Logistic Regression
Logistic Regression

Probabilistic Learning
Reminders

• Homework 3: KNN, Perceptron, Lin.Reg.
  – Out: Wed, Feb 7
  – Due: Wed, Feb 14 at 11:59pm
STOCHASTIC GRADIENT DESCENT
Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

1: procedure SGD($D, \theta^{(0)}$)
2: \hspace{1em} $\theta \leftarrow \theta^{(0)}$
3: \hspace{1em} while not converged do
4: \hspace{2em} for $i \in \text{shuffle} \{1, 2, \ldots, N\}$ do
5: \hspace{3em} \hspace{1em} $\theta \leftarrow \theta - \lambda \nabla_{\theta} J^{(i)}(\theta)$
6: \hspace{1em} return $\theta$

We need a per-example objective:

Let $J(\theta) = \sum_{i=1}^{N} J^{(i)}(\theta)$
Expectations of Gradients

\[ \frac{dJ(\theta)}{d\theta_j} = \frac{1}{N} \sum_{i=1}^{N} \frac{d}{d\theta_j} (J_i(\theta)) \]

\[ \nabla J(\theta) = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix} = \frac{1}{N} \sum_{i=1}^{N} \nabla J_i(\theta) \]

Recall: for any discrete r.v. \( X \)

\[ E_X [f(x)] \triangleq \sum_x P(x=x) f(x) \]

Q: What is the expected value of a randomly chosen \( \nabla J_i(\theta) \)?

Let \( I \sim \text{Uniform}(1, \ldots, N_3) \)

\[ P(I=i) = \frac{1}{N} \text{ if } i \in \{1, \ldots, N_3\} \]

\[ E_I[\nabla J_I(\theta)] = \sum_{i=1}^{N} P(I=i) \nabla J_i(\theta) \]

\[ \geq \frac{1}{N} \sum_{i=1}^{N} \nabla J_i(\theta) \]

\[ = \nabla J(\theta) \]
Convergence of Optimizers

Convergence Analysis:

**Def:** Convergence is when \( J(\theta) - J(\theta^*) < \epsilon \)

<table>
<thead>
<tr>
<th>Methods</th>
<th>Steps to Converge</th>
<th>Computation per iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newlin's Method</td>
<td>( O(\ln \ln \frac{1}{\epsilon}) )</td>
<td>( \nabla J(\theta) \leftarrow O(N M) )</td>
</tr>
<tr>
<td>GD</td>
<td>( O(\ln \frac{1}{\epsilon}) )</td>
<td>( \nabla J(\theta) \leftarrow O(N M) )</td>
</tr>
<tr>
<td>SGD</td>
<td>( O(\frac{1}{\epsilon}) )</td>
<td>( \nabla J_{i}(\theta) \leftarrow O(M) )</td>
</tr>
</tbody>
</table>

Note: SGD is not correct.

“Almost sure” convergence lots of caveats and conditions

Takeaway: SGD has much slower asymptotic convergence, but is often faster in practice.
Optimization Objectives

You should be able to...

• Apply gradient descent to optimize a function
• Apply stochastic gradient descent (SGD) to optimize a function
• Apply knowledge of zero derivatives to identify a closed-form solution (if one exists) to an optimization problem
• Distinguish between convex, concave, and nonconvex functions
• Obtain the gradient (and Hessian) of a (twice) differentiable function
Linear Regression Objectives

You should be able to...

• Design $k$-NN Regression and Decision Tree Regression

• Implement learning for Linear Regression using three optimization techniques: (1) closed form, (2) gradient descent, (3) stochastic gradient descent

• Choose a Linear Regression optimization technique that is appropriate for a particular dataset by analyzing the tradeoff of computational complexity vs. convergence speed

• Distinguish the three sources of error identified by the bias-variance decomposition: bias, variance, and irreducible error.
PROBABILISTIC LEARNING
Previously, we assumed that our output was generated using a deterministic target function:

\[ x^{(i)} \sim p^*(\cdot) \]
\[ y^{(i)} = c^*(x^{(i)}) \]

Our goal was to learn a hypothesis \( h(x) \) that best approximates \( c^*(x) \)

Today, we assume that our output is sampled from a conditional probability distribution:

\[ x^{(i)} \sim p^*(\cdot) \]
\[ y^{(i)} \sim p^*(\cdot|x^{(i)}) \]

Our goal is to learn a probability distribution \( p(y|x) \) that best approximates \( p^*(y|x) \)
# Robotic Farming

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Probabilistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>Is this a picture of a wheat kernel?</td>
<td>Is this plant drought resistant?</td>
</tr>
<tr>
<td>(binary output)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regression</td>
<td>How many wheat kernels are in this picture?</td>
<td>What will the yield of this plant be?</td>
</tr>
<tr>
<td>(continuous output)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Maximum Likelihood Estimation

The principle of maximum likelihood estimation (MLE):
Choose parameters that make the data “most likely”.

Assumptions: Data generated iid from distribution $p^*(x | \theta^*)$
and comes from a family of data parameterized
$\theta \in \Theta$ set of possible parameters

Formally:

$\theta_{MLE} = \arg \max_{\theta \in \Theta} p(D | \theta)$

$= \arg \max_{\theta \in \Theta} \log p(D | \theta)$

$= \arg \max_{\theta \in \Theta} \ell(\theta)$

where $\ell(\theta) = \log p(D | \theta)$

$log$-likelihood

$treat as function of $\theta$

where $D$ is constant
Learning from Data (Frequentist)

Whiteboard

– Principle of Maximum Likelihood Estimation (MLE)

– Strawmen:
  • Example: Bernoulli
  • Example: Gaussian
  • Example: Conditional #1
    (Bernoulli conditioned on Gaussian)
  • Example: Conditional #2
    (Gaussians conditioned on Bernoulli)
Outline

• Motivation:
  – Choosing the right classifier
  – Example: Image Classification

• Logistic Regression
  – Background: Hyperplanes
  – Data, Model, Learning, Prediction
  – Log-odds
  – Bernoulli interpretation
  – Maximum Conditional Likelihood Estimation

• Gradient descent for Logistic Regression
  – Stochastic Gradient Descent (SGD)
  – Computing the gradient
  – Details (learning rate, finite differences)
MOTIVATION:
LOGISTIC REGRESSION
Example: Image Classification

• ImageNet LSVRC-2010 contest:
  – **Dataset**: 1.2 million labeled images, 1000 classes
  – **Task**: Given a new image, label it with the correct class
  – **Multiclass** classification problem
• Examples from http://image-net.org/
German iris, Iris kochii

Iris of northern Italy having deep blue-purple flowers, similar in size but smaller than Iris germanica.

- Nature (32)
- succulent (32)
- cattleya (32)
- cultivated plant (32)
- weed (32)
- monocot, monocot plant (32)
- dicot plant (32)
- vine (32)
- creeper (32)
- winter plant, fleshy plant (144)
- geophyte (32)
- desert plant, xerophyte, xerophytic plant, xerophyte, xerophytic
- mesophytic plant (32)
- aquatics plant, aquatic plant, nymphaetum, nymphaeal plant (32)
- bulbar plant (32)
- bulbous plants (17)
- i. kochii (27)
- Iris, flag, blue flag, water flag (11)
- I. kochii (27)
- Germaniris, Iris kochii (27)
- German iris, Iris kochii (27)
- German iris, Iris kochii (27)
- I. kochii, I. x. kochii (27)
- I. x. kochii (27)
- I. kochii (27)
- I. kochii (27)
- I. kochii (27)
Court, courtyard

An area wholly or partly surrounded by walls of buildings. "The house was built around an inner court."
Example: Image Classification

CNN for Image Classification
(Krizhevsky, Sutskever & Hinton, 2011)
17.5% error on ImageNet LSVRC-2010 contest

Input image (pixels)

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax

---

Input image (pixels)

- Max pooling
- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax

---

Input image (pixels)

- Max pooling
- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax

---

Input image (pixels)

- Max pooling
- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax

---

Input image (pixels)

- Max pooling
- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax

---

Input image (pixels)

- Max pooling
- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax

---

Input image (pixels)

- Max pooling
- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax

---
**Example: Image Classification**

**CNN for Image Classification**
(Krizhevsky, Sutskever & Hinton, 2011)
17.5% error on ImageNet LSVRC-2010 contest

**Input image (pixels)**

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

**The rest is just some fancy feature extraction (discussed later in the course)**

**This “softmax” layer is Logistic Regression!**

**1000-way softmax**

---

The rest is just some fancy feature extraction (discussed later in the course)

This “softmax” layer is Logistic Regression!
LOGISTIC REGRESSION
Logistic Regression

**Data:** Inputs are continuous vectors of length K. Outputs are discrete.

\[ D = \{x^{(i)}, y^{(i)}\}_{i=1}^{N} \text{ where } x \in \mathbb{R}^M \text{ and } y \in \{0, 1\} \]
**Linear Models for Classification**

Key idea: Try to learn this hyperplane directly

Directly modeling the hyperplane would use a decision function:

\[ h(x) = \text{sign}(\theta^T x) \]

for:

\[ y \in \{-1, +1\} \]
Background: Hyperplanes

Hyperplane (Definition 1):

\[ \mathcal{H} = \{ \mathbf{x} : \mathbf{w}^T \mathbf{x} = b \} \]

Hyperplane (Definition 2):

\[ \mathcal{H} = \{ \mathbf{x} : \mathbf{\theta}^T \mathbf{x} = 0 \text{ and } x_0 = 1 \} \]

\[ \mathbf{\theta} = [b, w_1, \ldots, w_M]^T \]

Half-spaces:

\[ \mathcal{H}^+ = \{ \mathbf{x} : \mathbf{\theta}^T \mathbf{x} > 0 \text{ and } x_0 = 1 \} \]

\[ \mathcal{H}^- = \{ \mathbf{x} : \mathbf{\theta}^T \mathbf{x} < 0 \text{ and } x_0 = 1 \} \]
Using gradient ascent for linear classifiers

Key idea behind today’s lecture:

1. Define a linear classifier (logistic regression)
2. Define an objective function (likelihood)
3. Optimize it with gradient descent to learn parameters
4. Predict the class with highest probability under the model
Using gradient ascent for linear classifiers

This decision function isn’t differentiable:

\[ h(x) = \text{sign}(\theta^T x) \]

Use a differentiable function instead:

\[ p_\theta(y = 1|x) = \frac{1}{1 + \exp(-\theta^T x)} \]

[Graph showing the sign function and logistic function]
Using gradient ascent for linear classifiers

This decision function isn’t differentiable:

\[ h(x) = \text{sign}(\theta^T x) \]

Use a differentiable function instead:

\[ p_\theta(y = 1|x) = \frac{1}{1 + \exp(-\theta^T x)} \]

\[ \text{logistic}(u) \equiv \frac{1}{1 + e^{-u}} \]
Logistic Regression

<table>
<thead>
<tr>
<th>Data:</th>
<th>Inputs are continuous vectors of length K. Outputs are discrete.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mathcal{D} = {x^{(i)}, y^{(i)}}_{i=1}^{N}$ where $x \in \mathbb{R}^M$ and $y \in {0, 1}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model:</th>
<th>Logistic function applied to dot product of parameters with input vector.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$p_\theta(y = 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Learning:</th>
<th>finds the parameters that minimize some objective function.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\theta^* = \arg\min_{\theta} J(\theta)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prediction:</th>
<th>Output is the most probable class.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\hat{y} = \arg\max_{y \in {0,1}} p_\theta(y</td>
</tr>
</tbody>
</table>
Logistic Regression

Whiteboard

– Bernoulli interpretation
– Logistic Regression Model
– Decision boundary
Logistic Regression
Logistic Regression
LEARNING LOGISTIC REGRESSION
Maximum Conditional Likelihood Estimation

**Learning:** finds the parameters that minimize some objective function.

\[
\theta^* = \arg\min_{\theta} J(\theta)
\]

We minimize the *negative* log conditional likelihood:

\[
J(\theta) = -\log \prod_{i=1}^{N} p_{\theta}(y^{(i)} | x^{(i)})
\]

**Why?**

1. We can’t maximize likelihood (as in Naïve Bayes) because we don’t have a joint model \( p(x,y) \)
2. It worked well for Linear Regression (least squares is MCLE)
Maximum Conditional Likelihood Estimation

**Learning**: Four approaches to solving \( \theta^* = \arg\min_{\theta} J(\theta) \)

**Approach 1**: Gradient Descent  
(take larger – more certain – steps opposite the gradient)

**Approach 2**: Stochastic Gradient Descent (SGD)  
(take many small steps opposite the gradient)

**Approach 3**: Newton’s Method  
(use second derivatives to better follow curvature)

**Approach 4**: Closed Form??  
(set derivatives equal to zero and solve for parameters)
Maximum Conditional Likelihood Estimation

Learning: Four approaches to solving $\theta^* = \arg\min_{\theta} J(\theta)$

**Approach 1:** Gradient Descent  
(take larger – more certain – steps opposite the gradient)

**Approach 2:** Stochastic Gradient Descent (SGD)  
(take many small steps opposite the gradient)

**Approach 3:** Newton’s Method  
(use second derivatives to better follow curvature)

**Approach 4:** Closed Form??  
(set derivatives equal to zero and solve for parameters)

Logistic Regression does not have a closed form solution for MLE parameters.
In order to apply GD to Logistic Regression all we need is the gradient of the objective function (i.e. vector of partial derivatives).
Stochastic Gradient Descent (SGD)

We can also apply SGD to solve the MCLE problem for Logistic Regression.

We need a per-example objective:

Let $J(\theta) = \sum_{i=1}^{N} J^{(i)}(\theta)$

where $J^{(i)}(\theta) = - \log p_{\theta}(y^i | x^i)$. 

**Algorithm 1 Stochastic Gradient Descent (SGD)**

1: procedure SGD($\mathcal{D}, \theta^{(0)}$)
2: $\theta \leftarrow \theta^{(0)}$
3: while not converged do
4:     for $i \in \text{shuffle}([1, 2, \ldots, N])$ do
5:         $\theta \leftarrow \theta - \lambda \nabla_{\theta} J^{(i)}(\theta)$
6: return $\theta$
GRADIENT FOR LOGISTIC REGRESSION
Learning for Logistic Regression

Whiteboard

– Partial derivative for Logistic Regression
– Gradient for Logistic Regression
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, ....)
We can also apply SGD to solve the MCLE problem for Logistic Regression.

We need a per-example objective:

\[
J(\theta) = \sum_{i=1}^{N} J^{(i)}(\theta)
\]
where \(J^{(i)}(\theta) = -\log \rho^{(i)}(y^i|x^i)\).
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)
Probabilistic Interpretation of Linear Regression

Whiteboard

– Conditional Likelihood
– Case #1: 1D Linear Regression
– Case #2: Multiple Linear Regression
– Equivalence: Predictions
– Equivalence: Learning