Case Study 1: Estimating Click Probabilities

Intro
Logistic Regression
Gradient Descent + SGD

Machine Learning for Big Data
CSE547/STAT548, University of Washington
Sham Kakade
March 28, 2017

Announcements:

• Lecture 2 cancelled
  — TAs will hold a python recitation
• HW1 posted today.
• Readings: please do them.
• Project Proposals: please start thinking about it!

• Today:
  — Review: click prediction and logistic regression
Ad Placement Strategies

- Companies bid on ad prices

- Which ad wins? (many simplifications here)
  - Naively:
  - But:
  - Instead:

Key Task: Estimating Click Probabilities

- What is the probability that user $i$ will click on ad $j$?

- Not important just for ads:
  - Optimize search results
  - Suggest news articles
  - Recommend products

- Methods much more general, useful for:
  - Classification
  - Regression
  - Density estimation
Learning Problem for Click Prediction

- Prediction task:
- Features:
- Data:
  - Batch:
  - Online:

- Many approaches (e.g., logistic regression, SVMs, naïve Bayes, decision trees, boosting,...)
  - Focus on logistic regression; captures main concepts, ideas generalize to other approaches

Logistic Regression

Learn \( P(Y|X) \) directly
- Assume a particular functional form
- Sigmoid applied to a linear function of the data:

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

Features can be discrete or continuous!
Very convenient!

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

implies

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

Digression: Logistic regression more generally

• Logistic regression in more general case, where \( Y \) in \( \{y_1, \ldots, y_R\} \)

  for \( k < R \)

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

  for \( k = R \) (normalization, so no weights for this class)

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

Features can be discrete or continuous!

©Kakade 2017
Loss function: Conditional Likelihood

- Have a bunch of iid data of the form:

- Discriminative (logistic regression) loss function:

  Conditional Data Likelihood

\[
\ln P(D_Y \mid D_X, w) = \sum_{j=1}^{N} \ln P(y^j \mid x^j, w)
\]

Expressing Conditional Log Likelihood

\[
l(w) \equiv \sum_{j} \ln P(y^j \mid x^j, w)
\]

\[
\ell(w) = \sum_{j} y^j \ln P(Y = 1 \mid x^j, w) + (1 - y^j) \ln P(Y = 0 \mid x^j, w)
\]

\[
= \sum_{j} y^j (w_0 + \sum_{i=1}^{d} w_i x_i^j) - \ln \left( 1 + \exp \left( w_0 + \sum_{i=1}^{d} w_i x_i^j \right) \right)
\]
Maximizing Conditional Log Likelihood

\[ l(w) \equiv \ln \prod_j P(y^j|x^j, w) \]

\[ = \sum_j y^j(w_0 + \sum_{i=1}^d w_i x_i^j) - \ln \left( 1 + \exp(w_0 + \sum_{i=1}^d w_i x_i^j) \right) \]

**Good news:** \( l(w) \) is concave function of \( w \), no local optima problems

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

**Good news:** concave functions easy to optimize

Optimizing concave function – Gradient ascent

- Conditional likelihood for logistic regression is **concave**
- Find optimum with **gradient ascent**

**Gradient:**

\[ \nabla_w l(w) = \left[ \frac{\partial l(w)}{\partial w_0}, \ldots, \frac{\partial l(w)}{\partial w_n} \right]' \]

**Update rule:**

\[ \Delta w = \eta \nabla_w l(w) \]

\[ w_i(t+1) \leftarrow w_i(t) + \eta \frac{\partial l(w)}{\partial w_i} \]

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

Gradient ascent algorithm: iterate until change $< \varepsilon$

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

For $i = 1, \ldots, d$,

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

repeat

Regularized Conditional Log Likelihood

- If data are linearly separable, weights go to infinity
- Leads to overfitting $\rightarrow$ Penalize large weights
- Add regularization penalty, e.g., $L_2$:

$$\ell(w) = \ln \prod_j P(y^j | x^j, w)) - \frac{\lambda \| w \|^2}{2}$$

- Practical note about $w_0$: 
Standard v. Regularized Updates

• Maximum conditional likelihood estimate

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

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

• Regularized maximum conditional likelihood estimate

\[
\mathbf{w}^* = \arg \max_{\mathbf{w}} \ln \left( \prod_{j} P(y^j | x^j, \mathbf{w}) \right) - \frac{\lambda}{2} \sum_{i>0} w_i^2
\]

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

Stopping criterion

\[
\ell(\mathbf{w}) = \ln \prod_j P(y^j | x^j, \mathbf{w})) - \frac{\lambda}{2} ||\mathbf{w}||^2
\]

• Regularized logistic regression is strongly concave
  – Negative second derivative bounded away from zero:

• Strong concavity (convexity) is super helpful!!
• For example, for strongly concave \( \ell(\mathbf{w}) \):

\[
\ell(\mathbf{w}^*) - \ell(\mathbf{w}) \leq \frac{1}{2\lambda} ||\nabla\ell(\mathbf{w})||^2
\]
Convergence rates for gradient descent/ascent

- Number of iterations to get to accuracy
  \[ \ell(w^*) - \ell(w) \leq \epsilon \]

- If func l(w) Lipschitz: \( O(1/\epsilon^2) \)

- If gradient of func Lipschitz: \( O(1/\epsilon) \)

- If func is strongly convex: \( O(\ln(1/\epsilon)) \)

Challenge 1: Complexity of computing gradients

- What’s the cost of a gradient update step for LR???

\[
 w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \sum_j x_i^j [y_i^j - \hat{P}(Y_i^j = 1 | x^j, w)] \right\}
\]
Challenge 2: Data is streaming

- Assumption thus far: **Batch data**

- But, click prediction is a streaming data task:
  - User enters query, and ad must be selected:
    - Observe $x^i$, and must predict $y^i$

  - User either clicks or doesn’t click on ad:
    - Label $y^i$ is revealed afterwards
      - Google gets a reward if user clicks on ad

  - Weights must be updated for next time:

---

Learning Problems as Expectations

- Minimizing loss in training data:
  - Given dataset:
    - Sampled iid from some distribution $p(x)$ on features:
    - Loss function, e.g., hinge loss, logistic loss,
    - We often minimize loss in training data:
      \[
      \ell_D(w) = \frac{1}{N} \sum_{j=1}^{N} \ell(w, x^j)
      \]

- However, we should really minimize expected loss on all data:
  \[
  \ell(w) = E_x [\ell(w, x)] = \int p(x) \ell(w, x) dx
  \]

- So, we are approximating the integral by the average on the training data
Gradient Ascent in Terms of Expectations

• “True” objective function:
\[ \ell(w) = E_x [\ell(w, x)] = \int p(x) \ell(w, x) dx \]
• Taking the gradient:
• “True” gradient ascent rule:
• How do we estimate expected gradient?

SGD: Stochastic Gradient Ascent (or Descent)

• “True” gradient: \[ \nabla \ell(w) = E_x [\nabla \ell(w, x)] \]
• Sample based approximation:

• What if we estimate gradient with just one sample???
  – Unbiased estimate of gradient
  – Very noisy!
  – Called stochastic gradient ascent (or descent)
    • Among many other names
  – VERY useful in practice!!!
Stochastic Gradient Ascent: General Case

- Given a stochastic function of parameters:
  - Want to find maximum

- Start from $w^{(0)}$
- Repeat until convergence:
  - Get a sample data point $x^t$
    - Update parameters:

- Works in the online learning setting!
- Complexity of each gradient step is constant in number of examples!
- In general, step size changes with iterations

Stochastic Gradient Ascent for Logistic Regression

- Logistic loss as a stochastic function:
  \[ E_x [\ell(w, x)] = E_x \left[ \ln P(y|x, w) - \frac{\lambda \|w\|_2^2}{2} \right] \]

- Batch gradient ascent updates:
  \[ w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \frac{1}{N} \sum_{j=1}^{N} x_i^{(j)} [y^{(j)} - P(Y = 1|x^{(j)}, w^{(t)})] \right\} \]

- Stochastic gradient ascent updates:
  - Online setting:
    \[ w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t \left\{ -\lambda w_i^{(t)} + x_i^{(t)} [y^{(t)} - P(Y = 1|x^{(t)}, w^{(t)})] \right\} \]
Convergence Rate of SGD

- **Theorem:**
  - (see Nemirovski et al ’09 from readings)
  - Let $f$ be a strongly convex stochastic function
  - Assume gradient of $f$ is Lipschitz continuous and bounded

- Then, for step sizes:

- The expected loss decreases as $O(1/t)$:

Convergence Rates for Gradient Descent/Ascent vs. SGD

- Number of iterations to get to accuracy
  \[
  \ell(w^*) - \ell(w) \leq \varepsilon
  \]

- Gradient descent:
  - If func is strongly convex: $O(\ln(1/\varepsilon))$ iterations

- Stochastic gradient descent:
  - If func is strongly convex: $O(1/\varepsilon)$ iterations

- Seems exponentially worse, but much more subtle:
  - Total running time, e.g., for logistic regression:
    - Gradient descent:
    - SGD:
    - SGD can win when we have a lot of data
  - See readings for more details
What you should know about Logistic Regression (LR) and Click Prediction

- Click prediction problem:
  - Estimate probability of clicking
  - Can be modeled as logistic regression
- Logistic regression model: Linear model
- Gradient ascent to optimize conditional likelihood
- Overfitting + regularization
- Regularized optimization
  - Convergence rates and stopping criterion
- Stochastic gradient ascent for large/streaming data
  - Convergence rates of SGD

©Kakade 2017