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. $f(\boldsymbol{x}; \boldsymbol{w})$:
- $f(\boldsymbol{x}_i; \boldsymbol{w}) \geq 0 \; \forall i, y_i = +1$
- $f(\boldsymbol{x}_i; \boldsymbol{w}) \leq 0 \; \forall i, y_i = -1$
- Or equivalently $y_i \cdot f(\boldsymbol{x}_i; \boldsymbol{w}) \geq 0$
- The algorithm is then simply:
- Initialize $\boldsymbol{w}$ by random
- Loop over all training points:
  - If $y \cdot f(\boldsymbol{x}; \boldsymbol{w}) \leq 0$, update $\boldsymbol{w}$ to $\boldsymbol{w}'$ such that

$$y \cdot f(\boldsymbol{x}; \boldsymbol{w}') > y \cdot f(\boldsymbol{x}; \boldsymbol{w})$$

Simplitron Algorithm

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

Notice the fact:

$$y_i \cdot f(\boldsymbol{x}_i; \boldsymbol{w} + y_i \cdot \widetilde{\boldsymbol{x}}_i) \geq y_i \cdot f(\boldsymbol{x}_i; \boldsymbol{w})$$

Why? Prove this.

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

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 η_0, see how it affect the performance of simplitron.
- Try $\frac{\eta_0}{\sqrt{\mathrm{iter}}}$ and $\frac{\eta_0}{\mathrm{iter}^2}$

- Complitron: Let $f(\boldsymbol{x}; \boldsymbol{w})$ 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!