10-601 Introduction to Machine Learning

Logistic Regression

Readings:
Murphy Ch. 8.1-3, 8.6
Elken (2014) Notes

Slides:
Courtesy William Cohen

Matt Gormley
Lecture 4
September 12, 2016
Reminders

• Homework 2:
  – Extension: due Friday (9/16) at 5:30pm
• Recitation schedule posted on course website
Outline

• **Background: Hyperplanes**
• **Learning as Optimization**
  – MLE Example
  – Gradient descent (in pictures)
• **Gradient descent for Linear Classifiers**
  – Logistic Regression
  – Stochastic Gradient Descent (SGD)
  – Computing the gradient
  – Details (learning rate, finite differences)
• **Logistic Regression and Overfitting**
  – (non-stochastic) Gradient Descent
  – Difference of expectations
• **Regularization**
  – L2 Regularization
  – Regularization as MAP estimation
• **Discriminative vs. Generative Classifiers**
Why don’t we drop the generative model and try to learn this hyperplane directly?
Hyperplanes

Hyperplane (Definition 1):
\[ \mathcal{H} = \{ \mathbf{x} : \mathbf{w}^T \mathbf{x} = b \} \]

Hyperplane (Definition 2):
\[ \mathcal{H} = \{ \mathbf{x} : \mathbf{w}^T \mathbf{x} = 0 \text{ and } x_1 = 1 \} \]

Half-spaces:
\[ \mathcal{H}^+ = \{ \mathbf{x} : \mathbf{w}^T \mathbf{x} > 0 \text{ and } x_1 = 1 \} \]
\[ \mathcal{H}^- = \{ \mathbf{x} : \mathbf{w}^T \mathbf{x} < 0 \text{ and } x_1 = 1 \} \]
LEARNING AS OPTIMIZATION
Learning as optimization

• The main idea today
  – Make two assumptions:
    • the classifier is linear, like naïve Bayes
      – ie roughly of the form $f_w(x) = \text{sign}(x \cdot w)$
    • we want to find the classifier $w$ that “fits the data best”
  – Formalize as an optimization problem
    • Pick a loss function $J(w, D)$, and find
      $\arg\min_w J(w)$
    • OR: Pick a “goodness” function, often $\Pr(D | w)$, and find the $\arg\max_w f_D(w)$
Learning as optimization: warmup

Find: optimal (MLE) parameter $\theta$ of a binomial

Dataset: $D=\{x_1,\ldots,x_n\}$, $x_i$ is 0 or 1, $k$ of them are 1

$$P(D \mid \theta) = const \prod_i \theta^{x_i} (1 - \theta)^{1-x_i} = \theta^k (1 - \theta)^{n-k}$$

$$\frac{d}{d\theta} P(D \mid \theta) = \frac{d}{d\theta} \left( \theta^k (1 - \theta)^{n-k} \right)$$

$$= \left( \frac{d}{d\theta} \theta^k \right) (1 - \theta)^{n-k} + \theta^k \frac{d}{d\theta} (1 - \theta)^{n-k}$$

$$= k\theta^{k-1} (1 - \theta)^{n-k} + \theta^k (n-k)(1 - \theta)^{n-k-1} (-1)$$

$$= \theta^{k-1} (1 - \theta)^{n-k-1} \left( k(1 - \theta) - \theta(n - k) \right)$$
Learning as optimization: warmup

Goal: Find the best parameter $\theta$ of a binomial
Dataset: $D=\{x_1, \ldots, x_n\}$, $x_i$ is 0 or 1, $k$ of them are 1

$$\theta = 0$$

$$\theta = 1$$

$$k - k\theta - n\theta + k\theta = 0$$

$$\Rightarrow n\theta = k$$

$$\Rightarrow \theta = k/n$$
Learning as optimization with gradient ascent

• Goal: Learn the parameter $w$ of ...
• Dataset: $D=\{(x_1,y_1),\ldots,(x_n,y_n)\}$
• Use your model to define
  – $\Pr(D \mid w) = \ldots$
• Set $w$ to maximize Likelihood
  – Usually we use numeric methods to find the optimum
  – i.e., gradient ascent: repeatedly take a small step in the direction of the gradient

Difference between ascent and descent is only the sign: I’m going to be sloppy here about that.
Gradient ascent

To find $\text{argmin}_x f(x)$:

- Start with $x_0$
- For $t=1$...
  - $x_{t+1} = x_t + \lambda f'(x_t)$
    where $\lambda$ is small
Gradient descent

Likelihood: ascent

Loss: descent
Pros and cons of gradient descent

• Simple and often quite effective on ML tasks
• Often very scalable
• Only applies to smooth functions (differentiable)
• Might find a local minimum, rather than a global one
Pros and cons of gradient descent

There is only one local optimum if the function is convex
Outline

• **Background: Hyperplanes**
• **Learning as Optimization**
  – MLE Example
  – Gradient descent (in pictures)
• **Gradient descent for Linear Classifiers**
  – Logistic Regression
  – Stochastic Gradient Descent (SGD)
  – Computing the gradient
  – Details (learning rate, finite differences)
• **Logistic Regression and Overfitting**
  – (non-stochastic) Gradient Descent
  – Difference of expectations
• **Regularization**
  – L2 Regularization
  – Regularization as MAP estimation
• **Discriminative vs. Generative Classifiers**
LEARNING LINEAR CLASSIFIERS WITH GRADIENT ASCENT

PART 1: THE IDEA
Using gradient ascent for linear classifiers

1. Replace \( \text{sign}(\mathbf{x} \cdot \mathbf{w}) \) with something differentiable: e.g. the logistic(\( \mathbf{x} \cdot \mathbf{w} \))

\[
P(Y = \text{pos} \mid X = \mathbf{x}) \equiv \frac{1}{1 + e^{-x \cdot \mathbf{w}}}
\]

logistic(\( u \)) \equiv \frac{1}{1 + e^{-u}}
Using gradient ascent for linear classifiers

1. Replace sign($x \cdot w$) with something differentiable: e.g., the logistic($x \cdot w$)

2. Define a function on the data that formalizes “how well $w$ predicts the data” (loss function)

- Assume $y=0$ or $y=1$
- Our optimization target is log of conditional likelihood

$$P(Y = 1 \mid X = x) \equiv \frac{1}{1 + e^{-x \cdot w}}$$

$$P(y_i \mid x_i, w) \equiv \begin{cases} 
\frac{1}{1 + \exp(-x_i \cdot w)} & \text{if } y_i = 1 \\
\left(1 - \frac{1}{1 + \exp(-x_i \cdot w)}\right) & \text{if } y_i = 0 
\end{cases}$$

$$LCL_D(w) \equiv \sum_i \log P(y_i \mid x_i, w)$$
Using gradient ascent for linear classifiers

1. Replace \( \text{sign}(x \cdot w) \) with something differentiable: e.g. the logistic\((x \cdot w)\)

2. Define a function on the data that formalizes “how well \( w \) predicts the data” (log likelihood)

3. Differentiate the likelihood function and use gradient ascent
   - Start with \( w_0 \)
   - For \( t=1 \ldots \)
     - \( w_{t+1} = w_t + \lambda \text{Loss'}_D(w_t) \)
     - where \( \lambda \) is small

\[
\text{LCL}_D(w) \equiv \sum_i \log P(y_i | x_i, w)
\]
Stochastic Gradient Descent (SGD)

• Goal: Learn the parameter $\theta$ of …
• Dataset: $D=\{x_1, \ldots, x_n\}$
  – or maybe $D=\{(x_1, y_1), \ldots, (x_n, y_n)\}$
• Write down $Pr(D | \theta)$ as a function of $\theta$
• For large datasets this is expensive: we don’t want to load all the data $D$ into memory, and the gradient depends on all the data
• An alternative:
  – pick a small subset of examples $B<<D$
  – approximate the gradient using them
    • “on average” this is the right direction
  – take a step in that direction
  – repeat….

B = one example is a very popular choice

Difference between ascent and descent is only the sign: we’ll be sloppy and use the more common “SGD”
Using SGD for logistic regression

1. \( P(y|x) = \text{logistic}(x \cdot w) \)

2. Define the likelihood: \( \text{LCL}_D(w) \equiv \sum_i \log P(y_i | x_i, w) \)

3. Differentiate the LCL function and use gradient descent to minimize
   - Start with \( w_0 \)
   - For \( t=1,\ldots,T \) - until convergence
     - For each example \( x,y \) in \( D \):
       - \( w_{t+1} = w_t + \lambda \ L_{x,y}(w_t) \)
       - where \( \lambda \) is small

More steps, noisier path toward the minimum, but each step is cheaper
Outline

• **Background: Hyperplanes**
• **Learning as Optimization**
  – MLE Example
  – Gradient descent (in pictures)
• **Gradient descent for Linear Classifiers**
  – Logistic Regression
  – Stochastic Gradient Descent (SGD)
  – Computing the gradient
  – Details (learning rate, finite differences)
• **Logistic Regression and Overfitting**
  – (non-stochastic) Gradient Descent
  – Difference of expectations
• **Regularization**
  – L2 Regularization
  – Regularization as MAP estimation
• **Discriminative vs. Generative Classifiers**
I will start by deriving the gradient for one example (e.g., SGD) and then move to “batch” gradient descent.
Likelihood on one example is:

$$\log P(Y = y | X = x, w) = \begin{cases} 
\log p & \text{if } y = 1 \\
\log(1 - p) & \text{if } y = 0 
\end{cases}$$

$$p \equiv \frac{1}{1 + e^{-x \cdot w}} = \frac{1}{1 + \exp(-\sum_j x^j w^j)}$$

We’re going to dive into this thing here: $\frac{d}{dw}(p)$

$$\frac{\partial}{\partial w_j} \log P(Y = y | X = x, w) = \begin{cases} 
\frac{1}{p} \frac{\partial}{\partial w_j} p & \text{if } y = 1 \\
\frac{1}{1 - p} \left(-\frac{\partial}{\partial w_j} p\right) & \text{if } y = 0 
\end{cases}$$

$$(\log f)' = \frac{1}{f} f'.$$
\[ p \equiv \frac{1}{1 + e^{-x \cdot w}} = \frac{1}{1 + \exp(-\sum_j x^j w^j)} \]

\[
1 - p = \frac{1 + \exp(-\sum_j x^j w^j)}{1 + \exp(-\sum_j x^j w^j)} - \frac{1}{1 + \exp(-\sum_j x^j w^j)} = \frac{\exp(-\sum_j x^j w^j)}{1 + \exp(-\sum_j x^j w^j)}
\]

\[
\frac{\partial}{\partial w^j} p = \frac{\partial}{\partial w^j} (1 + \exp(-\sum_j x^j w^j))^{-1}
\]

\[
= (-1)(1 + \exp(-\sum_j x^j w^j))^{-2} \frac{\partial}{\partial w^j} \exp(-\sum_j x^j w^j)
\]

\[
= (-1)(1 + \exp(-\sum_j x^j w^j))^{-2} \exp(-\sum_j x^j w^j)(-x^j)
\]

\[
p = \frac{1}{1 + \exp(-\sum_j x^j w^j)}
\]

\[
\frac{\partial}{\partial w^j} p = p(1 - p)x^j
\]
\[ p \equiv \frac{1}{1 + e^{-x \cdot w}} = \frac{1}{1 + \exp(-\sum_j x^j w^j)} \]

\[ \frac{\partial}{\partial w^j} p = p(1 - p)x^j \]
\[ \log P(Y = y | X = x, w) = \begin{cases} 
\log p & \text{if } y = 1 \\
\log(1 - p) & \text{if } y = 0 
\end{cases} \]

\[ \frac{\partial}{\partial w^j} \log P(Y = y | X = x, w) = \begin{cases} 
\frac{1}{p} \frac{\partial}{\partial w^j} p & \text{if } y = 1 \\
\frac{1}{1-p} (-\frac{\partial}{\partial w^j} p) & \text{if } y = 0 
\end{cases} \]

\[ \frac{\partial}{\partial w^j} p = p(1 - p)x^j \]

\[ \frac{\partial}{\partial w^j} \log P(Y = y | X = x, w) = \begin{cases} 
\frac{1}{p} p(1 - p)x^j = (1 - p)x^j & \text{if } y = 1 \\
\frac{1}{1-p} (-1)p(1 - p)x^j = -px^j & \text{if } y = 0 
\end{cases} \]

\[ \frac{\partial}{\partial w^j} \log P(Y = y | X = x, w) = (y - p)x^j \]

\[ w^{(t+1)} = w^{(t)} + \lambda(y - p)x \]
Breaking it down: SGD for logistic regression

1. \( P(y|x) = \text{logistic}(x \cdot w) \)
2. Define a function

\[
LCL_D(w) = \sum \log P(y_i | x_i, w)
\]

3. Differentiate the function and use gradient descent
   - Start with \( w_0 \)
   - For \( t=1,...,T \) - until convergence
     - For each example \( x,y \) in \( D \):
       - \( p_i = \left(1 + \exp(-x \cdot w)\right)^{-1} \)
       - \( w_{t+1} = w_t + \lambda L_{x,y}(w_t) = w_t + \lambda(y - p_i)x \)
         where \( \lambda \) is small
Details: Picking learning rate

• Use grid-search in log-space over small values on a tuning set:
  – e.g., 0.01, 0.001, …

• Sometimes, decrease after each pass:
  – e.g factor of $1/(1 + dt)$, $t=$epoch
  – sometimes $1/t^2$

• Fancier techniques I won’t talk about:
  – Adaptive gradient: scale gradient differently for each dimension (Adagrad, ADAM, ….)
Details: Debugging

- Check that gradient is indeed a locally good approximation to the likelihood
  - “finite difference test”
Outline

• Background: Hyperplanes
• Learning as Optimization
  – MLE Example
  – Gradient descent (in pictures)
• Gradient descent for Linear Classifiers
  – Logistic Regression
  – Stochastic Gradient Descent (SGD)
  – Computing the gradient
  – Details (learning rate, finite differences)
• Logistic Regression and Overfitting
  – (non-stochastic) Gradient Descent
  – Difference of expectations
• Regularization
  – L2 Regularization
  – Regularization as MAP estimation
• Discriminative vs. Generative Classifiers
LOGISTIC REGRESSION: OBSERVATIONS
Convexity and logistic regression

This LCL function is convex: there is only one local minimum.

So gradient descent will give the global minimum.
Non-stochastic gradient descent

\[ \frac{\partial}{\partial w^j} \log P(Y = y|X = x, w) = (y - p)x^j \]

- In batch gradient descent, average the gradient over all the examples \( D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \)

\[ \frac{\partial}{\partial w^j} \log P(D|w) = \frac{1}{n} \sum_i (y_i - p_i)x_i^j = \]

\[ = \frac{1}{n} \sum_{i: x_i^j = 1} y_i - \frac{1}{n} \sum_{i: x_i^j = 1} p_i \]
Non-stochastic gradient descent

• This can be interpreted as a difference between the expected value of \( y \mid x^j = 1 \text{ in the data} \) and the expected value of \( y \mid x^j = 1 \text{ as predicted by the model} \)

• Gradient ascent tries to make those equal

\[
\frac{\partial}{\partial w^j} \log P(D|w) = \frac{1}{n} \sum_i (y_i - p_i) x_i^j = \\
= \frac{1}{n} \sum_{i: x_i^j = 1} y_i - \frac{1}{n} \sum_{i: x_i^j = 1} p_i
\]
This LCL function “overfits”

- This can be interpreted as a difference between the expected value of $y \vert x^j=1$ in the data and the expected value of $y \vert x^j=1$ as predicted by the model.

- Gradient ascent tries to make those equal.

$$\frac{\partial}{\partial w^j} \log P(D \vert w) = \frac{1}{n} \sum_i (y_i - p_i) x_i^j = \frac{1}{n} \sum_{i: x_i^j=1} y_i - \frac{1}{n} \sum_{i: x_i^j=1} p_i$$

- That’s impossible for some $w^j$!
  - e.g., if $x^j = 1$ only in positive examples, the gradient is always positive.
This LCL function “overfits”

- This can be interpreted as a difference between the expected value of $y | x^j = 1 \text{ in the data}$ and the expected value of $y | x^j = 1 \text{ as predicted by the model}$

- Gradient ascent tries to make those equal

$$\frac{\partial}{\partial w^j} \log P(D|w) = \frac{1}{n} \sum_i (y_i - p_i) x^j_i = \frac{1}{n} \sum_{i:x^j_i = 1} y_i - \frac{1}{n} \sum_{i:x^j_i = 1} p_i$$

- That’s impossible for some $w^j$ e.g., if they appear only in positive examples, gradient is always possible.

- Using this LCL function for text: practically, it’s important to discard rare features to get good results.
This LCL function “overfits”

- Overfitting is often a problem in supervised learning.
  - When you fit the data (minimize LCL) are you fitting “real structure” in the data or “noise” in the data?
  - Will the patterns you see appear in a test set or not?

$\varepsilon$

hi error

Error/LCL on training set $D$

Error/LCL on an unseen test set $D_{test}$

more features

38
Outline

• Background: Hyperplanes
• Learning as Optimization
  – MLE Example
  – Gradient descent (in pictures)
• Gradient descent for Linear Classifiers
  – Logistic Regression
  – Stochastic Gradient Descent (SGD)
  – Computing the gradient
  – Details (learning rate, finite differences)
• Logistic Regression and Overfitting
  – (non-stochastic) Gradient Descent
  – Difference of expectations
• Regularization
  – L2 Regularization
  – Regularization as MAP estimation
• Discriminative vs. Generative Classifiers
REGULARIZED LOGISTIC REGRESSION
Regularized logistic regression as a MAP

• Minimizing our LCL function maximizes log conditional likelihood of the data (LCL):

\[
LCL_D(w) \equiv \sum \log P(y_i \mid x_i, w)
\]

– like MLE: \( \max_w \Pr(D \mid w) \)

– … but focusing on just a part of \( D \)

• Another possibility: introduce a prior over \( w \)

– maximize something like a MAP

\[
\Pr(w \mid D) = 1/Z \times \Pr(D \mid w)\Pr(w)
\]

• If the prior \( w^j \) is zero-mean Gaussian then

\[
\Pr(w^j) = 1/Z \exp(w^j)^{-2}
\]
Regularized logistic regression

- Replace LCL

\[
\log P(Y = y|X = x, \mathbf{w}) = \begin{cases} 
\log p & \text{if } y = 1 \\
\log(1 - p) & \text{if } y = 0
\end{cases}
\]

- with LCL + penalty for large weights, eg

\[
LCL - \mu \sum_{j=1}^{d} (w^j)^2
\]

- So the update for \( w^j \) becomes:

\[
w^j = w^j + \lambda((y - p)x^j - 2\mu w^j)
\]

- Or

\[
w^j = w^j + \lambda(y - p)x^j - \lambda 2\mu w^j
\]

(Log Conditional Likelihood, our LCL function)
Breaking it down: regularized SGD for logistic regression

1. \( P(y|x) = \text{logistic}(x \cdot w) \)

2. Define a function

\[
LCL_D(w) = \sum_i \log P(y_i | x_i, w) + \mu \|w\|^2
\]

3. Differentiate the function and use gradient descent
   - Start with \( w_0 \)
   - For \( t=1,\ldots,T \) - until convergence
     - For each example \( x,y \) in \( D \):
       - \( p_i = \left( 1 + \exp(-x \cdot w) \right)^{-1} \)
       - \( w_{t+1} = w_t + \lambda L_{x,y}(w_t) = w_t + \lambda(y - p_i)x - 2\lambda\mu w_t \)

where \( \lambda \) is small
Outline

• **Background: Hyperplanes**
• **Learning as Optimization**
  – MLE Example
  – Gradient descent (in pictures)
• **Gradient descent for Linear Classifiers**
  – Logistic Regression
  – Stochastic Gradient Descent (SGD)
  – Computing the gradient
  – Details (learning rate, finite differences)
• **Logistic Regression and Overfitting**
  – (non-stochastic) Gradient Descent
  – Difference of expectations
• **Regularization**
  – L2 Regularization
  – Regularization as MAP estimation
• **Discriminative vs. Generative Classifiers**
DISCRIMINATIVE AND GENERATIVE CLASSIFIERS
Generative vs. Discriminative

• **Generative Classifiers:**
  – Example: Naïve Bayes
  – Define a joint model of the observations $x$ and the labels $y$: $p(x, y)$
  – Learning maximizes (joint) likelihood
  – Use Bayes’ Rule to classify based on the posterior:
    $$ p(y|x) = \frac{p(x|y)p(y)}{p(x)} $$

• **Discriminative Classifiers:**
  – Example: Logistic Regression
  – Directly model the conditional: $p(y|x)$
  – Learning maximizes conditional likelihood
Finite Sample Analysis (Ng & Jordan, 2002)

[Assume that we are learning from a finite training dataset]

**If model assumptions are correct:** Naive Bayes is a more efficient learner (requires fewer samples) than Logistic Regression

**If model assumptions are incorrect:** Logistic Regression has lower asymptotic error, and does better than Naïve Bayes
solid: NB  dashed: LR
Naïve Bayes makes stronger assumptions about the data but needs fewer examples to estimate the parameters

“On Discriminative vs Generative Classifiers: ....”
Andrew Ng and Michael Jordan, NIPS 2001.
Generative vs. Discriminative Learning (Parameter Estimation)

**Naïve Bayes:**
Parameters are decoupled $\rightarrow$ Closed form solution for MLE

**Logistic Regression:**
Parameters are coupled $\rightarrow$ No closed form solution – must use iterative optimization techniques instead
Summary

1. Discriminative classifiers directly model the conditional, $p(y|x)$

2. Logistic regression is a simple linear classifier, that retains a probabilistic semantics

3. Parameters in LR are learned by iterative optimization (e.g. SGD)

4. Regularization helps to avoid overfitting