Log-linear perceptron learning

CS 585, Fall 2017

Introduction to Natural Language Processing

http://people.cs.umass.edu/~brenocon/inlp2017

Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts Amherst

NB as log-linear model

$$P(\text{spam} \mid D) = \frac{1}{Z} P(\text{spam}) \prod_{t=1}^{\text{len}(D)} P(w_t \mid \text{spam})$$

$$P(\text{spam} \mid D) = \frac{1}{Z} P(\text{spam}) \prod_{w \in \mathcal{V}} P(w \mid \text{spam})^{x_w}$$

$$\log P(\text{spam} \mid D) = \log P(\text{spam}) + \sum_{w \in \mathcal{V}} x_w \log P(w \mid \text{spam}) - \log Z$$

Log-linear models (NB, LogReg, HMM, CRF...)

- x: Text Data
- y: Proposed class or sequence
- θ: Feature weights (model parameters)
- f(x,y): Feature extractor, produces feature vector

$$p(y|x) = \frac{1}{Z} \exp \underbrace{\left(\theta^\mathsf{T} f(x,y)\right)}_{G(y)}$$

Decision rule: $$\arg \max_{y^* \in outputs(x)} G(y^*)$$

Log-linear classifier

$\theta_{BIAS,+}$ =
$\theta_{happy,+}$ =
$\theta_{dog,+}$ =

$\theta_{BIAS,-}$ =
$\theta_{happy,-}$ =
$\theta_{dog,-}$ =

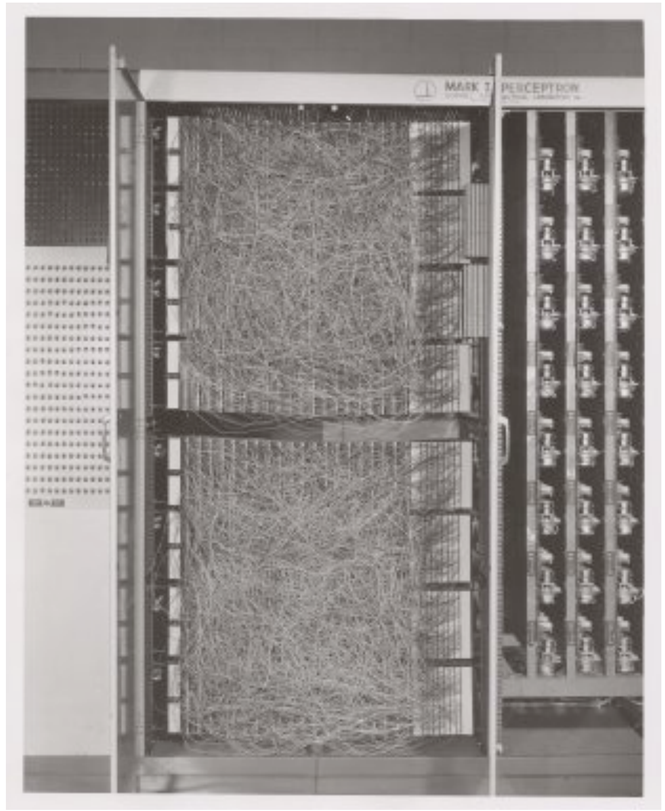
- P(+ | “happy dog”) =
- P(– | “happy dog”) =

The Perceptron Algorithm

- Perceptron is not a model:
  it is a learning algorithm
  - Rosenblatt 1957

- Insanely simple algorithm
  - Iterate through dataset.
    Predict.
    Update weights to fix prediction errors.

- Can be used for classification OR structured prediction
  - *structured perceptron*
- Discriminative learning algorithm for *any* log-linear model (our view in this course)



The Mark I Perceptron machine was the first implementation of the perceptron algorithm. The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image. The main visible feature is a patchboard that allowed experimentation with different combinations of input features. To the right of that are arrays ofpotentiometers that implemented the adaptive weights.

Binary perceptron

- For ~10 iterations
  - For each (x,y) in dataset
    - PREDICT

$$y^* = POS \text{ if } \theta^\mathsf{T} x \geq 0$$
$$= NEG \text{ if } \theta^\mathsf{T} x < 0$$

    - IF y=y*, do nothing
    - ELSE update weights

$$\theta := \theta + r\ x$$
$$\theta := \theta - r\ x$$

if POS misclassified as NEG:
let's make it more positive-y next time around

if NEG misclassified as POS:
let's make it more negative-y next time

learning rate constant
e.g. r=1

Structured/multiclass Perceptron

(for any log-linear model)

- For ~10 iterations
  - For each (x,y) in dataset
    - PREDICT

$$y^* = \arg\max_{y'} \theta^\mathsf{T} f(x, y')$$

    - IF y=y*, do nothing
    - ELSE update weights

$$\theta := \theta + r[f(x, y) - f(x, y^*)]$$

learning rate constant
e.g. r=1

Features for
TRUE label

Features for
PREDICTED label

Perceptron notes/issues

- Issue: does it converge? (generally no)
  - Solution: the *averaged* perceptron
- Can you regularize it? No... just averaging...

- By the way, there's also *likelihood* training out there (gradient ascent on the log-likelihood function)
  - structperc is easier to implement/conceptualize and performs similarly in practice