Topics

• Generalized linear models (cont’d)
  – logistic regression
  – gradient ascent, learning rate, convergence, examples
  – additive models, neural networks, back-propagation

• Regularization
  – basic idea
  – effective number of parameters
Review: logistic regression

- In a logistic regression model the conditional probability of the label $y$ given the input example $x$ is expressed as

$$P(y = 1|x, w) = g \left( w_0 + w_1 x_1 + \ldots + w_d x_d \right)$$

where the coefficients $w$ are the adjustable parameters.

The “squashing function”

$$g(z) = \left(1 + \exp(-z)\right)^{-1}$$

known as the logistic function turns linear predictions into probabilities.
Example problem

- The problem: classification of radar returns from the ionosphere (data is available from the UCI ML repository)
  - binary class label
  - 34 input “features” (2 values per radar pulse) defining the input vector \( \mathbf{x} = [x_1, \ldots, x_{34}]^T \).
  - 200 training and 150 testing examples

- We would like to estimate a simple logistic regression model for this classification task

\[
P(y = 1|\mathbf{x}, \mathbf{w}) = g \left( w_0 + w_1 x_1 + \ldots + w_d x_d \right)
\]

where \( d = 34 \).
Fitting logistic regression models

- As in the case of linear regression models we can fit the logistic models using the maximum log-likelihood criterion

\[
l(D; w) = \sum_{i=1}^{n} \log P(y_i | x_i, w)
\]

where

\[
P(y = 1 | x, w) = g \left( w_0 + w_1 x_1 + \ldots + w_d x_d \right)
\]

- The log-likelihood function \( l(D; w) \) is a concave function of the parameters \( w \); a number of optimization techniques are available for finding the maximizing parameters
Gradient ascent

- We can maximize the log-likelihood by iteratively adjusting the parameters in small increments.

In each iteration we adjust $w$ slightly in the direction that increases the log-likelihood (towards the gradient):

$$w \leftarrow w + \epsilon \frac{\partial}{\partial w} \sum_{i=1}^{n} \log P(y_i|x_i, w)$$

$$= \cdots$$

$$= w + \epsilon \sum_{i=1}^{n} \left( y_i - P(y_i = 1|x_i, w) \right) \begin{bmatrix} 1 \\ x_i \end{bmatrix}$$

where $\epsilon$ is the learning rate.
Gradient ascent cont’d

- To understand the procedure graphically we can focus on a single example

\[ \mathbf{w} \leftarrow \mathbf{w} + \epsilon \left( y_i - P(y_i = 1|x_i, \mathbf{w}) \right) \left[ \begin{array}{c} 1 \\ x_i \end{array} \right] \]

prediction error
Setting the learning rate: Armijo rule

The learning rate in

\[ w \leftarrow w + \epsilon \frac{\partial}{\partial w} l(D; w) \]

“should” satisfy

\[
l \left( D; w + \epsilon \frac{\partial}{\partial w} l(D; w) \right) - l(D; w) \geq \epsilon \cdot \frac{1}{2} \| \frac{\partial}{\partial w} l(D; w) \|^2
\]

The Armijo rule suggests finding the smallest integer \( m \) such that \( \epsilon = \epsilon_0 q^m \), \( q < 1 \) is a valid choice in this sense.

- Armijo rule is guaranteed to converge to a (local) maximum under certain technical assumptions
Example cont’d

- We get a monotonically increasing log-likelihood of the training labels as a function of the gradient ascent iterations.

- The resulting error rate on the (independent) test set is 9.3%.

Tommi Jaakkola, MIT AI Lab
Gradient ascent: convergence

- The gradient ascent learning method converges when there is no incentive to move the parameters in any particular direction:

\[
\sum_{i=1}^{n} \left( y_i - P(y_i = 1|x_i, \hat{w}) \right) \begin{bmatrix} 1 \\ x_i \end{bmatrix} = 0
\]

- This condition means again that the prediction error is decorrelated with the components of the input vector.
Additive models and classification

- Similarly to linear regression models, we can extend the logistic regression models to additive (logistic) models

\[ P(y = 1|x, w) = g \left( w_0 + w_1 \phi_1(x) + \ldots w_m \phi_m(x) \right) \]

- We are again free to choose the basis functions \( \phi_i(x) \)
Two layer neural network model

- In a neural network model, the basis functions themselves are adjustable (e.g., squashed linear regression models) representing the probability that a “feature” is present in the input

\[ P(y = 1 | x, w) = g \left( w_0 + w_1 \phi_1(x) + \ldots + w_m \phi_m(x) \right) \]

\[ \phi_m(x) = g \left( w_{m0} + w_{m1}x_1 + w_{m2}x_2 \right) \]
Computing the gradient: back-propagation

Let \( z, z_i, i = 1, \ldots, m \) be the total “input” to each “node” computed in response to a training example \( x \)

\[
\begin{align*}
z &= w_0 + w_1 g(z_1) + \ldots + w_m g(z_m) \\
z_i &= w_{i0} + w_{i1} x_1 + w_{i2} x_2, \quad i = 1, \ldots, m
\end{align*}
\]
Back-propagation cont’d

- We can propagate the derivatives with respect to the inputs

\[
\delta = \frac{\partial}{\partial z} \log P(y|x, w)
\]

\[
\delta_i = \frac{\partial}{\partial z_i} \log P(y|x, w)
\]

\[
= \frac{\partial g(z_i)}{\partial z_i} \times \frac{\partial z}{\partial g(z_i)} \times \frac{\partial}{\partial z} \log P(y|x, w)
\]

\[
= g'(z_i) \times w_i \times \delta
\]

![Diagram of a neural network with inputs, hidden layers, and output. The output is a function of the inputs and weights, with derivatives highlighted.]
We can propagate the derivatives with respect to the inputs

\[
\delta = \frac{\partial}{\partial z} \log P(y|\mathbf{x}, \mathbf{w})
\]

\[
\delta_i = \frac{\partial}{\partial z_i} \log P(y|\mathbf{x}, \mathbf{w})
\]

\[
= \frac{\partial g(z_i)}{\partial z_i} \times \frac{\partial z}{\partial g(z_i)} \times \frac{\partial}{\partial z} \log P(y|\mathbf{x}, \mathbf{w})
\]

\[
= g'(z_i) \times w_i \times \delta
\]

The derivatives with respect to the weights \(w_{ij}\) are obtained from \(\delta\)'s

\[
\frac{\partial}{\partial w_{ij}} \log P(y|\mathbf{x}, \mathbf{w}) = \frac{\partial z_i}{\partial w_{ij}} \times \frac{\partial}{\partial z_i} \log P(y|\mathbf{x}, \mathbf{w}) = x_j \times \delta_i
\]
Topics

- Regularization
  - basic idea
  - effective number of parameters
The key idea ... is to limit “choices”

Questions to answer:
1. What are the “choices”? 
2. How do we limit the choices? 
3. Why do we need to limit the choices? (next lecture)
Example

- The set of (0/1) coins parameterized by the probability $p$ of getting “1”

How many coins are there?
Example

- The set of (0/1) coins parameterized by the probability $p$ of getting “1”

  How many coins are there?

  Case 1: $\infty$
Example

- The set of (0/1) coins parameterized by the probability $p$ of getting “1”

How many coins are there?

Case 1: $\infty$
Case 2: 9 coins $(p_1, \ldots, p_9)$ so that predictions of any other coin (indexed by $p$) is no more than $\epsilon = 0.1$ away

for any $p$, $|p - p_j| \leq \epsilon$ for at least one $j$
Example

- The set of (0/1) coins parameterized by the probability $p$ of getting “1”

How many coins are there?

- Case 1: $\infty$
- Case 2: 9 coins $(p_1, \ldots, p_9)$ so that predictions of any other coin (indexed by $p$) is no more than $\epsilon = 0.1$ away for any $p$, $|p - p_j| \leq \epsilon$ for at least one $j$
- Case 3: only 1 coin if $\epsilon = 0.5$
Logistic regression example

- Simple logistic regression model

\[ P(y = 1|x, \mathbf{w}) = g(w_0 + w_1x) \]

parameterized by \( \mathbf{w} = (w_0, w_1) \). We assume that \( x \in [-1, 1] \), i.e., that the inputs remain bounded.

- We can now divide the parameter space into regions with centers \( \mathbf{w}_1, \mathbf{w}_2, \ldots \) such that the predictions of any \( \mathbf{w} \) (for any \( x \in [-1, 1] \)) are close to those of one of the centers:

\[ | \log P(y = 1|x, \mathbf{w}) - \log P(y = 1|x, \mathbf{w}_j) | \leq \epsilon \]
By constraining $\|w\| \leq C$ for some regularization parameter $C$, we have fewer effective parameter choices in the logistic regression model

$$P(y = 1| x, w) = g(w_0 + w_1 x)$$
We can also regularize by imposing a penalty in the estimation criterion that encourages $\|\mathbf{w}\|$ to remain small.

Maximum penalized likelihood

$$l(D; \mathbf{w}, \lambda) = \sum_{i=1}^{n} \log P(y_i | \mathbf{x}_i, \mathbf{w}) - \frac{\lambda}{2} \|\mathbf{w}\|^2$$

where larger values of $\lambda$ impose stronger regularization.