Linear Models Continued: Perceptron & Logistic Regression

CMSC 723 / LING 723 / INST 725

Marine Carpuat

Slides credit: Graham Neubig, Jacob Eisenstein

Linear Models for Classification

Feature function representation

$$\hat{y} = \arg\max_{y} \boldsymbol{\theta}^{\mathsf{T}} \mathbf{f}(\mathbf{x}, y)$$

Weights

Naïve Bayes recap

- Define $\mathrm{p}(\boldsymbol{x}, \boldsymbol{y})$ via a *generative model*
- Prediction: $\hat{y} = \arg\max_y \mathrm{p}(\boldsymbol{x}_i, y)$
- Learning:

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} \mathrm{p}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$$

$$\mathrm{p}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = \prod_i \mathrm{p}(\boldsymbol{x}_i, y_i; \boldsymbol{\theta}) = \prod_i \mathrm{p}(\boldsymbol{x}_i | y_i) \mathrm{p}(y_i)$$

$$\phi_{y,j} = \frac{\sum_{i:Y_i=y} x_{ij}}{\sum_{i:Y_i=y} \sum_j x_{ij}}$$

$$\mu_y = \frac{\text{count}(Y = y)}{N}$$

This gives the maximum likelihood estimator (MLE; same as relative frequency estimator)

The Perceptron

The perceptron

- A linear model for classification

- An algorithm to learn feature weights given labeled data
    - online algorithm
    - error-driven

Multiclass perceptron

$$\hat{y} = \arg\max_{y} \boldsymbol{\theta}^\top \mathbf{f}(\mathbf{x}, y)$$

Algorithm 1 Perceptron learning algorithm

procedure PERCEPTRON($\boldsymbol{x}_{1:N}, y_{1:N}$)
  repeat
    Select an instance i
    $\hat{y} \leftarrow \arg\max_y \boldsymbol{\theta}_t^\top \boldsymbol{f}(\boldsymbol{x}_i, y)$
    if $\hat{y} \neq y_i$ **then**
      $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \boldsymbol{f}(\boldsymbol{x}_i, y_i) - \boldsymbol{f}(\boldsymbol{x}_i, \hat{y})$
    else
      do nothing
  until tired

Understanding the perceptron

- What's the impact of the update rule on parameters?

- The perceptron algorithm will converge if the training data is **linearly separable**
    - Proof: see "A Course In Machine Learning" Ch.4

- Practical issues
  - How to initalize?
  - When to stop?
  - How to order training examples?

When to stop?

- One technique
  - When the accuracy on held out data starts to decrease
  - Early stopping

Requires splitting data into 3 sets: training/development/test

ML fundamentals aside: overfitting/underfitting/generalization

Training error is not sufficient

- We care about **generalization** to new examples

- A classifier can classify training data perfectly, yet classify new examples incorrectly
  - Because training examples are only a sample of data distribution
    - a feature might correlate with class by coincidence
  - Because training examples could be noisy
    - e.g., accident in labeling

Overfitting

- Consider a model θ and its:
  - Error rate over training data $error_{train}(\theta)$
  - True error rate over all data $error_{true}(\theta)$

- We say h overfits the training data if

$$error_{train}(\theta) < error_{true}(\theta)$$

Evaluating on test data

- Problem: we don’t know $error_{true}(\theta)$!

- Solution:
  - we set aside a test set
    - some examples that will be used for evaluation
  - we don’t look at them during training!
  - after learning a classifier θ, we calculate

$$error_{test}(\theta)$$

Overfitting

- Another way of putting it

- A classifier θ is said to overfit the training data, if there is another hypothesis θ', such that
  - θ has a smaller error than θ' on the training data
  - but θ has larger error on the test data than θ'.

Underfitting/Overfitting

- Underfitting
  - Learning algorithm had the opportunity to learn more from training data, but didn't
- Overfitting
  - Learning algorithm paid too much attention to idiosyncracies of the training data; the resulting classifier doesn't generalize

Back to the Perceptron

Averaged Perceptron improves generalization

Algorithm 2 Averaged perceptron learning algorithm

procedure AVG-PERCEPTRON($\boldsymbol{x}_{1:N}, y_{1:N}$)
    repeat
        Select an instance i
        $\hat{y} \leftarrow \arg\max_y \boldsymbol{\theta}_t^\top \boldsymbol{f}(\boldsymbol{x}_i, y)$
        if $\hat{y} \neq y_i$ **then**
            $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \boldsymbol{f}(\boldsymbol{x}_i, y_i) - \boldsymbol{f}(\boldsymbol{x}_i, \hat{y})$
            $\boldsymbol{m} \leftarrow \boldsymbol{m} + \boldsymbol{\theta}_{t+1}$
        else
            do nothing
    until tired
    $\overline{\boldsymbol{\theta}} \leftarrow \frac{1}{t}\boldsymbol{m}$

What objective/loss does the perceptron optimize?

- Zero-one loss function

$$\ell_{\text{perceptron}}(\boldsymbol{\theta}; \boldsymbol{x}_i, y_i) = \begin{cases} 0, & y_i = \mathbf{arg}\max_y \boldsymbol{\theta}^\top \boldsymbol{f}(x_i, y) \\ 1, & \text{otherwise} \end{cases}$$

- What are the pros and cons compared to Naïve Bayes loss?

Logistic Regression

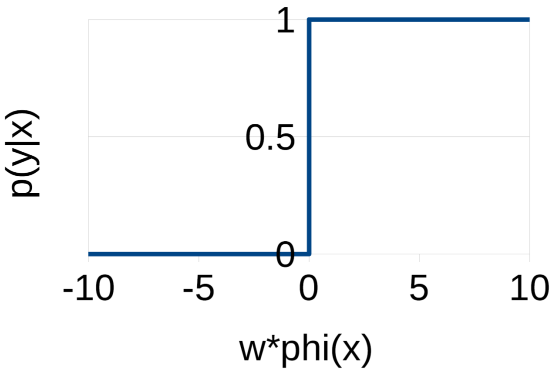
Perceptron & Probabilities

- What if we want a probability p(y|x)?

- The perceptron gives us a prediction y
  - Let's illustrate this with binary classification

In other words:

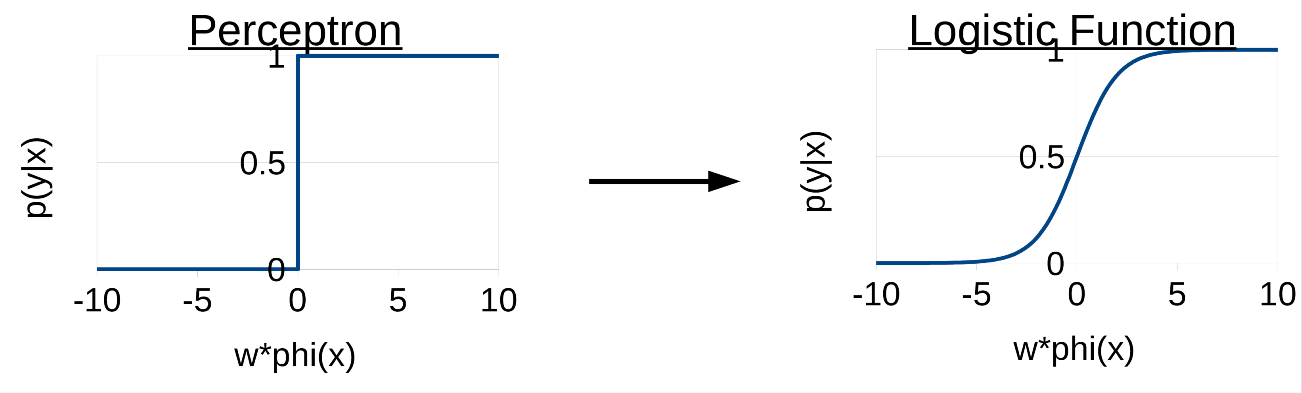
$$P(y=1|x)=1 \text{ if } \mathbf{w}\cdot\varphi(x)\geq 0$$
$$P(y=1|x)=0 \text{ if } \mathbf{w}\cdot\varphi(x)<0$$



Illustrations: Graham Neubig

The logistic function

$$P(y=1|x)=\frac{e^{\mathbf{w}\cdot\boldsymbol{\varphi}(x)}}{1+e^{\mathbf{w}\cdot\boldsymbol{\varphi}(x)}}$$



- “Softer” function than in perceptron
- Can account for uncertainty
- Differentiable

Logistic regression: how to train?

- Train based on **conditional likelihood**
- Find parameters w that maximize conditional likelihood of all answers y_i given examples x_i

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_i P(y_i | x_i ; \mathbf{w})$$

Stochastic gradient ascent (or descent)

- Online training algorithm for logistic regression
  - and other probabilistic models

```
create map w
for I iterations
    for each labeled pair x, y in the data
        w += α * dP(y|x)/dw
```

- Update weights for every training example
- Move in direction given by gradient
- Size of update step scaled by learning rate

What you should know

- Standard supervised learning set-up for text classification
  - Difference between train vs. test data
  - How to evaluate
- 3 examples of supervised linear classifiers
  - Naïve Bayes, Perceptron, Logistic Regression
  - Learning as optimization: what is the objective function optimized?
  - Difference between generative vs. discriminative classifiers
  - Smoothing, regularization
  - Overfitting, underfitting

An **online** learning algorithm

```
create map w
for I iterations
    for each labeled pair x, y in the data
        phi = CREATE_FEATURES(x)
        y' = PREDICT_ONE(w, phi)
        if y' != y
            UPDATE_WEIGHTS(w, phi, y)
```

Perceptron weight update

$$w \leftarrow w + y\,\varphi(x)$$

- If y = 1, increase the weights for features in $\varphi(x)$
- If y = -1, decrease the weights for features in $\varphi(x)$