Linear and non-linear classifiers

- Machine Learning
  - Supervised Learning
    - Basic Models — Decision Trees — linear and non-linear classifiers

Inspired by Numerical Methods:
Regression Review.

Examples $E \langle x_1, x_2, x_3 \ldots x_n, y \rangle \quad x_i \in \mathbb{R} \quad y \in \mathbb{R}$

we are trying to learn $\hat{Y} = (x_2, x_2, \ldots x_n) : Y$

Linear regression.
Assume $\hat{Y} = w_0 + x_1 w_1 + x_2 w_2 + x_3 w_3 + \ldots + x_n w_n$

where $w_i$ are unknown.
find such $w_i$!!

- Introduce $x_0 = 1$ always. then:
  $\hat{Y} = \sum_{i=0}^{n} w_i x_i$

Error Function: to compare $\hat{Y}$ and $Y$

$\text{Error} (E) = \sum_{e \in E} (Y(e) - \hat{Y}(e))^2$

$= \sum_{e \in E} \left( Y(e) - \sum_{i=0}^{n} w_i x_i(e) \right)^2$
find $w_i$'s that minimize $\text{Error}(E)$?

**Technique:** Gradient Descent

Iterative Method to find Minima of Functions

![Cartoon Version of Gradient Descent](image)

- $a_{n+1} \leftarrow a_n + \eta \cdot f'(a)$
- $\eta$: learning rate
- $f'(a)$: derivative of $f$

Function must be differentiable.

$b = f(a)$

---

So: Apply Gradient Descent to our Error $(E)$ function:

**Examples:**

$Y$, $X_1$,

$\hat{Y} = w_0 + X_1 w_1$

$\text{Error} = (Y - (w_0 + X_1 w_1))^2$

**Update Rule:**

$w_i \leftarrow w_i - \eta \cdot \frac{\partial}{\partial w_i} \text{Error}(E)$

- $\eta$: Learning Rate

If using sum of square errors:

$$\frac{\partial}{\partial w_i} \text{Error}(E) = \sum_{e} -2 \cdot (Y(e) - \hat{Y}(e)) \cdot X_i(e)$$
$\frac{\partial E}{\partial w_i}(E) = \sum_{e \in E} -2 \cdot (Y(e) - \hat{Y}(e)) \cdot X_i(e) \cdot \delta(e)$

**Iterative Gradient Descent:**
- Compute for each $w_i$:
  $\frac{\partial E}{\partial w_i}(E) = \sum_{e \in E} -2 \cdot \delta(e) \cdot X_i(e)$
- Update each $w_i$ by $w_i := w_i - \eta \cdot \frac{\partial E}{\partial w_i}(E)$
- Repeat until stop criteria reached.
- $\delta(e)$'s become small
- $w_i$ become small.

**Variant: Incremental Gradient Descent:**
- Update $w_i$ after each example:
  $w_i := w_i - \eta \cdot \delta(e) \cdot X_i(e)$
- $\eta$ approaches solution faster
- Does not converge.

- **Stochastic:** Choose examples at random
- **Batched Gradient descent:** Update of some number $n\epsilon$ of examples
  - $\delta$ starts with $n\epsilon=1$ afterwords $n\epsilon$ increases until $n\epsilon=|E|$
  - Vary the size of $\eta$ start with large value, decrease latter

**Procedure LinearLearner (E, $\eta$):**
- $E$: set of examples, each of the form $<X_1, X_2, X_3, ..., Y>$
- $\eta$: learning rate.

```plaintext
initialize $w_0,...,w_n$ randomly
REPEAT
  FOR EACH example $e$ in $E$ DO
    $Y_{\text{cap}} := \sum_i w_i \cdot X_i(e)$
    $\delta := Y(e) - Y_{\text{cap}}$
    $\text{update} := \eta \cdot \delta$
    FOR EACH $w_i$ DO
      $w_i := w_i + \text{update} \cdot X_i(e)$
    END FOR
  END FOR
UNTIL some stop criteria is true
RETURN $w_0,...,w_n$
```

From regression to classification.
Examples

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>

\( e_9 \quad T \quad F \quad F \quad T \quad ? \)

Regression: \( \hat{Y} = \sum_{i=0}^{n} x_i w_i \quad \hat{Y} \in \mathbb{R} \)

Squash function: \( \hat{Y} = f\left( \sum_{i=0}^{n} x_i w_i \right) \) if squashes into \([0, 1]\)

One option for \( f \): Step function \( f(x) = \begin{cases} 1 & \text{if} \ x > 0 \\ 0 & \text{otherwise} \end{cases} \)

Second option for \( f \): Sigmoid function \( f(x) = \frac{1}{1 + e^{-x}} = \text{sig}(x) \)

\( \text{sig}'(x) = \text{sig}(x)(1 - \text{sig}(x)) \)

Form of linear classification: "Logistic Regression" \( \hat{Y}(e) = \text{sig}\left( \sum_{i=0}^{n} w_i X_i(e) \right) \)

\( \text{Error}(E) = \sum_{e \in E} (Y(e) - \hat{Y}(e))^2 \)

\( \frac{\partial}{\partial w_i} \text{Error}(E) = \frac{1}{\sum_{e \in E} (Y(e) - \text{sig}(\sum_{i=0}^{n} w_i X_i(e)))^2} \)
\[
\frac{\partial \text{Error}(E)}{\partial w_i} = \sum_{e \in E} \left( Y(e) - \sigma\left( \sum_{r=0}^{n} w_r X_r(e) \right) \right)^2
\]
\[
= \sum_{e \in E} 2 \cdot \left( \frac{\partial}{\partial w_i} \left( Y(e) - \sigma\left( \sum_{r=0}^{n} w_r X_r(e) \right) \right) \right) \\
= \sum_{e \in E} 2 \cdot \left( Y(e) - \sigma\left( \sum_{r=0}^{n} w_r X_r(e) \right) \right) \cdot \frac{\partial}{\partial w_i} \left( \sigma\left( \sum_{r=0}^{n} w_r X_r(e) \right) \right) \\
\quad \cdot \left( 1 - \sigma\left( \sum_{r=0}^{n} w_r X_r(e) \right) \right) \cdot \frac{\partial}{\partial w_i} \sum_{r=0}^{n} w_r X_r(e) \\
\quad \cdot X_i(e)
\]

\[
p_e = \sigma\left( \sum_{r=0}^{n} w_r X_r(e) \right) = \hat{Y}(e)
\]
\[
\delta(e) = Y(e) - \hat{Y}(e)
\]

\[
\frac{\partial \text{Error}(E)}{\partial w_i} = \sum_{e \in E} -2 \cdot \delta(e) \cdot p \cdot (1-p) \cdot X_i(e)
\]

PROCEDURE LogisticRegression ( E, eta )

- E : set of examples, each of the form <X1, X2, X3, ..., Y>
  - Y is in \{0,1\}
- eta : learning rate.

initialize w0,...,wn randomly
REPEAT
  FOR EACH example e in E DO
    p := \sigma( \sum_{i} wi * Xi(e) )
    delta := Y(e) - p
    update := eta * delta * p * (1 - p)
    FOR EACH wi DO
      wi := wi + update * Xi(e)
  UNTIL some stop criteria is true
RETURN w0,...,wn

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>
Example 2:
\[
\langle x_1, x_2, x_3 \ldots x_n \rangle \quad \quad Y = \{ \text{cat, dog, donut} \} \\
Y_{\text{cat}} = \{T, F\} \\
Y_{\text{dog}} = \{T, F\} \\
Y_{\text{donut}} = \{T, F\}
\]

indicator variables

Step function:
\[
f(y) = \begin{cases} 
1 & \text{if } x > 0 \\
0 & \text{otherwise}
\end{cases}
\]

threshold.

Frank Rosenblatt: "Perceptron"

Circuit: \( f(\sum w_i x_i) \)

'58 "(Perceptrons) are the embryo of an electronic computer that will be able to talk, see, write, translate languages and reproduce itself and be conscious of its existence."

'69 = Marvin Minsky
Seymour Papert "Perceptron"

XOR function

= first AI winter.

\( \sum w_i x_i \) is a line (hyperplane)

Data non-linearly separable

Why just \( \sum w_i x_i \) ???
a sufficiently complicated model can fit any data.
a sufficiently complicated model can fit any data.

Ockham's Razor: when faced with more than one explanation prefer the simplest one.

modify algorithm with a "regularizer":
a component that rewards simplicity and punishes complexity

\[
\hat{y}(e) = \sigma\left(\sum_{i=0}^{n} w_i x_i\right) \quad \text{Error}(E) = \sum_{e} (y(e) - \hat{y}(e))^2 + \lambda \left( \sum_{i=0}^{n} |w_i| \right)
\]

regularization

\[
\frac{\partial}{\partial w} = (y(e) - \hat{y}(e))^2 \quad \frac{\partial}{\partial w} f(g(x)) = f'(g(x)) \cdot \frac{\partial}{\partial w} g(x)
\]

- Errors & Pitfalls

  - Bias: Learner to find imperfect model
    - representation model not good enough
    - Search is not good enough
  
  - Data is not good
    - Lack data
    - Data is noisy
  
  - Overfitting:
    Learner specializes to the training data.
One technique to avoid over-fitting: **Cross-Validation**

Let's imagine we have a dataset. We can divide this dataset into *k* folds, for example, into 5 folds. Then, we pick 1 fold to test and train on the remaining folds. We repeat this process *k* times, using all the data to train.

**Diagram:**
- **X-axis:** Training Steps
- **Y-axis:** Training Data
- **Red line:** Test data
- **Green line:** Training data