Setup

\[ \hat{y} = f(\hat{a}) \]  
probability estimate

\[ \hat{a} = w \cdot x \]  
linear activation

w weight vector

x feature vector
Purpose of transfer function

- Transfers linear activation $\rightarrow$ probability

- Logistic regression:
  uses probability estimates $\hat{y} = h(\hat{a}) = \frac{\exp(\hat{a})}{1+\exp(\hat{a})}$

- Will use other non-decreasing functions

  $h : \mathbb{R} \rightarrow [0, 1]$
Data?

Examples \((x_t, y_t)\)
- Feature vectors \(x_t\) typically binary
- Label \(y_t\) “true” probability
  Typically binary
Logistic loss

Estimate: \( \hat{y} = h(\mathbf{w} \cdot \mathbf{x}) = \frac{\exp(\mathbf{w} \cdot \mathbf{x})}{1 + \exp(\mathbf{w} \cdot \mathbf{x})} \)

Loss: \( \text{loss}(y, \hat{y}) = y \ln \frac{y}{\hat{y}} + (1 - y) \ln \frac{1 - y}{1 - \hat{y}} \)

\( y \in \{0, 1\} \quad \Rightarrow \quad \left\{ \begin{array}{ll}
- \ln(1 - \hat{y}) = \ln(1 + \exp(\mathbf{w} \cdot \mathbf{x})) & \text{if } y = 0 \\
- \ln \hat{y} = \ln(1 + \exp(\mathbf{w} \cdot \mathbf{x})) - \mathbf{w} \cdot \mathbf{x} & \text{if } y = 1
\end{array} \right. \)

= negative log likelihood
Crucial property

\[
\frac{\partial}{\partial \mathbf{w}} \text{loss}(y, h(\mathbf{w} \cdot \mathbf{x})) = \left( h(\mathbf{w} \cdot \mathbf{x}) - y \right) \mathbf{x}
\]

(delta rule)

Derivatives for sum of examples = 0

\[\sum_t \hat{y}_t \chi_{t,i} = \sum_t y_t \chi_{t,i} \quad \text{for all features } i\]

est. prob of 1 when i on

true prob of 1 when i on
Outline

3 Overfitting

4 Regularization
### Danger of enforcing constraints

**One feature - inseparable**

<table>
<thead>
<tr>
<th>$y_t$</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{t,1}$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

**Add one sparse feature - separable**

<table>
<thead>
<tr>
<th>$y_t$</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{t,1}$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>$x_{t,2}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Adding sparse feature

Top row - Logic regression on original data set
Bottom row - ditto after feature was added
Adding sparse features

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Trivial to find sol if each example has private feature
Classical case of overfitting
data with sparse features added

Our data set: 1.6K features, 150K examples
(Typical datasets: $10^9$ features, $10^{12}$ examples)
data with sparse features added (2)

Sparse noisy features improve training performance but degrade test performance
Optimization versus machine learning

Not enough to minimize a convex loss function
In machine learning minimize

\[ \text{convex regularization} + \eta \text{ convex loss} \]

Goal: improve test performance
Standard Fix: throw more data at it

Sparse feature easily overfit

more examples always help eventually not enough

more features danger of overfitting
Outline

3 Overfitting

4 Regularization
Weights need to be “controlled”

- Early stopping of training algorithm
  - No large weights in logistic regression because \( \text{sigmoid}(10) = 1 \)
- Clip the weights
- Regularize with \( \sum_i w_i^2 \)
- Regularize with \( \sum_i |w_i| \) or relative entropies
- Feature selection
- New trick: clip range of labels \( y_t \)
Canonical hard example for $\sum_i w_i^2$

Random $n \times n$ matrix

Target is one of the rows

Any alg. that predicts with linear combination of instances has error half after seeing half of the examples

- Lower bound hold for linear (and logistic?) regression based on gradient descent or $\sum_i w_i^2$ regularization
- Solution is unit vector that picks out the right row
- Additional features/kernels don’t help
- Easy to learn with 1-norm (or entropic regularization)
### 1 versus 2 norm - $n = 256$

|       | $\sum_i w_i^2$ | $\sum_i |w_i|$ |
|-------|----------------|----------------|
| lin.regr. | ![Graph](image1.png) | ![Graph](image2.png) |
| log.regr.  | ![Graph](image3.png) | ![Graph](image4.png) |
1 versus 2 norm

- 1 norm solution sparser
- 2 norm solution lots of small weights on random features
- 1 norm solution slightly smaller loss on test set
Regularization by clipping labels

Top row: logistic regression w. single feature
Middle row: logistic regression w. added feature
Bottom row: logistic regression w. added feature and clipped labels

\[ y = 1 \quad \rightarrow \quad y = \text{high value} \]
\[ y = 0 \quad \rightarrow \quad y = \text{low value} \]

Clipping ameliorates the negative effect of sparse features
Better way: clip average features in derivative equations
Clipping the data with added sparse features

Added \(\approx\) third more features with 5 ones each

Training based on clipped labels
- rescaled predictions achieve smaller logistic loss on test set
Clipping is "new" method for preventing overfitting with sparse features