10-601 Recitation #4
Gaussian Naive Bayes
and Logistic Regression

October 5th, 2011
Shing-hon Lau
Office hours: Friday 3-4 PM
Agenda

- HW #2 due tomorrow 5 PM
  - Submit written copy and post code to Blackboard
- Gaussian Naive Bayes
- Logistic Regression
- Gradient Descent
- Discriminative vs. Generative classifiers
- Bias/Variance Tradeoff
Naive Bayes

- Features are conditionally independent given class

\[
P(X_1, \ldots, X_n | Y) = \prod_{i=1}^{n} P(X_i | Y)
\]

- Assumed that all variables were binary (or discrete)

- What if we have continuous features?
Gaussian Naive Bayes

- Still assume features are conditionally independent given class

\[
\mathbb{P}(X_1, \ldots, X_n | Y) = \prod_{i=1}^{n} \mathbb{P}(X_i | Y)
\]

\[
\mathbb{P}(X_i | Y) \sim N(\mu, \sigma^2)
\]

- Generally assume Gaussian features (why?)

- Can use other distributions as well
Gaussian Distribution

- Also called normal distribution

\[
f(x) = \frac{1}{\sqrt{2\pi \sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
\]

\[
E(X) = \mu
\]

\[
V(X) = \sigma^2
\]

You will compute the MLE and MAP estimates in HW2
Why Gaussian?

- Many natural phenomenon are normally distributed
  - Biological functions (height, weight, etc.)
- Central Limit Theorem implies that sample means tend to a normal distribution
- Mathematically easy to work with
Working with Continuous Variables

**Discrete variables:**

\[
E(X) = \sum_{x \in Val(X)} x \cdot P(x)
\]

\[
V(X) = E(X - E(X))^2 = \sum_{x \in Val(X)} (x - E(X))^2 \cdot P(x)
\]

\[
P(a \leq X \leq b) = \sum_{x \in [a,b]} P(X = x)
\]

**Continuous variables:**

\[
E(X) = \int_{-\infty}^{\infty} x \cdot f(x)dx
\]

\[
V(X) = E(X - E(X))^2 = \int_{-\infty}^{\infty} (x - E(X))^2 \cdot f(x)dx
\]

\[
P(a \leq X \leq b) = \int_{a}^{b} f(x)dx
\]
Generative vs. Discriminative Classifiers

- Generative classifiers learn $P(X|Y)$
- Use Bayes rule to calculate $P(Y|X)$
- Discriminative classifiers learn $P(Y|X)$
- Which type is Naive Bayes?
Generative vs. Discriminative Classifiers

- Discriminative classifiers are only good for classification
- Generative classifiers enable other tasks (e.g., data generation)
- Generally speaking, generative is more accurate with less data, discriminative with more data
Logistic Regression

- Example of a discriminative classifier

\[
P(Y = 1 | X = < X_1, ... X_n >) = \frac{1}{1 + e^{w_0 + \sum_i w_i X_i}}
\]

\[
P(Y = 0 | X = < X_1, ... X_n >) = \frac{e^{w_0 + \sum_i w_i X_i}}{1 + e^{w_0 + \sum_i w_i X_i}}
\]
Logistic Regression

- Can handle arbitrarily many classes

Now \( y \in \{y_1 \ldots y_R\} \): learn \( R-1 \) sets of weights

For \( k < R \)

\[
P(Y = y_k | X) = \frac{\exp(w_{k0} + \sum_{i=1}^{n} w_{ki}X_i)}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji}X_i)}
\]

For \( k = R \)

\[
P(Y = y_R | X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji}X_i)}
\]
Logistic Regression

Consider learning $f: X \rightarrow Y$, where

- $X$ is a vector of real-valued features, $< X_1 \ldots X_n >$
- $Y$ is boolean
- assume all $X_i$ are conditionally independent given $Y$
- model $P(X_i \mid Y = y_k)$ as Gaussian $N(\mu_{ik}, \sigma_i)$
- model $P(Y)$ as Bernoulli ($\pi$)

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

\[
P(Y = 1 \mid X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}
\]
Finding the weights

- We were able to derive an analytic expression for the weights for the special Gaussian case.

- How can we find the weights in the general case?
Good weights

\[ a = \frac{1}{1 + \exp(-b)} \]

\[ P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)} \]
Maximum Conditional Likelihood Estimate

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

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

\[ D = \{ \langle X^1, Y^1 \rangle, \ldots \langle X^L, Y^L \rangle \} \]

Data conditional likelihood = \[ \prod_l P(Y^l|X^l, W) \]

\[ W_{MCLE} = \arg \max_W \prod_l P(Y^l|W, X^l) \]
Maximum Conditional Likelihood Estimate

\[ l(W) \equiv \ln \prod_l P(Y^l | X^l, W) = \sum_l \ln P(Y^l | X^l, W) \]

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

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

\[ l(W) = \sum_l Y^l \ln P(Y^l = 1 | X^l, W) + (1 - Y^l) \ln P(Y^l = 0 | X^l, W) \]

\[ = \sum_l Y^l \ln \frac{P(Y^l = 1 | X^l, W)}{P(Y^l = 0 | X^l, W)} + \ln P(Y^l = 0 | X^l, W) \]

\[ = \sum_l Y^l (w_0 + \sum_i w_i X^l_i) - \ln(1 + \exp(w_0 + \sum_i w_i X^l_i)) \]
Maximizing $l(w)$

\[
l(W) \equiv \ln \prod_l P(Y^l | X^l, W) = \sum_l Y^l (w_0 + \sum_i^n w_i X^l_i) - \ln(1 + \exp(w_0 + \sum_i^n w_i X^l_i))
\]

- No closed form for the maximum
- So how do we find the MCLE?
Gradient Descent

- Iterative optimization algorithm

- Basic idea:
  - Find the direction of greatest decrease
  - Take a small step in that direction
  - Repeat these two steps until we are satisfied
  - Usually stop when change is very small

- Guaranteed to find optimal point in some cases
1-D Gradient Descent

\[ y = f(x) = x^2 \]

\[ \frac{dy}{dx} = 2x \]

\[ \eta = 0.1 \]

- 1-D gradient is the derivative
1-D Gradient Descent

\[ y = f(x) = x^2 \]
\[ \frac{dy}{dx} = 2x \]
\[ \eta = 0.1 \]

- Start at a random point \((1, 1)\)
1-D Gradient Descent

\[ y = f(x) = x^2 \]
\[ \frac{dy}{dx} = 2x \]
\[ \eta = 0.1 \]

- Take a step in the negative gradient direction
1-D Gradient Descent

\[ y = f(x) = x^2 \]
\[ \frac{dy}{dx} = 2x \]
\[ \eta = 0.1 \]

- Repeat the process
1-D Gradient Descent

\[ y = f(x) = x^2 \]
\[ \frac{dy}{dx} = 2x \]
\[ \eta = 0.1 \]

- Eventually converge to the optimum point
Problems with Gradient Descent

- If step-size is too big, can end up going back and forth between two values
- If the function is not convex/concave, we may end up in a local optima
Bad Step-Size

\[ y = f(x) = x^2 \]
\[ \frac{dy}{dx} = 2x \]
\[ \eta = 1.0 \]

- Start at \((1, 1)\)
Bad Step-Size

\[ y = f(x) = x^2 \]
\[ \frac{dy}{dx} = 2x \]
\[ \eta = 1.0 \]

- Step to \((-1, 1)\)
Bad Step-Size

\[ y = f(x) = x^2 \]
\[ \frac{dy}{dx} = 2x \]
\[ \eta = 1.0 \]

- Step back to 
  \((1, 1)\)
Non-convexity
Non-convexity
Non-convexity
Gradient Descent

- Generally have to shrink your step size as you continue to iterate
- Try to stick to convex/concave functions
- Do random restarts if you must use a non-convex/non-concave objective
n-dimensional Gradient Descent

- Use partial derivatives to compute the gradient
- Partial derivatives:

\[
\begin{align*}
f(x, y, z) &= xyz - 3xln(z) \\
\frac{\partial f}{\partial x} &= yz - 3ln(z) \\
\frac{\partial f}{\partial z} &= xy - \frac{3x}{z}
\end{align*}
\]
n-dimensional Gradient Descent

- Gradient is n-dimensional vector
- Gradient direction is the direction of greatest increase
- First-order (i.e., linear) approximation
- Second-order Newton methods
Logistic Regression and Gradient Descent

\[ l(W) \equiv \ln \prod_l P(Y^l | X^l, W) \]

\[ = \sum_l Y^l (w_0 + \sum_i^n w_i X^l_i) - \ln(1 + \exp(w_0 + \sum_i^n w_i X^l_i)) \]

\[ \frac{\partial l(W)}{\partial w_i} = \sum_l X^l_i (Y^l - \hat{P}(Y^l = 1 | X^l, W)) \]

- Objective function is concave!
MAP and Logistic Regression

Maximum a posteriori estimate with prior $W \sim N(0, \sigma I)$

$$W \leftarrow \arg \max_W \ln[P(W) \prod_l P(Y^l|X^l, W)]$$

$$w_i \leftarrow w_i - \eta \lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W))$$
MAP and Logistic Regression

- Use a MAP estimate to avoid overfitting (just like Naive Bayes)
- What can happen to weights without regularization term?
- What does the regularization term help do?
Other Forms of Regularization

- Can apply a Lasso or Ridge penalty to weights
- Lasso makes many weights zero
- Ridge shrinks all of the weights

\[
\ell(W) = \sum_l Y^l (w_0 + \sum_i^n w_i X^l_i) - \ln(1 + \exp(w_0 + \sum_i^n w_i X^l_i)) - \lambda ||w||_1
\]

\[
\ell(W) = \sum_l Y^l (w_0 + \sum_i^n w_i X^l_i) - \ln(1 + \exp(w_0 + \sum_i^n w_i X^l_i)) - \lambda ||w||_2^2
\]
Bias/Variance Tradeoff

- Simpler models are more biased because they make more assumptions.
- More complex models are more variable, since they depend on the particulars of the data provided.
- Have to trade the two off to get the best classifier possible.
Extra Slides
Correlated Features

- Worst case scenario: duplicated features
- What will Naive Bayes do with duplicated features?
- What will Logistic Regression do with duplicated features?
- What if there is just correlation?
Estimating Some Features Jointly

- What if we are not willing to assume that all features are conditionally independent?
- How can we do Naive Bayes?
- What is the price we pay for not assuming conditional independence?
Estimating Some Features Jointly

- Graphical models are a formalization of this idea
- Can do things like Tree-Augmented Naive Bayes (TAN)
- More general framework for an arbitrary set of conditional independence assumptions
Neural Network Preview

- One sigmoid function is good (Logistic Regression), so more must be better
- Can chain them so that the output of one are the inputs to the next
- “Mimics” the brain (kind of), so such systems are termed Neural Networks
Numerical Gradient Descent

• What if we can only evaluate the function but cannot evaluate its derivative?
• Can take tiny steps in each direction to determine gradient
• Generally a lot more expensive because of all the function evaluations