Linear classifiers: 
Overfitting and regularization

CSE 446: Machine Learning
Emily Fox
University of Washington
January 25, 2017

Logistic regression recap
Thus far, we focused on decision boundaries

\[
\text{Score}(x_i) = w_0 h_0(x_i) + w_1 h_1(x_i) + ... + w_D h_D(x_i) = w^T h(x_i)
\]

Relate Score\((x_i)\) to \(\hat{P}(y=+1|x, w)\)?

Compare and contrast regression models

- **Linear regression with Gaussian errors**
  \[
y_i = w^T h(x_i) + \varepsilon_i \quad \varepsilon_i \sim N(0, \sigma^2)
  \Rightarrow p(y|x,w) = N(y; w^T h(x), \sigma^2)
  \]

- **Logistic regression**
  \[
P(y|x, w) = \begin{cases} 
  \frac{1}{1 + e^{-w^T h(x)}} & \text{if } y = +1 \\
  \frac{e^{-w^T h(x)}}{1 + e^{-w^T h(x)}} & \text{if } y = -1
  \end{cases}
  \]}
Understanding the logistic regression model

\[ P(y=+1|x_i, w) = \text{sigmoid}(\text{Score}(x_i)) = \frac{1}{1 + e^{-w^T h(x)}} \]

| Score\((x_i)\) | \(P(y=+1|x_i, w)\) |
|-------------|-----------------|
| 0           | 0.5             |
| -2          | 0.12 < 0.5 \Rightarrow \hat{y} = -1 |
| 2           | 0.88 > 0.5 \Rightarrow \hat{y} = +1 |
| 4           | 0.98 < 1        |

Sentence from review

Input: \(x\)

Predict most likely class

\(\hat{P}(y|x)\) = estimate of class probabilities

If \(\hat{P}(y=+1|x) > 0.5\):

\(\hat{y} = +1\)

Else:

\(\hat{y} = -1\)

Estimating \(\hat{P}(y|x)\) improves interpretability:

- Predict \(\hat{y} = +1\) and tell me how sure you are
Gradient descent algorithm, cont’d

Step 5: Gradient over all data points

$$\frac{\partial \ell \ell (\mathbf{w})}{\partial w_j} = \sum_{i=1}^{N} h_j(x_i) (\mathbbm{1}[y_i = +1] - P(y = +1 | x_i, \mathbf{w}))$$

| $y_i = +1$ | $P(y = +1 | x_i, \mathbf{w}) \approx 1$ | $P(y = +1 | x_i, \mathbf{w}) \approx 0$ |
|-----------|-------------------------------|-------------------------------|
| $y_i = +1$ |                               |                               |
| $y_i = -1$ |                               |                               |
Summary of gradient ascent for logistic regression

\[ \text{init } w^{(1)} = 0 \text{ (or randomly, or smartly), } t = 1 \]

\[ \text{while } || \nabla \ell(w^{(t)}) || > \varepsilon \]

\[ \text{for } j = 0, \ldots, D \]

\[ \text{partial}[j] = \sum_{i=1}^{N} h_j(x_i) \left( 1[y_i = +1] - P(y = +1 | x_i, w^{(t)}) \right) \]

\[ w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \text{ partial}[j] \]

\[ t \leftarrow t + 1 \]

Overfitting in classification
Learned decision boundary

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Coefficient learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0(x)$</td>
<td>1</td>
<td>0.23</td>
</tr>
<tr>
<td>$h_1(x)$</td>
<td>$x[1]$</td>
<td>1.12</td>
</tr>
<tr>
<td>$h_2(x)$</td>
<td>$x[2]$</td>
<td>-1.07</td>
</tr>
</tbody>
</table>

Quadratic features (in 2d)

Note: we are not including cross terms for simplicity

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Coefficient learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0(x)$</td>
<td>1</td>
<td>1.68</td>
</tr>
<tr>
<td>$h_1(x)$</td>
<td>$x[1]$</td>
<td>1.39</td>
</tr>
<tr>
<td>$h_2(x)$</td>
<td>$x[2]$</td>
<td>-0.59</td>
</tr>
<tr>
<td>$h_3(x)$</td>
<td>$(x[1])^2$</td>
<td>-0.17</td>
</tr>
<tr>
<td>$h_4(x)$</td>
<td>$(x[2])^2$</td>
<td>-0.96</td>
</tr>
</tbody>
</table>
**Degree 6 features (in 2d)**

<table>
<thead>
<tr>
<th>Feature ( h_i(x) )</th>
<th>Value</th>
<th>Coefficient learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_0(x) )</td>
<td>1</td>
<td>21.6</td>
</tr>
<tr>
<td>( h_1(x) )</td>
<td>( x[1] )</td>
<td>5.3</td>
</tr>
<tr>
<td>( h_2(x) )</td>
<td>( x[2] )</td>
<td>-42.7</td>
</tr>
<tr>
<td>( h_3(x) )</td>
<td>( (x[1])^2 )</td>
<td>-15.9</td>
</tr>
<tr>
<td>( h_4(x) )</td>
<td>( (x[2])^2 )</td>
<td>-48.6</td>
</tr>
<tr>
<td>( h_5(x) )</td>
<td>( (x[1])^3 )</td>
<td>-11.0</td>
</tr>
<tr>
<td>( h_6(x) )</td>
<td>( (x[2])^3 )</td>
<td>67.0</td>
</tr>
<tr>
<td>( h_7(x) )</td>
<td>( (x[1])^4 )</td>
<td>1.5</td>
</tr>
<tr>
<td>( h_8(x) )</td>
<td>( (x[2])^4 )</td>
<td>48.0</td>
</tr>
<tr>
<td>( h_9(x) )</td>
<td>( (x[1])^5 )</td>
<td>4.4</td>
</tr>
<tr>
<td>( h_{10}(x) )</td>
<td>( (x[2])^5 )</td>
<td>-14.2</td>
</tr>
<tr>
<td>( h_{11}(x) )</td>
<td>( (x[1])^6 )</td>
<td>0.8</td>
</tr>
<tr>
<td>( h_{12}(x) )</td>
<td>( (x[2])^6 )</td>
<td>-8.6</td>
</tr>
</tbody>
</table>

Coefficient values getting large

Score(\( x \)) < 0

**Degree 20 features (in 2d)**

<table>
<thead>
<tr>
<th>Feature ( h_i(x) )</th>
<th>Value</th>
<th>Coefficient learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_0(x) )</td>
<td>1</td>
<td>8.7</td>
</tr>
<tr>
<td>( h_1(x) )</td>
<td>( x[1] )</td>
<td>5.1</td>
</tr>
<tr>
<td>( h_2(x) )</td>
<td>( x[2] )</td>
<td>78.7</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>( h_{11}(x) )</td>
<td>( (x[1])^6 )</td>
<td>-7.5</td>
</tr>
<tr>
<td>( h_{12}(x) )</td>
<td>( (x[2])^6 )</td>
<td>3803</td>
</tr>
<tr>
<td>( h_{13}(x) )</td>
<td>( (x[1])^7 )</td>
<td>21.1</td>
</tr>
<tr>
<td>( h_{14}(x) )</td>
<td>( (x[2])^7 )</td>
<td>-2406</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>( h_{37}(x) )</td>
<td>( (x[1])^{19} )</td>
<td>-2*10^{-6}</td>
</tr>
<tr>
<td>( h_{38}(x) )</td>
<td>( (x[2])^{19} )</td>
<td>-0.15</td>
</tr>
<tr>
<td>( h_{39}(x) )</td>
<td>( (x[1])^{20} )</td>
<td>-2*10^{-8}</td>
</tr>
<tr>
<td>( h_{40}(x) )</td>
<td>( (x[2])^{20} )</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Often, overfitting associated with very large estimated coefficients \( \hat{w} \)
Overfitting in classification

Overfitting if there exists $w^*$:
- $\text{training_error}(w^*) > \text{training_error}(\hat{w})$
- $\text{true_error}(w^*) < \text{true_error}(\hat{w})$

Overfitting in classifiers ➔ Overconfident predictions
Logistic regression model

\[ w^T h(x_i) \]

\[ P(y=+1|x_i, w) = \text{sigmoid}(w^T h(x_i)) \]

The subtle (negative) consequence of overfitting in logistic regression

- Overfitting \( \Rightarrow \) Large coefficient values
- \( \hat{w}^T h(x_i) \) is very positive (or very negative) \( \Rightarrow \) \( \text{sigmoid}(\hat{w}^T h(x_i)) \) goes to 1 (or to 0)
- Model becomes extremely overconfident of predictions
Effect of coefficients on logistic regression model

Input x: #awesome=2, #awful=1

\[
\begin{array}{c|c|c}
\text{Coefficient} & \#\text{awesome} & \#\text{awful} \\
\hline
w_0 & 0 & 0 \\
w_{\#\text{awesome}} & +1 & +2 \\
w_{\#\text{awful}} & -1 & -2 \\
\end{array}
\]

Learned probabilities

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Coefficient learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>(h_0(x))</td>
<td>1</td>
<td>0.23</td>
</tr>
<tr>
<td>(h_1(x))</td>
<td>(x[1])</td>
<td>1.12</td>
</tr>
<tr>
<td>(h_2(x))</td>
<td>(x[2])</td>
<td>-1.07</td>
</tr>
</tbody>
</table>

\[
P(y = +1 \mid x, w) = \frac{1}{1 + e^{-w^T h(x)}}
\]
Quadratic features: Learned probabilities

\[ P(y = +1 \mid x, w) = \frac{1}{1 + e^{-w^\top h(x)}} \]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Coefficient learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_0(x) )</td>
<td>1</td>
<td>1.68</td>
</tr>
<tr>
<td>( h_1(x) )</td>
<td>( x[1] )</td>
<td>1.39</td>
</tr>
<tr>
<td>( h_2(x) )</td>
<td>( x[2] )</td>
<td>-0.58</td>
</tr>
<tr>
<td>( h_3(x) )</td>
<td>( (x[1])^2 )</td>
<td>-0.17</td>
</tr>
<tr>
<td>( h_4(x) )</td>
<td>( (x[2])^2 )</td>
<td>-0.96</td>
</tr>
</tbody>
</table>

Overfitting ➔ Overconfident predictions

Degree 6: Learned probabilities

Degree 20: Learned probabilities

Tiny uncertainty regions

Overfitting & overconfident about it!!!

We are sure we are right, when we are surely wrong!
Overfitting in logistic regression: Another perspective

Linearly-separable data

Data are linearly separable if:
- There exist coefficients $\mathbf{w}$ such that:
  - For all positive training data
  - For all negative training data

Note 1: If you are using D features, linear separability happens in a D-dimensional space

Note 2: If you have enough features, data are (almost) always linearly separable

$\text{training\_error}(\mathbf{w}) = 0$
Effect of linear separability on coefficients

Data are linearly separable with $\hat{w}_1 = 1.0$ and $\hat{w}_2 = -1.5$

Data also linearly separable with $\hat{w}_1 = 10$ and $\hat{w}_2 = -15$

Data also linearly separable with $\hat{w}_1 = 10^9$ and $\hat{w}_2 = -1.5 \times 10^9$

Maximum likelihood estimation (MLE) prefers most certain model →
Coefficients go to infinity for linearly-separable data!!!
Overfitting in logistic regression is “twice as bad”

Learning tries to find decision boundary that separates data

Overly complex boundary

If data are linearly separable

Coefficients go to infinity!

\[ \hat{w}_1 = 10^9 \]
\[ \hat{w}_2 = -1.5 \times 10^9 \]

Penalizing large coefficients to mitigate overfitting
Desired total cost format

Want to balance:

i. How well function fits data
ii. Magnitude of coefficients

Total quality = \[ \text{measure of fit} - \text{measure of magnitude of coefficients} \]

Consider resulting objective

Select \( \hat{w} \) to minimize:

\[
\ell(w) - \lambda \|w\|_2^2
\]

\( L_2 \) regularized logistic regression

Pick \( \lambda \) using:

- Validation set (for large datasets)
- Cross-validation (for smaller datasets)
  (as in ridge/lasso regression)
Degree 20 features, effect of regularization penalty $\lambda$

<table>
<thead>
<tr>
<th>Regularization</th>
<th>$\lambda = 0$</th>
<th>$\lambda = 0.00001$</th>
<th>$\lambda = 0.001$</th>
<th>$\lambda = 1$</th>
<th>$\lambda = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of coefficients</td>
<td>-3170 to 3803</td>
<td>-8.04 to 12.14</td>
<td>-0.70 to 1.25</td>
<td>-0.13 to 0.57</td>
<td>-0.05 to 0.22</td>
</tr>
</tbody>
</table>

Coefficient path

- Disappointed
- Best
- This
- Awesome
- Review
- Hate
- :)

<table>
<thead>
<tr>
<th>Coefficients $\hat{w}_j$</th>
<th>0</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
</tr>
</thead>
<tbody>
<tr>
<td>: ( : )</td>
<td>-3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Best</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>This</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Awesome</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Review</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Hate</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>:)</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Degree 20 features: regularization reduces “overconfidence”

<table>
<thead>
<tr>
<th>Regularization</th>
<th>$\lambda = 0$</th>
<th>$\lambda = 0.00001$</th>
<th>$\lambda = 0.001$</th>
<th>$\lambda = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of coefficients</td>
<td>-3170 to 3803</td>
<td>-8.04 to 12.14</td>
<td>-0.70 to 1.25</td>
<td>-0.13 to 0.57</td>
</tr>
<tr>
<td>Learned probabilities</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Finding best $L_2$ regularized linear classifier with gradient ascent
Understanding contribution of $L_2$ regularization

$$\frac{\partial \ell(w)}{\partial w_j} - 2\lambda w_j$$

<table>
<thead>
<tr>
<th>$w_j &gt; 0$</th>
<th>Impact on $w_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_j &lt; 0$</td>
<td></td>
</tr>
</tbody>
</table>

Summary of gradient ascent for logistic regression + $L_2$ regularization

init $w^{(1)} = 0$ (or randomly, or smartly), $t = 1$

while not converged:

for $j = 0, ..., D$

$\text{partial}[j] = \sum_{i=1}^N h_j(x_i)(\mathbb{1}[y_i = +1] - P(y = +1 | x_i, w^{(t)})$)

$w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \ (\text{partial}[j] - 2\lambda w_j^{(t)})$

t $\leftarrow t + 1$
Summary of overfitting in logistic regression

What you can do now...

- Identify when overfitting is happening
- Relate large learned coefficients to overfitting
- Describe the impact of overfitting on decision boundaries and predicted probabilities of linear classifiers
- Motivate the form of L2 regularized logistic regression quality metric
- Describe what happens to estimated coefficients as tuning parameter $\lambda$ is varied
- Interpret coefficient path plot
- Estimate L2 regularized logistic regression coefficients using gradient ascent
- Describe the use of L1 regularization to obtain sparse logistic regression solutions
Linear classifiers: Scaling up learning via SGD

Stochastic gradient descent: Learning, one data point at a time
Why gradient ascent is slow...

Every update requires a full pass over data

More formally: How expensive is gradient ascent?

Contribution of data point \( x_i, y_i \) to gradient
Every step requires touching every data point!!!

\[
\frac{\partial \ell(w)}{\partial w_j} = \sum_{i=1}^{N} \frac{\partial \ell_i(w)}{\partial w_j}
\]

<table>
<thead>
<tr>
<th>Time to compute contribution of (x_iy_i)</th>
<th># of data points (N)</th>
<th>Total time to compute 1 step of gradient ascent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 millisecond</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>1 second</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>1 millisecond</td>
<td>10 million</td>
<td></td>
</tr>
<tr>
<td>1 millisecond</td>
<td>10 billion</td>
<td></td>
</tr>
</tbody>
</table>

Stochastic gradient ascent

\[ w^{(t)} \rightarrow w^{(t+1)} \rightarrow w^{(t+2)} \rightarrow w^{(t+3)} \rightarrow w^{(t+4)} \]

Update coefficients

Use only small subsets of data

Many updates for each pass over data
Example: Instead of all data points for gradient, use 1 data point only???

**Gradient ascent**

$$\frac{\partial \ell(w)}{\partial w_j} = \sum_{i=1}^{N} \frac{\partial \ell_i(w)}{\partial w_j}$$

**Stochastic gradient ascent**

$$\frac{\partial \ell(w)}{\partial w_j} \approx \frac{\partial \ell_i(w)}{\partial w_j}$$

Stochastic gradient ascent for logistic regression

$\text{init } w^{(1)} = 0, t = 1$

$\text{for } i = 1, \ldots, N$

$\text{for } j = 0, \ldots, D$

$\text{partial}[j] = \prod_{i=1}^{N} h_j(x_i)(1[y_i = +1] - P(y = +1 | x_i, w^{(t)}))$

$w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \text{ partial}[j]$

$t \leftarrow t + 1$
Stochastic gradient for L2-regularized objective

Total derivative
\[
\frac{\partial \ell_i(w)}{\partial w_j} - 2 \lambda w_j
\]

What about regularization term?

Stochastic gradient ascent

Each data point contributes 1/N to regularization

Comparing computational time per step

Gradient ascent

Stochastic gradient ascent

<table>
<thead>
<tr>
<th>Time to compute contribution of $x_i y_i$</th>
<th># of data points (N)</th>
<th>Total time for 1 step of gradient</th>
<th>Total time for 1 step of stochastic gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 millisecond</td>
<td>1000</td>
<td>1 second</td>
<td></td>
</tr>
<tr>
<td>1 second</td>
<td>1000</td>
<td>16.7 minutes</td>
<td></td>
</tr>
<tr>
<td>1 millisecond</td>
<td>10 million</td>
<td>2.8 hours</td>
<td></td>
</tr>
<tr>
<td>1 millisecond</td>
<td>10 billion</td>
<td>115.7 days</td>
<td></td>
</tr>
</tbody>
</table>
Which one is better??? Depends...

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time per iteration</th>
<th>Sensitivity to parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient</td>
<td>Slow for large data</td>
<td>Moderate</td>
</tr>
<tr>
<td>Stochastic gradient</td>
<td>Always fast</td>
<td>Very high</td>
</tr>
</tbody>
</table>

Summary of stochastic gradient

- Tiny change to gradient ascent
- Much better scalability
- Huge impact in real-world
- Very tricky to get right in practice
Why would stochastic gradient ever work???

Gradient is direction of steepest ascent

Gradient is “best” direction, but any direction that goes “up” would be useful
In ML, steepest direction is sum of “little directions” from each data point

\[
\frac{\partial \ell(w)}{\partial w_j} = \sum_{i=1}^{N} \frac{\partial \ell_i(w)}{\partial w_j}
\]

For most data points, contribution points “up”

Stochastic gradient: Pick a data point and move in direction

\[
\frac{\partial \ell(w)}{\partial w_j} \approx \frac{\partial \ell_i(w)}{\partial w_j}
\]

Most of the time, total likelihood will increase
Stochastic gradient ascent:
Most iterations increase likelihood, but sometimes decrease it ➔
**On average, make progress**

```plaintext
until converged
for i=1,...,N
    for j=0,...,D
        \( w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \frac{\partial \ell_i(w)}{\partial w_j} \)
    t \leftarrow t + 1
```

Convergence path
Convergence paths

Stochastic gradient convergence is "noisy"

Gradient usually increases likelihood smoothly.

Stochastic gradient makes "noisy" progress.

Stochastic gradient achieves higher likelihood sooner, but it's noisier.

Total time proportional to # passes over data.
Eventually, gradient catches up

Note: should only trust "average" quality of stochastic gradient

The last coefficients may be really good or really bad!! 😞

Minimize noise:
don’t return last learned coefficients

Output average:

$$
\hat{w} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}
$$
Summary of why stochastic gradient works

- Gradient finds direction of steepest ascent
- Gradient is sum of contributions from each data point
- Stochastic gradient uses direction from 1 data point
- On average increases likelihood, sometimes decreases
- Stochastic gradient has “noisy” convergence

Online learning:
Fitting models from streaming data
Batch vs online learning

Batch learning
• All data is available at start of training time

Online learning
• Data arrives (streams in) over time
  – Must train model as data arrives!

Online learning example: Ad targeting

$t=1$ $t=2$ $t=3$ $t=4$

Data $\rightarrow$ $\hat{W}$

Data $\rightarrow$ $\hat{W}(2)$

Data $\rightarrow$ $\hat{W}(3)$

Data $\rightarrow$ $\hat{W}(4)$

$t=1$ $t=2$ $t=3$ $t=4$

Data $\rightarrow$ $\hat{W}(\text{final})$

$t=1$ $t=2$ $t=3$ $t=4$

User info, page text $\rightarrow$ $\hat{W}(t)$$\rightarrow$$\hat{W}(t+1)$

$\hat{W}$ $\rightarrow$ $\hat{y} = \text{Suggested ads}$

$\hat{y}$ $\rightarrow$ $\hat{W}(t+1)$

$\hat{y}$ $\rightarrow$ $\hat{W}(2)$

$\hat{y}$ $\rightarrow$ $\hat{W}(3)$

$\hat{y}$ $\rightarrow$ $\hat{W}(4)$

$\hat{y}$ $\rightarrow$ $\hat{W}(\text{final})$
Online learning problem

- Data arrives over each time step $t$:
  - Observe input $x_t$
    - Info of user, text of webpage
  - Make a prediction $\hat{y}_t$
    - Which ad to show
  - Observe true output $y_t$
    - Which ad user clicked on

Need ML algorithm to update coefficients each time step!

Stochastic gradient ascent can be used for online learning!!!

- init $w^{(1)}=0$, $t=1$
- Each time step $t$:
  - Observe input $x_t$
  - Make a prediction $\hat{y}_t$
  - Observe true output $y_t$
  - Update coefficients:

  $\textbf{for } j=0,\ldots,D$

  $w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \frac{\partial \ell_t(w)}{\partial w_j}$
Summary of online learning

- Data arrives over time
- Must make a prediction every time new data point arrives
- Observe true class after prediction made
- Want to update parameters immediately

Summary of stochastic gradient descent
What you can do now...

- Significantly speedup learning algorithm using stochastic gradient
- Describe intuition behind why stochastic gradient works
- Apply stochastic gradient in practice
- Describe online learning problems
- Relate stochastic gradient to online learning

Generalized linear models

OPTIONAL
Models for data

• Linear regression with Gaussian errors
  \[ y_i = \mathbf{w}^T \mathbf{h}(\mathbf{x}_i) + \varepsilon_i \quad \varepsilon_i \sim N(0, \sigma^2) \]
  \[ \Rightarrow p(y|x, \mathbf{w}) = N(y; \mathbf{w}^T \mathbf{h}(\mathbf{x}), \sigma^2) \]

• Logistic regression
  \[ P(y|x, \mathbf{w}) = \begin{cases} 
  1 & y = +1 \\
  \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{h}(\mathbf{x})}} & y = -1 
\end{cases} \]

Examples of “generalized linear models”

A generalized linear model has

\[ E[y | \mathbf{x}, \mathbf{w}] = g(\mathbf{w}^T \mathbf{h}(\mathbf{x})) \]

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Link function g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear regression</td>
<td>( \mathbf{w}^T \mathbf{h}(\mathbf{x}) )</td>
<td>identity function</td>
</tr>
<tr>
<td>Logistic regression</td>
<td>( P(y=+1</td>
<td>\mathbf{x}, \mathbf{w}) )</td>
</tr>
<tr>
<td>(assuming y in (0,1)</td>
<td>(mean is different if</td>
<td>(need slight modification</td>
</tr>
<tr>
<td>instead of (-1,1)</td>
<td>y is not in (0,1)</td>
<td>if y is not in (0,1))</td>
</tr>
<tr>
<td>other set of values)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Learning Problems as Expectations

• Minimizing loss in training data:
  - Given dataset: $X^1, \ldots, X^N$
    - Sampled iid from some distribution $p(x)$ on features:
    - Loss function, e.g., hinge loss, logistic loss, ...
    - We often minimize loss in training data:

$\min_w \ell_D(w) = \frac{1}{N} \sum_{j=1}^{N} \ell(w, x^j)$

• However, we should really minimize expected loss on all data:

$\min_w \ell(w) = E_x [\ell(w, x)] = \int p(x) \ell(w, x) dx$

• 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:
  \[ \nabla_w \ell(w) = \nabla_w E_x[\ell(w, x)] = E_x[\nabla_w \ell(w, x)] \]

• “True” gradient ascent rule:
  \[ w^{t+1} = w^t + \eta E_x[\nabla_w \ell(w, x)] \]

• 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:
  \[ \nabla l(w) = E_x[\nabla \ell(w, x)] \approx \frac{1}{N} \sum_{j=1}^N \nabla \ell(w, x_j) \]

  the bigger \( N \), the closer \( \hat{\nabla} \ell \) to \( \nabla \ell \)

• What if we estimate gradient with just one sample???
  – Unbiased estimate of gradient
  – Very noisy!

  \[ N=1 \]

  \[ \nabla \ell(w) \approx \nabla \ell(w) = \nabla \ell(w, x^n) \]

    \[ E_x[\nabla \ell(w)] = E_{x^n}[\nabla \ell(w, x^n)] = \nabla \ell(w) \]

  • Called stochastic gradient ascent (or descent)
    • Among many other names
    • VERY useful in practice!!!