CSE 446
Logistic Regression
Perceptron Learning
Winter 2012

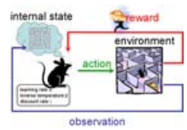
Dan Weld

Some slides from Carlos Guestrin, Luke Zettlemoyer

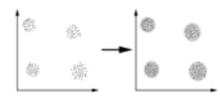
Machine Learning

Supervised Learning
Labeled Examples → Function

Reinforcement Learning
Experience + Rewards → Policy



Unsupervised Learning
Unlabeled Examples → Clusters



Machine Learning

Supervised Learning
Labeled Examples → Function
$<X_1,\ldots,X_n, Y>$

Parametric

Non-parametric

Machine Learning

Supervised Learning
Labeled Examples → Function
$<X_1,\ldots,X_n, Y>$

Parametric

Y Continuous

Y Discrete

Gaussians
Learned in closed form

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

Machine Learning

Supervised Learning
Labeled Examples → Function
$<X_1,\ldots,X_n, Y>$

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

Probabilistic Inference
(Making Predictions)

- After learning
  – (And also during learning)
- We need to
  – Determine probabilities of hypotheses
  – Use these hypotheses to make predictions
    - MLE
    - MAP
    - Bayesian Inference

Which Coin is Dan Using?

C_1 C_2 C_3

$P(H|C_1) = 0.1$ $P(H|C_2) = 0.5$ $P(H|C_3) = 0.9$

Prior Probabilities

C_1 C_2 C_3

$P(H|C_1) = 0.1$ $P(H|C_2) = 0.5$ $P(H|C_3) = 0.9$

$P(C_1) = 0.05$ $P(C_2) = 0.25$ $P(C_3) = 0.70$

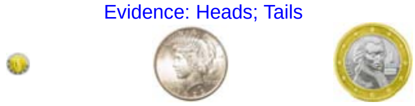
Forms of Inference

C_1 C_2 C_3

$P(H|C_1) = 0.1$ $P(H|C_2) = 0.5$ $P(H|C_3) = 0.9$

$P(C_1) = 0.05$ $P(C_2) = 0.25$ $P(C_3) = 0.70$

Evidence: Heads; Tails



$P(C_1|HT) = 0.035$ $P(C_2|HT) = 0.481$ $P(C_3|HT) = 0.485$

Predict Next Toss??

Bayesian Inference
68% heads

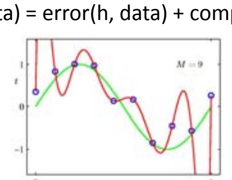
MAP
90% heads

Making predictions after (and during) learning from a distribution of hypotheses

	Prior	Hypothesis
Maximum Likelihood Estimate	Uniform	The most likely
Maximum A Posteriori Estimate	Any	The most likely
Bayesian Estimate	Any	Weighted combination

Themes:

- Learning as function approximation
  - What's a ***good*** approximation?
- Learning as optimization
  - Minimize ***loss*** over training data (test data)
  - Loss(h,data) = error(h, data) + complexity(h)



Machine Learning

Supervised Learning
Labeled Examples → Function
$\langle X_1, \ldots, X_n, Y \rangle$

Parametric

Y Continuous

Gaussians
Learned in closed form

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

Y Discrete

Decision Trees
Greedy search; pruning

Probability of class | features
1. Learn P(Y), P(X|Y) in closed form
   Then apply Bayes rule
2. Learn P(Y|X) w/ gradient descent

Generative *vs.* Discriminative Classifiers

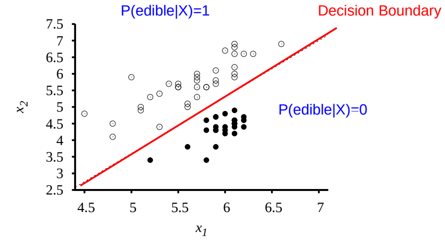
- **Want to Learn**: h:**X** $\mapsto$ Y
  - **X** – features
  - Y – target classes
- **Generative classifier**, e.g., Naïve Bayes:
  - Assume some **functional form for P(X|Y), P(Y)**
  - Estimate parameters of P(X|Y), P(Y) directly from training data
  - Use Bayes rule to calculate P(Y|X= x)
  - This is a **'*generative*' model**
    - **Indirect** computation of P(Y|X) through Bayes rule
    - As a result, **can also generate a sample of the data**, $P(X) = \sum_y P(y) P(X|y)$
- **Discriminative classifiers**, e.g., Logistic Regression:
  - Assume some **functional form for P(Y|X)**
  - Estimate parameters of P(Y|X) directly from training data
  - This is the **'*discriminative*' model**
    - Directly learn P(Y|X)
    - But **cannot obtain a sample of the data**, because P(X) is not available

$$P(Y \mid \mathbf{X}) \propto P(\mathbf{X} \mid Y)\, P(Y)$$

Are decision trees generative or discriminative?

Discriminative Classification

What Functional Form should we use for P(Y|X)?

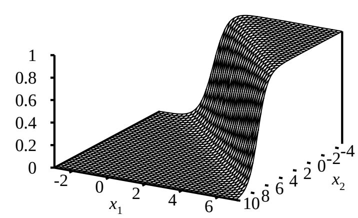


Want something ***smooth***…

Logistic Function in n Dimensions

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

Sigmoid applied to a linear function of the data:



Features can be discrete or continuous!

Very convenient!

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

implies

$$P(Y=0|X=<X_1,...X_n>) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

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

linear classification rule!

implies

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

Loss Functions ("Error" Term): Likelihood vs. Conditional Likelihood

Generative (Naïve Bayes) Loss function: **Data likelihood**

$$\begin{aligned}\ln P(\mathcal{D} \mid \mathbf{w}) &= \sum_{j=1}^N \ln P(\mathbf{x}^j, y^j \mid \mathbf{w}) \\ &= \sum_{j=1}^N \ln P(y^j \mid \mathbf{x}^j, \mathbf{w}) + \sum_{j=1}^N \ln P(\mathbf{x}^j \mid \mathbf{w})\end{aligned}$$

Discriminative (Logistic Regr.) Loss funct: **Conditional Data Likelihood**

$$\ln P(\mathcal{D}_Y \mid \mathcal{D}_{\mathbf{X}}, \mathbf{w}) = \sum_{j=1}^N \ln P(y^j \mid \mathbf{x}^j, \mathbf{w})$$

Discriminative models *can't* compute P($\mathbf{x}^j$|**w**)!

Or, … "They don't *waste effort* learning P(**X**)"

Focus only on P(Y|**X**) - all that matters for classification

Maximizing Conditional Log Likelihood

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

$$P(Y=1|X,W) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

$$\begin{aligned} l(\mathbf{w}) &\equiv \ln \prod_j P(y^j|\mathbf{x}^j,\mathbf{w}) \\ &= \sum_j y^j(w_0 + \sum_i^n w_i x_i^j) - \ln(1+exp(w_0+\sum_i^n w_i x_i^j)) \end{aligned}$$

Bad news: no closed-form solution to maximize *l*(**w**)

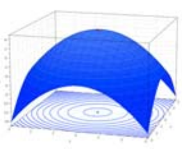
Good news: *l*(**w**) is concave function of **w**!

No local minima

Concave functions easy to optimize

Optimizing concave function – Gradient ascent

- Conditional likelihood for Logistic Regression is concave !



Gradient: $\nabla_{\mathbf{w}} l(\mathbf{w}) = [\frac{\partial l(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial l(\mathbf{w})}{\partial w_n}]'$

Learning rate, η>0

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 is simplest of optimization approaches
  - e.g., Conjugate gradient ascent much better (see reading)

Maximize Conditional Log Likelihood: Gradient ascent

$$P(Y=1|X,W) = \frac{exp(w_0 + \sum_i w_i X_i)}{1 + exp(w_0 + \sum_i w_i X_i)}$$

$$l(\mathbf{w}) = \sum_j y^j (w_0 + \sum_i^n w_i x_i^j) - \ln(1 + exp(w_0 + \sum_i^n w_i x_i^j))$$

$$\frac{\partial l(w)}{\partial w_i} = \sum_j \left[\frac{\partial}{\partial w} y^j (w_0 + \sum_i w_i x_i^j) - \frac{\partial}{\partial w} \ln \left(1 + \exp(w_0 + \sum_i w_i x_i^j) \right) \right]$$

$$= \sum_j \left[y^j x_i^j - \frac{x_i^j \exp(w_0 + \sum_i w_i x_i^j)}{1 + \exp(w_0 + \sum_i w_i x_i^j)} \right]$$

$$= \sum_j x_i^j \left[y^j - \frac{\exp(w_0 + \sum_i w_i x_i^j)}{1 + \exp(w_0 + \sum_i w_i x_i^j)} \right]$$

$$\boxed{\frac{\partial l(w)}{\partial w_i} = \sum_j x_i^j \left(y^j - P(Y^j = 1 | x^j, w) \right)}$$

Gradient Descent for LR

Gradient ascent algorithm: (learning rate $\eta > 0$)

do:

$$w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \sum_j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w})]$$

For i=1 to n: (iterate over weights)

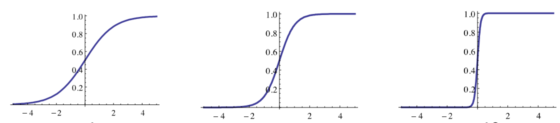
$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w})]$$

until "change" < ε

Loop over training examples!

Large parameters…

$$\frac{1}{1 + e^{-ax}}$$



a=1 a=5 a=10

- Maximum likelihood solution: prefers higher weights
  - higher likelihood of (properly classified) examples close to decision boundary
  - larger influence of corresponding features on decision
  - *can cause overfitting!!!*
- Regularization: penalize high weights

That's all M_CLE. How about M_CAP?

$$p(\mathbf{w} \mid Y, \mathbf{X}) \propto P(Y \mid \mathbf{X}, \mathbf{w}) p(\mathbf{w})$$

- One common approach is to define priors on **w**
  - Normal distribution, zero mean, identity covariance
  - "Pushes" parameters towards zero

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

- Often called ***Regularization***
  - Helps avoid very large weights and overfitting
- MAP estimate:

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

M_CAP as Regularization

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ln \left[p(\mathbf{w}) \prod_{j=1}^N P(y^j \mid \mathbf{x}^j, \mathbf{w}) \right] \qquad p(\mathbf{w}) = \prod_i \frac{1}{\kappa\sqrt{2\pi}} \; e^{\frac{-w_i^2}{2\kappa^2}}$$

- Add log p(w) to objective:

$$\ln p(w) \propto -\frac{\lambda}{2} \sum_i w_i^2 \qquad \frac{\partial \ln p(w)}{\partial w_i} = -\lambda w_i$$

  - Quadratic penalty: drives weights towards zero
  - Adds a negative linear term to the gradients

Penalizes high weights, like we did in linear regression

MLE vs. MAP

- Maximum conditional likelihood estimate

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

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

- Maximum conditional a posteriori estimate

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

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