CSE 446
Perceptron Learning
Winter 2012

Dan Weld

Some slides from Carlos Guestrin, Luke Zettlemoyer

Machine Learning

Supervised Learning | Reinforcement Learning | Unsupervised Learning

Parametric | Non-parametric

Y Continuous | Y Discrete

Gaussians
Learned in closed form

Linear Functions
1. Learned in closed form
2. Using gradient descent

Decision Trees
Greedy search; pruning

Probability of Class | Features
1. Learn P(Y), P(X|Y); apply Bayes
2. Learn P(Y|X) w/ gradient descent

Non-probabilistic Linear Classifier
Perceptron – w/ gradient descent

Hypothesis Expressiveness

Linear
- ***Naïve Bayes***
- ***Logistic Regression***
- ***Perceptron***
- Support Vector Machines

Nonlinear
- ***Decision Trees***
- Neural Networks
- ***Ensembles***
- ***Kernel Methods***
- ***Nearest Neighbor***
- Graphical Models

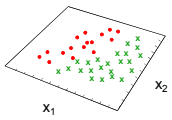
Logistic Regression

- **Want to Learn**: h:**X** ↦ Y
  - **X** – features
  - Y – target classes
- **Probabilistic Discriminative Classifier**
  - Assume some **functional form for P(Y|X)**
    - **Logistic Function**
    - **Accepts both discrete & continuous features**
  - Estimate parameters of P(Y|X) directly from training data
  - This is the ***'discriminative'* model**
    - Directly learn P(Y|X)
    - But **cannot generate a sample of the data**,
    - No way to compute P(X)

Earthquake or Nuclear Test?

$$P(Y=1|X=<X_1,...X_n>) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

implies



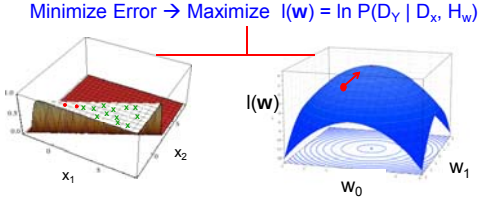
$$\ln \frac{P(Y=0|X)}{P(Y=1|X)} = w_0 + \sum_i w_i X_i$$

linear classification rule!

Logistic w/ Initial Weights

w_0=20 w_1= -5 w_2 =10

Loss(H_w) = Error(H_w, data)
Minimize Error → Maximize l(**w**) = ln P(D_Y | D_x, H_w)



Update rule:

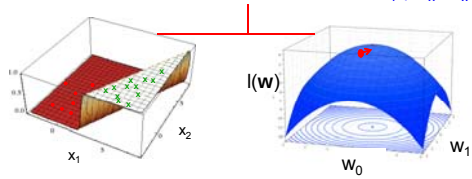
$$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})}{\partial w_i}$$

Gradient Ascent

w_0=40 w_1= -10 w_2 =5

Maximize l(**w**) = ln P(D_Y | D_x, H_w)



Update rule:

$$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})}{\partial w_i}$$

7

M_CLE *vs.* M_CAP

- M_CLE – Uniform Prior on Weights

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ln \left[\prod_{j=1}^{N} P(y^j \mid \mathbf{x}^j, \mathbf{w}) \right]$$

$$\boxed{w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w})]}$$

- M_CAP – Regularization - Gaussian Prior on Weights

$$p(\mathbf{w}) = \prod_i \frac{1}{\kappa\sqrt{2\pi}} \; e^{\frac{-w_i^2}{2\kappa^2}}$$

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ln \left[p(\mathbf{w}) \prod_{j=1}^{N} P(y^j \mid \mathbf{x}^j, \mathbf{w}) \right]$$

$$\boxed{w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w})] \right\}}$$

Naïve Bayes *vs.* Logistic Regression

Learning: h:**X** ↦ Y

X – features
Y – target classes

Generative

- Assume functional form for
  - P(X|Y) **assume cond indep**
  - P(Y)
  - Est params from train data
- Gaussian NB for cont features
- Bayes rule to calc. P(Y|X= x)
  - P(Y | **X**) ∝ P(**X** | Y) P(Y)
- **Indirect** computation
  - Can also generate a sample of the data

Discriminative

- Assume functional form for
  - P(Y|X) **no assumptions**
  - Est params from training data
- Handles discrete & cont features
- Directly calculate P(Y|X=x)
  - Can't generate data sample

9

Gaussian Naïve Bayes *vs.* Logistic Regression

Learning: h:**X** ↦ Y

X – ***Real-valued*** features
Y – target classes

Generative

- Assume functional form for
  - P(X|Y) assume X_i cond indep given Y
  - P(Y)
  - Est params from train data
- Gaussian NB for continuous features
  - model P(X_i | Y = y_k) as **Gaussian** N(μ_{ik},σ_i)
  - model P(Y) as **Bernoulli** (θ,1-θ)
- Bayes rule to calc. P(Y|X= x)
  - P(Y | **X**) ∝ P(**X** | Y) P(Y)

What can we say about the form of P(Y=1 | …X_i…)?

$$\boxed{\frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}}$$

Cool!!!!

10

Derive form for P(Y|X) for continuous X_i

$$P(Y=1|X) = \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1) + P(Y=0)P(X|Y=0)}$$

$$= \frac{1}{1 + \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}}$$

up to now, all arithmetic

$$= \frac{1}{1 + \exp(\ln \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)})}$$

only for Naïve Bayes models

$$= \frac{1}{1 + \exp(\ (\ln \frac{1-\theta}{\theta}) + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)})}$$

Looks like a setting for w_0?

Can we solve for w_i ?

- Yes, but only in Gaussian case

Ratio of class-conditional probabilities

$$\ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)}$$

$$P(X_i = x \mid Y = y_k) = \frac{1}{\sigma_i\sqrt{2\pi}} \; e^{\frac{-(x-\mu_{ik})^2}{2\sigma_i^2}}$$

$$= \ln \left[\frac{\frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{(x_i-\mu_{i0})^2}{2\sigma_i^2}}}{\frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{(x_i-\mu_{i1})^2}{2\sigma_i^2}}} \right]$$

$$= -\frac{(x_i-\mu_{i0})^2}{2\sigma_i^2} + \frac{(x_i-\mu_{i1})^2}{2\sigma_i^2}$$

$$\ldots$$

$$= \frac{\mu_{i0}+\mu_{i1}}{\sigma_i^2} \boxed{x_i} + \frac{\mu_{i0}^2+\mu_{i1}^2}{2\sigma_i^2}$$

Linear function! Coefficents expressed with original Gaussian parameters!

Derive form for P(Y|X) for continuous X_i

$$P(Y=1|X) = \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1)+P(Y=0)P(X|Y=0)}$$

$$= \frac{1}{1+\exp(\ (\ln\frac{1-\theta}{\theta}) + \boxed{\sum_i \ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)}})}$$

$$\boxed{\sum_i \left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i + \frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right)}$$

$$P(Y=1|X) = \frac{1}{1+\exp(w_0+\sum_{i=1}^n w_i X_i)}$$

$$\boxed{w_0 = \ln\frac{1-\theta}{\theta} + \frac{\mu_{i0}^2+\mu_{i1}^2}{2\sigma_i^2}}$$

$$\boxed{w_i = \frac{\mu_{i0}+\mu_{i1}}{\sigma_i^2}}$$

Gaussian Naïve Bayes vs. Logistic Regression

Set of Gaussian Naïve Bayes parameters (feature variance independent of class label) ⟷ **Set of Logistic Regression parameters**

Can go both ways, we only did one way

- **Representation equivalence**
  - **But only in a special case!!!** (GNB with class-independent variances)
- **But what's the difference???**
- **LR makes no assumptions about P(X|Y) in learning!!!**
- **Loss function!!!**
  - Optimize different functions ! Obtain different solutions

Naïve Bayes vs. Logistic Regression

Consider Y boolean, X=<X_1 … X_n> continuous

Number of parameters:
- Naïve Bayes: 4n+1
- Logistic Regression: n+1

$P(Y) = \theta$
$P(X|Y) = N(\mu, \sigma)$
$P(X|\neg Y) = N(\mu', \sigma')$

Estimation method:
- Naïve Bayes parameter estimates are uncoupled
- Logistic Regression parameter estimates are coupled

Naïve Bayes vs. Logistic Regression

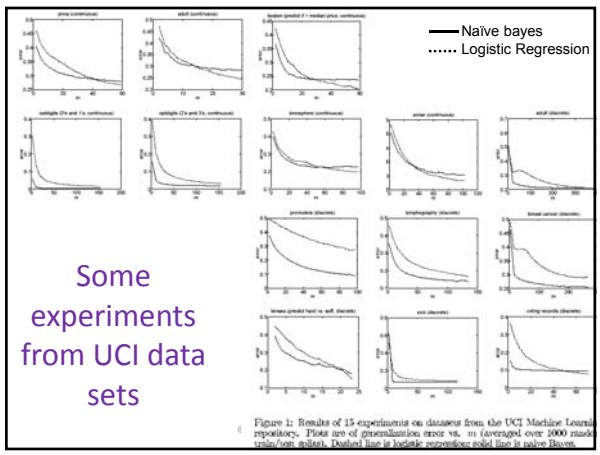
[Ng & Jordan, 2002]

- Generative vs. Discriminative classifiers
- Asymptotic comparison
  (# training examples → infinity)
  - when model correct
    - GNB (with class independent variances) and LR produce identical classifiers
  - when model incorrect
    - LR is less biased – does not assume conditional independence
      - **therefore LR expected to outperform GNB**

Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

- Generative vs. Discriminative classifiers
- Non-asymptotic analysis
  - convergence rate of parameter estimates,
    (n = # of attributes in X)
    - Size of training data to get close to infinite data solution
    - Naïve Bayes needs ***O*(log n)** samples
    - Logistic Regression needs ***O*(n)** samples
  - GNB converges more quickly to its ***(perhaps less helpful)*** asymptotic estimates



Some experiments from UCI data sets

Figure 1: Results of 15 experiments on datasets from the UCI Machine Learni repository. Plots are of generalization error vs. n (averaged over 1000 rand train/test splits). Dashed line is logistic regression; solid line is naive Bayes.

What you should know about Logistic Regression (LR)

- Gaussian Naïve Bayes with class-independent variances representationally equivalent to LR
  - Solution differs because of objective (loss) function
- In general, NB and LR make different assumptions
  - NB: Features independent given class ! assumption on P(**X**|Y)
  - LR: Functional form of P(Y|**X**), no assumption on P(**X**|Y)
- LR is a linear classifier
  - decision rule is a hyperplane
- LR optimized by conditional likelihood
  - no closed-form solution
  - concave ! global optimum with gradient ascent
  - Maximum conditional a posteriori corresponds to regularization
- Convergence rates
  - GNB (usually) needs less data
  - LR (usually) gets to better solutions in the limit

Perceptrons

Who needs probabilities?

- Previously: model data with distributions
- Joint: P(X,Y)
  - e.g. Naïve Bayes
- Conditional: P(Y|X)
  - e.g. Logistic Regression
- But wait, why probabilities?
- Lets try to be error-driven!

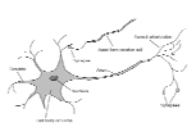
mpg	cylinders	displacemen	horsepower	weight	acceleration	modelyear	make
good	4	97	75	2265	18.2	77	asia
bad	6	199	90	2648	15	70	ames
bad	4	121	110	2600	12.8	77	euro
bad	8	350	175	4100	13	73	ames
bad	6	198	95	3102	16.5	74	ames
bad	4	108	94	2379	16.5	73	asia
bad	4	113	95	2228	14	71	asia
bad	8	302	139	3570	12.8	78	ames
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
good	4	120	79	2625	18.6	82	ames
bad	8	455	225	4425	10	70	ames
good	4	107	86	2464	15.5	76	euro
bad	5	131	103	2830	15.9	78	euro

Generative vs. Discriminative

- Generative classifiers:
  - E.g. naïve Bayes
  - A joint probability model with evidence variables
  - Query model for causes given evidence
- Discriminative classifiers:
  - No generative model, no Bayes rule, often no probabilities at all!
  - Try to predict the label Y directly from X
  - Robust, accurate with varied features
  - Loosely: mistake driven rather than model driven

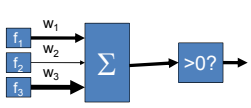
Linear Classifiers

- Inputs are feature values
- Each feature has a weight
- Sum is the activation



$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

- If the activation is:
  - Positive, output *class 1*
  - Negative, output *class 2*



Example: Spam

- Imagine 3 features (spam is "positive" class):
  - free (number of occurrences of "free")
  - money (occurrences of "money")
  - BIAS (intercept, always has value 1)

$$w \cdot f(x)$$

$$\sum_i w_i \cdot f_i(x)$$

x

"free money"

$f(x)$

```
BIAS  :  1
free  :  1
money :  1
...
```

w

```
BIAS  : -3
free  :  4
money :  2
...
```

$(1)(-3)$ +
$(1)(4)$ +
$(1)(2)$ +
$\ldots$
$= 3$

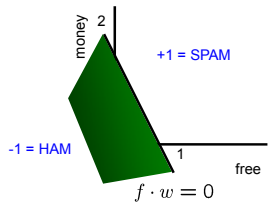
w.f(x) > 0 ➔ SPAM!!!

Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1

w

```
BIAS  : -3
free  :  4
money :  2
...
```



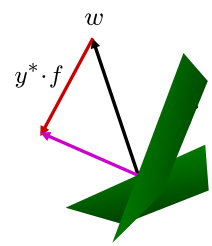
Binary Perceptron Algorithm

- Start with zero weights
- For each training instance (x,y*):
  - Classify with current weights

$$y = \begin{cases} +1 & \text{if } w \cdot f(x) \geq 0 \\ -1 & \text{if } w \cdot f(x) < 0 \end{cases}$$

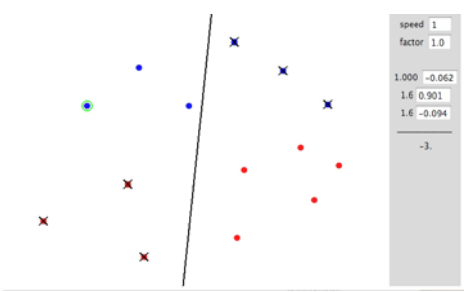
  - If correct (i.e., y=y*), no change!
  - If wrong: update

$$w = w + y^* \cdot f$$



Examples: Perceptron

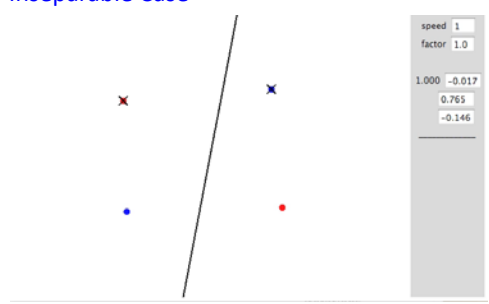
- Separable Case



http://isl.ira.uka.de/neuralNetCourse/2004/VL_11_5/Perceptron.html

Examples: Perceptron

- Inseparable Case



http://isl.ira.uka.de/neuralNetCourse/2004/VL_11_5/Perceptron.html

From Logistic Regression to the Perceptron: 2 easy steps!

- Logistic Regression: (in vector notation): y is {0,1}

$$w = w + \eta \sum_j [y_j^* - p(y_j^* | x_j, w)] f(x_j)$$

- Perceptron: y is {0,1}, y(x;w) is prediction given w

$$w = w + [y^* - y(x;w)] f(x)$$

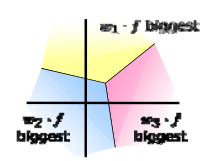
Differences?

•Drop the Σ_j over training examples: online vs. batch learning

•Drop the dist'n: probabilistic vs. error driven learning

Multiclass Decision Rule

- If we have more than two classes:
  - Have a weight vector for each class: w_y
  - Calculate an activation for each class



$$\text{activation}_w(x, y) = w_y \cdot f(x)$$

  - Highest activation wins

$$y = \arg\max_y \ (\text{activation}_w(x, y))$$

Example

"win the vote"

"win the election"

"win the game"

w_{SPORTS}

```
BIAS  :
win   :
game  :
vote  :
the   :
...
```

$w_{POLITICS}$

```
BIAS  :
win   :
game  :
vote  :
the   :
...
```

w_{TECH}

```
BIAS  :
win   :
game  :
vote  :
the   :
...
```

Example

"win the vote"

```
BIAS  : 1
win   : 1
game  : 0
vote  : 1
the   : 1
...
```

w_{SPORTS}

```
BIAS  : -2
win   :  4
game  :  4
vote  :  0
the   :  0
...
```

$w_{POLITICS}$

```
BIAS  : 1
win   : 2
game  : 0
vote  : 4
the   : 0
...
```

w_{TECH}

```
BIAS  : 2
win   : 0
game  : 2
vote  : 0
the   : 0
...
```

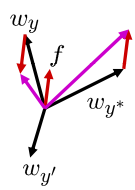
The Multi-class Perceptron Alg.

- Start with zero weights
- Iterate training examples
  - Classify with current weights

$$y = \arg\max_y \; w_y \cdot f(x)$$
$$= \arg\max_y \; \sum_i w_{y,i} \cdot f_i(x)$$

  - If correct, no change!
  - If wrong: lower score of wrong answer, raise score of right answer

$$w_y = w_y - f(x)$$
$$w_{y^*} = w_{y^*} + f(x)$$

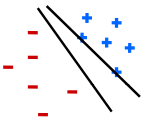


Properties of Perceptrons

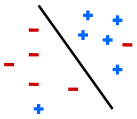
- Separability: some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

$$\text{mistakes} < \frac{k}{\delta^2}$$

Separable

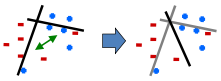


Non-Separable



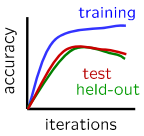
Problems with the Perceptron

- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



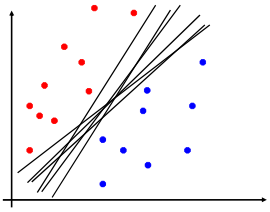
- Mediocre generalization: finds a "barely" separating solution

- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting



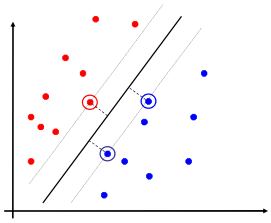
Linear Separators

- Which of these linear separators is optimal?



Support Vector Machines

- Maximizing the margin: good according to intuition, theory, practice
- SVMs find the separator with max margin

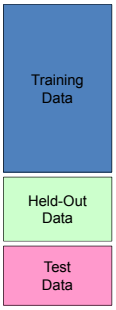


SVM

$$\min_{w} \frac{1}{2}||w||^2$$

$$\forall i, y \; w_{y^*} \cdot f(x_i) \geq w_y \cdot f(x_i) + 1$$

Three Views of Classification



- Naïve Bayes:
  - Parameters from data statistics
  - Parameters: probabilistic interpretation
  - Training: one pass through the data
- Logistic Regression:
  - Parameters from gradient ascent
  - Parameters: linear, probabilistic model, and discriminative
  - Training: one pass through the data per gradient step; regularization essential
- The Perceptron:
  - Parameters from reactions to mistakes
  - Parameters: discriminative interpretation
  - Training: go through the data until held-out accuracy maxes out