1]$</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>$y = \mathbb{R}$</td>
</tr>
</tbody>
</table>
3 Learning Solutions

<table>
<thead>
<tr>
<th>Problem</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Classification</td>
<td>$h(\hat{x}) = \text{sign}(\overline{w}^T \hat{x})$</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>$h(\hat{x}) = \theta(\overline{w}^T \hat{x})$</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>$h(\hat{x}) = \overline{w}^T \hat{x}$</td>
</tr>
</tbody>
</table>
What is $\theta$?

- $\theta$ is a sigmoid function, an “$S$”-shaped function that maps $\mathbb{R} \to [0, 1]$

- $\theta(x) = \frac{1}{1 + e^{-x}}$ or $\theta(x) = \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{y^2}{2}} dy$
Recall that a learning model requires specifying a hypothesis set, a learning algorithm and an error metric (these last two things are often closely related e.g. specifying a learning algorithm implies an error metric or specifying an error metric implies a learning algorithm)
Squared Error

\[ E_{ln}(h) = \frac{1}{n} \sum_{i=1}^{n} (h(x^*_i) - y_i)^2 \]
\[ E_{ln}(\bar{w}) = \frac{1}{n} \sum_{i=1}^{n} (\bar{w}^T \bar{x}_i - y_i)^2 \]

\[ = \frac{1}{n} \sum_{i=1}^{n} (\bar{x}_i^T \bar{w} - y_i)^2 \]

\[ = \frac{1}{n} \|X\bar{w} - \bar{y}\|^2 \text{ where } \|\bar{z}\| = \sqrt{\sum_{i=1}^{d} z_i^2} = \sqrt{\bar{z}^T \bar{z}} \]

\[ = \frac{1}{n} (X\bar{w} - \bar{y})^T (X\bar{w} - \bar{y}) \]
Minimizing Error

- Find the gradient
- Set it equal to zero
- Solve
- (Check that the solution is a minimum)
Minimizing Error

\[ E_{in}(\mathbf{w}) = \frac{1}{n} (X\mathbf{w} - \mathbf{y})^T (X\mathbf{w} - \mathbf{y}) \]
\[ = \frac{1}{n} (X\mathbf{w})^T - \mathbf{y}^T \) (X\mathbf{w} - \mathbf{y}) \]
\[ = \frac{1}{n} (\mathbf{w}^T X^T X\mathbf{w} - 2\mathbf{w}^T X^T \mathbf{y} + \mathbf{y}^T \mathbf{y}) \]
\[ \rightarrow \nabla_{\mathbf{w}} E_{in}(\mathbf{w}) = \frac{1}{n} (2X^T X\mathbf{w} - 2X^T \mathbf{y}) \]
Minimizing Error

\[ E_{in}(\bar{w}) = \frac{1}{n} (X\bar{w} - \bar{y})^T(X\bar{w} - \bar{y}) \]
\[ = \frac{1}{n} \left( (X\bar{w})^T - \bar{y}^T \right) (X\bar{w} - \bar{y}) \]
\[ = \frac{1}{n} (\bar{w}^T X^T X\bar{w} - 2\bar{w}^T X\bar{y} + \bar{y}^T \bar{y}) \]

\[ \Rightarrow \nabla_{\bar{w}} E_{in}(\bar{w}^*) = \frac{1}{n} (2X^T X\bar{w}^* - 2X^T \bar{y}) = 0 \]
\[ \Rightarrow 2X^T X\bar{w}^* = 2X^T \bar{y} = 0 \]
\[ \Rightarrow X^T X\bar{w}^* = X^T \bar{y} \]
\[ \Rightarrow \bar{w}^* = (X^T X)^{-1} X^T \bar{y} \]
\[ \nabla_{\tilde{w}} E_{\text{lin}}(\tilde{w}) = \frac{1}{n} (2X^T X \tilde{w} - 2X^T \tilde{y}) \]

\[ H_{\tilde{w}} E_{\text{lin}}(\tilde{w}) = \frac{1}{n} (2X^T X) \]

\[ \rightarrow H_{\tilde{w}} E_{\text{lin}}(\tilde{w}) \text{ is positive semidefinite} \]

\[ \frac{2}{n} (v^T X^T X v) = \frac{2}{n} (Xv)^T (Xv) \geq 0 \text{ because all squares are greater than or equal to 0 and } n \geq 0 \]
If \( n \gg d \), then it is very likely that \( X^T X \) is nonsingular (because there will likely be \( d+1 \) linearly independent vectors \( \overline{x}_i \)), which in turn implies both that \( X^T X \) is invertible and that \( X^T X \) is positive definite.

If \( X^T X \) is singular then there will be multiple \( \overline{w}^* \) that minimize the in-sample error and there are different ways of finding them that we won’t go into (see the book).
Don’t forget to pad X with a column of all 1’s in the front

You’ll sometimes hear this algorithm called ordinary least squares regression because it returns a linear regressor (or predictor) that has the smallest squared error on the training data.

Just because there aren’t any iterations (like PLA) doesn’t mean we haven’t learned; linear regression is one of the few learning algorithms where a closed form solution exists.
\( \mathcal{X} = \mathbb{R}^d \) and \( Y = \{-1, +1\} \)

\[ \mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \]

\[ X = \begin{bmatrix} 1 & x_1^T \\ 1 & x_2^T \\ \vdots & \vdots \\ 1 & x_n^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nd} \end{bmatrix} \text{ and } \bar{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \]

• Binary Error
If we have data that is not linearly separable, which is more commonly the case, then not only will PLA never terminate but it’s behavior can be indefinitely unstable.
Recall from the first lecture how the first few iterations of PLA resulted in crazy swings (going from predicting all points as positive to predicting all points as negative); with linearly inseparable data, you could see that behavior forever, even after a thousand or a million iterations.

This instability means that simply picking some cutoff number of iterations and using whatever $\mathbf{w}$ that PLA has saved when the algorithm completes that many iterations can result in some really bad predictors.
Alright so PLA isn’t going to work; let’s try and do what we did with linear regression: start with the error metric and try and find an exact solution, some set of weights $\overline{w}^*$ that minimizes the error

$$E_{\text{lin}}(h) = \frac{1}{n} \sum_{i=1}^{n} \left[ h(x_i) \neq y_i \right]$$
Here’s the problem: both the identity function (the double brackets) and the sign function are non-differentiable functions. For those of you interested in this kind of thing, minimizing this quantity exactly is NP-hard.
Now $\vec{y}$ is a vector of all plus 1’s and minus 1’s but that’s fine; we can still calculate $\vec{w}^*$

However, if want to use this $\vec{w}^*$, we have to make sure we’re making valid predictions for classification so our final hypothesis still uses the sign function.
So we talked about how simply stopping PLA after a fixed number of iterations can be bad. Here's a better alternative:

Let $T$ be a budget of PLA iterations

Each iteration will take longer than a single iteration of PLA because computing $E_{in}(\overline{w})$ requires going through each training point.

Unfortunately, there are no guarantees about how well the pocket algorithm will do or how quickly it’ll converge to a good $E_{in}$.