

# Computing Lab

---

Song Liu (song.liu@bristol.ac.uk)



Mark I perceptron machine

# Perceptron

---

- Perceptron is a binary classifier.
- Perceptron uses a simplified model for biological neurons.
- Works on perceptron in the 1950s gave birth to many machine learning concepts that are well known today
  - such as artificial neural networks.
- You can read the full history of perceptron classifier here:
  - <https://en.wikipedia.org/wiki/Perceptron#History>

# Simplified Perceptron (Simplitron)

- Let us implement a simplified perceptron.
- Recall the desired behavior of a prediction func.  $f(\mathbf{x}; \mathbf{w})$ :
- $f(\mathbf{x}_i; \mathbf{w}) \geq 0 \ \forall i, y_i = +1$
- $f(\mathbf{x}_i; \mathbf{w}) \leq 0 \ \forall i, y_i = -1$
- Or equivalently  $y_i \cdot f(\mathbf{x}_i; \mathbf{w}) \geq 0$
- The algorithm is then simply:
- Initialize  $\mathbf{w}$  by random
- Loop over all training points:
  - If  $y \cdot f(\mathbf{x}; \mathbf{w}) \leq 0$ , update  $\mathbf{w}$  to  $\mathbf{w}'$  such that
$$y \cdot f(\mathbf{x}; \mathbf{w}') > y \cdot f(\mathbf{x}; \mathbf{w})$$

# Simpliton Algorithm

Suppose  $f(\mathbf{x}; \mathbf{w}) = \langle \mathbf{w}, \tilde{\mathbf{x}} \rangle, \tilde{\mathbf{x}} = [\mathbf{x}, 1]$

Notice the fact:

$$y_i \cdot f(\mathbf{x}_i; \mathbf{w} + y_i \cdot \tilde{\mathbf{x}}_i) \geq y_i \cdot f(\mathbf{x}_i; \mathbf{w})$$

Why? Prove this.

- Initialize  $\mathbf{w}$  by random
- For iter = 1 to max\_iteration
  - Set step size  $\eta = \frac{\eta_0}{\text{iter}}$
  - For  $i \in D$ 
    - If  $y_i \cdot f(\mathbf{x}_i; \mathbf{w}) \leq 0$
    - $\mathbf{w}' = \mathbf{w} + \eta \cdot y_i \cdot \tilde{\mathbf{x}}_i$

# Test

---

- Construct test cases and see how your algorithm works.
- On “reasonable datasets”, it should work like this:
  - <https://github.com/anewgithubname/anewgithubname.github.io/blob/gh-pages/perceptron.mp4?raw=true>
- Try different choices of  $\eta_0$ , see how it affect the performance of simplitron.
- Try  $\frac{\eta_0}{\sqrt{\text{iter}}}$  and  $\frac{\eta_0}{\text{iter}^2}$
- Complitron: Let  $f(\mathbf{x}; \mathbf{w})$  be a **generalized linear model**, does our algorithm still work?

# Formal Names

---

- The algorithm we implement, loop-update over all data points, is a version of **Stochastic Gradient Descent (SGD)**.
- The generalized linear model is also called **one-layer neural network model**.
- We just trained a one-layer neural network!