CSE 446
Logistic Regression
Perceptron Learning
Winter 2012
Dan Weld

Some slides from Carlos Guestrin, Luke Zettlemoyer

Machine Learning

Supervised Learning
Labeled Examples → Function
Parametric

Non-parametric

Reinforcement Learning
Experience + Rewards → Policy

Unsupervised Learning
Unlabeled Examples → Clusters

Machine Learning

Supervised Learning
Labeled Examples → Function
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
Parametric

Y Continuous

Y Discrete

Gaussians
Learned in closed form

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

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?

| Coin | P(H|C) | P(C) |
|------|-------|------|
| $C_1$ | 0.1   | 0.05 |
| $C_2$ | 0.5   | 0.25 |
| $C_3$ | 0.9   | 0.70 |

Prior Probabilities

| Coin | P(H|C) | P(C) |
|------|-------|------|
| $C_1$ | 0.1   | 0.05 |
| $C_2$ | 0.5   | 0.25 |
| $C_3$ | 0.9   | 0.70 |

Forms of Inference

| Coin | P(H|C) | Evidence | Predict Next Toss? |
|------|-------|----------|-------------------|
| $C_1$ | 0.1   | Heads; Tails | 0.035 |
| $C_2$ | 0.5   | Heads; Tails | 0.481 |
| $C_3$ | 0.9   | Heads; Tails | 0.485 |

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

<table>
<thead>
<tr>
<th>Prior Hypothesis</th>
<th>Maximum Likelihood Estimate</th>
<th>Maximum A Posteriori Estimate</th>
<th>Bayesian Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>The most likely</td>
<td>Any</td>
<td>Weighted combination</td>
</tr>
<tr>
<td>Any</td>
<td>The most likely</td>
<td>Any</td>
<td></td>
</tr>
</tbody>
</table>

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 $\rightarrow$ Function $\langle X_1, \ldots, X_n, Y \rangle$
- **Parametric**
  - **Y Discrete**
    - Decision Trees
      - Greedy search; pruning
    - Probability of class | features
      1. Learn $P(Y)$, $P(X|Y)$ in closed form
      2. Learn $P(Y|X)$ w/ gradient descent
     - Gaussians
       - Learned in closed form
     - Linear Functions
       - 1. Learned in closed form
       - 2. Using gradient descent
  - **Y Continuous**
    - Gaussians
      - Learned in closed form
    - Linear Functions
      - 1. Learned in closed form
      - 2. Using gradient descent
Generative vs. Discriminative Classifiers

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

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!

Loss Functions (“Error” Term):

Likelihood vs. Conditional Likelihood

Generative (Naïve Bayes) Loss function: Data likelihood

\[
\ln P(D \mid w) = \sum_{j=1}^N \ln P(y_j \mid x_j, w) = \sum_{j=1}^N \ln P(y_j \mid x_j, w) + \sum_{j=1}^N \ln P(x_j \mid w)
\]

Discriminative (Logistic Regr.) Loss function: Conditional Data Likelihood

\[
\ln P(D_{Y=1} \mid P_{X, w}) = \sum_{j=1}^N \ln P(y_j \mid x_j, w)
\]

Discriminative models can’t compute \( P(X \mid w) \)!

Or, ... "They don’t waste effort learning \( P(X) \)"

Focus only on \( P(Y \mid X) \) - all that matters for classification

Maximizing Conditional Log Likelihood

\[
l(w) \equiv \ln \prod_j P(y_j \mid x_j, w) \\
= \sum_j y_j(w_0 + \sum_i w_i x_i) - \ln(1 + \exp(w_0 + \sum_i w_i x_i))
\]

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

Are decision trees generative or discriminative?

Very convenient!

\[
P(Y = 1 \mid X = \langle X_1, \ldots, X_n \rangle) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}
\]

implies

\[
P(Y = 0 \mid X = \langle X_1, \ldots, X_n \rangle) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)}
\]

implies

\[
\frac{P(Y = 0 \mid X)}{P(Y = 1 \mid X)} = \exp(w_0 + \sum_i w_i X_i)
\]

implies

\[
\ln \frac{P(Y = 0 \mid X)}{P(Y = 1 \mid X)} = w_0 + \sum_i w_i X_i
\]
Optimizing concave function — Gradient ascent

• Conditional likelihood for Logistic Regression is concave!

Gradient:
\[ \nabla_w l(w) = \left[ \frac{\partial l(w)}{\partial w_0}, \ldots, \frac{\partial l(w)}{\partial w_m} \right] \]

Update rule:
\[ \Delta w = \eta \nabla_l(w) \]
\[ w_t^{(i+1)} = w_t^{(i)} + \eta \frac{\partial l(w)}{\partial w_t} \]

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

Maximize Conditional Log Likelihood: Gradient ascent

\[ l(w) = \sum_i y^i(w_0 + \sum_j w_j x_j^i) - \ln(1 + \exp(w_0 + \sum_j w_j x_j^i)) \]
\[ \nabla_l(w) = \sum_i \left[ y^i - \frac{x_i^j \exp(w_0 + \sum_j w_j x_j^i)}{1 + \exp(w_0 + \sum_j w_j x_j^i)} \right] \]
\[ \frac{\partial l(w)}{\partial w_t} = \sum_i x_t \left( y_t - P(Y_t = 1|x_t, w) \right) \]

Gradient Descent for LR

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

\[ w_t^{(i+1)} = w_t^{(i)} + \eta \sum_j [y_j - P(Y = 1|X, w)] \]

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

\[ w_t^{(i+1)} = w_t^{(i)} + \eta \sum_j x_t \left[ y_j - P(Y = 1|X, w) \right] \]

until “change” < \( \epsilon \)

That’s all \( M_{cLE} \). How about \( M_{2AP} \)?

\[ p(w | Y, X) \propto P(Y | X, w) p(w) \]

• One common approach is to define priors on \( w \)
  — Normal distribution, zero mean, identity covariance
  — “Pacifies” parameters towards zero
  \[ p(w) = \prod_i \frac{1}{\sqrt{2\pi}} e^{-w_i^2} \]

• Often called Regularization
  — Helps avoid very large weights and overfitting

\[ w^* = \arg \max_w \left[ \sum_i \ln \left( \prod_j P(y_j | x_i, w) \right) \right] \]

MAP estimate:

\[ w^* = \arg \max_w \left[ \sum_i \ln \left( \prod_j P(y_j | x_i, w) \right) \right] \]

M_{2AP} as Regularization

\[ w^* = \arg \max_w \left[ \sum_i \ln \left( \prod_j P(y_j | x_i, w) \right) \right] \]

• Add log \( p(w) \) to objective:
  \[ \ln p(w) - \frac{\lambda}{2} \sum_i w_i^2 \]

• 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

\[ w^* = \arg \max_w \prod_{i=1}^{N} \mathcal{L}(y^i \mid x^i, w) \]

\[ w_j^{(t+1)} = w_j^{(t)} + \eta \sum_{j} x_j^i (y_j^i - P(y_j = 1 \mid x^i, w)) \]

• Maximum conditional a posteriori estimate

\[ w^* = \arg \max_w \prod_{i=1}^{N} \mathcal{P}(y^i \mid x^i, w) \]

\[ w_j^{(t+1)} = w_j^{(t)} + \eta \left( -\lambda w_j^{(t)} + \sum_{j} x_j^i (y_j^i - P(y_j = 1 \mid x^i, w)) \right) \]