Outline

1. Administration
2. Review of last lecture
3. Logistic regression
A few announcements

- Homework 1: due 9/24 (see the homework sheets for detailed submission information)
- A lot of goodies on the Discussion Board: please participate or at least browse!
- Typos (corrected versions uploaded already or soon)
  - Lecture 2, slide# 39
    \[
    \sum_{d}(x_d - x_{nd})^p \rightarrow \sum_{d}|x_d - x_{nd}|^p
    \]
  - Lecture 4, slide#19: missing a “+” in eq.(7)
  - Lecture 4, slide#20: missing an extra subscript
    \[
    \sum_{n:y_n=c} \rightarrow \sum_{n:y_n=c,k}
    \]
Outline

1 Administration

2 Review of last lecture
   • Naive Bayes

3 Logistic regression
Naive Bayes

Assume $X \in \mathbb{R}^D$ and all $X_d \in [K]$

$$P(X = x, Y = c) = P(Y = c) \prod_k P(k|Y = c)^{z_k} = \pi_c \prod_k \theta_{ck}^{z_k}$$

where $z_k$ is the number of times $k$ in $x$.

Key assumption made

- Conditional independence:
  $$P(X_i, X_j|Y = c) = P(X_i|Y = c)P(X_j|Y = c).$$

- $P(X_i|Y = c)$ depends only the value of $X_i$, not $i$ itself (order of words does not matter in “bag-of-word” representation of documents)
Learning problem

Training data

\[ \mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N} \rightarrow \mathcal{D} = \{\{\{z_{nk}\}_{k=1}^{K}, y_n\}\}_{n=1}^{N} \]

maximum likelihood estimation

\[ \mathcal{L} = \sum_{n} \log \pi_{y_n} + \sum_{n,k} z_{nk} \log \theta_{ynk} \]

\[ \pi_{c}^* = \frac{\text{# of data points labeled as } c}{N} \]

\[ \theta_{ck}^* = \frac{\sum_{n:y_n=c} z_{nk}}{\sum_{k} \sum_{n:y_n=c} z_{nk}} \]
Given an unlabeled data point $x = \{z_k, k = 1, 2, \cdots, K\}$, label it with

$$y^* = \arg \max_{c \in [C]} P(y = c | x)$$

$$= \arg \max_{c \in [C]} P(y = c) P(x | y = c)$$

$$= \arg \max_{c} \left[ \log \pi_c + \sum_k z_k \log \theta_{ck} \right]$$
Moving forward

Examine the classification rule for naive Bayes

\[ y^* = \arg \max_c \log \pi_c + \sum \log \theta_{ck} \]

For binary classification problem, this is just to determine the label basing on

\[ \log \pi_1 + \sum \log \theta_{1k} - \left( \log \pi_2 + \sum \log \theta_{2k} \right) \]

This is just a linear function of the features \( \{z_k\} \)

\[ w_0 + \sum z_k w_k \]

where we “absorb” \( w_0 = \log \pi_1 - \log \pi_2 \) and \( w_k = \log \theta_{1k} - \log \theta_{2k} \).
Naive Bayes is a linear classifier

Fundamentally, what really matters in deciding decision boundary is

$$w_0 + \sum_k z_k w_k$$

Namely, if this quantity is greater than 0, then class 1, else class 2.

The linear form motivates many new methods. One of them is logistic regression.
Outline

1 Administration

2 Review of last lecture

3 Logistic regression
   - General setup
   - Maximum likelihood estimation
   - Numerical optimization
   - Gradient descent
   - Gradient descent for logistic regression
   - Newton method
   - Generative versus discriminative
Logistic regression

### Logistic classification

#### Setup for two classes

- **Input:** $\mathbf{x} \in \mathbb{R}^D$
- **Output:** $y \in \{0, 1\}$
- **Training data:** $\mathcal{D} = \{(\mathbf{x}_n, y_n), n = 1, 2, \ldots, N\}$
- **Model:**

  $$p(y = 1 | \mathbf{x}; b, \mathbf{w}) = \sigma [g(\mathbf{x})]$$

  where

  $$g(\mathbf{x}) = b + \sum_d w_d x_d = b + \mathbf{w}^T \mathbf{x}$$

  and $\sigma[\cdot]$ stands for the **sigmoid** function

  $$\sigma(a) = \frac{1}{1 + e^{-a}}$$
Logistic regression

Why the sigmoid function?

What does it look like?

\[ \sigma(a) = \frac{1}{1 + e^{-a}} \]

where

\[ a = b + \mathbf{w}^T \mathbf{x} \]
Why the sigmoid function?

What does it look like?

\[ \sigma(a) = \frac{1}{1 + e^{-a}} \]

where

\[ a = b + \mathbf{w}^T \mathbf{x} \]

Properties

- Bounded between 0 and 1 \( \Leftarrow \) thus, interpretable as probability
- Monotonically increasing \( \Leftarrow \) thus, usable to derive classification rules
  1. \( \sigma(a) > 0.5 \), positive (classify as '1')
  2. \( \sigma(a) < 0.5 \), negative (classify as '0')
  3. \( \sigma(a) = 0.5 \), undecidable
- Nice computationally properties \( \text{These will unfold in the next few slides} \)
Logistic regression

General setup

Linear or nonlinear?

\( \sigma(a) \) is nonlinear, however, the decision boundary is determined by

\[
\sigma(a) = 0.5 \Rightarrow a = 0 \Rightarrow g(x) = b + w^T x = 0
\]

which is a *linear* function in \( x \).

We often call \( b \) the bias term.
Contrast Naive Bayes and our new model

**Similar**

Both look at the linear function of features for classification

**Difference**

Naive Bayes models the *joint* distribution

\[ P(X, Y) = P(Y)P(X|Y) \]

Logistic regression models the *conditional* distribution

\[ P(Y|X) \]
Likelihood function

**Probability of a single training sample** \((x_n, y_n)\)

\[
p(y_n | x_n; b; w) = \begin{cases} 
\sigma(b + w^T x_n) & \text{if } y_n = 1 \\
1 - \sigma(b + w^T x_n) & \text{otherwise}
\end{cases}
\]
### Likelihood function

**Probability of a single training sample** \((x_n, y_n)\)

\[
p(y_n|x_n; b; w) = \begin{cases} 
\sigma(b + w^T x_n) & \text{if } y_n = 1 \\
1 - \sigma(b + w^T x_n) & \text{otherwise}
\end{cases}
\]

**Compact expression, exploring that** \(y_n\) **is either 1 or 0**

\[
p(y_n|x_n; b; w) = \sigma(b + w^T x_n)^{y_n} [1 - \sigma(b + w^T x_n)]^{1-y_n}
\]
Cross-entropy error

Log-likelihood of the whole training data $\mathcal{D}$

$$\log P(\mathcal{D}) = \sum_n \{ y_n \log \sigma(b + \mathbf{w}^T \mathbf{x}_n) + (1 - y_n) \log[1 - \sigma(b + \mathbf{w}^T \mathbf{x}_n)] \}$$
Logistic regression

Maximum likelihood estimation

Cross-entropy error

Log-likelihood of the whole training data $\mathcal{D}$

$$\log P(\mathcal{D}) = \sum_{n} \{y_n \log \sigma(b + w^T x_n) + (1 - y_n) \log[1 - \sigma(b + w^T x_n)]\}$$

It is convenient to work with its negation, which is called \textit{cross-entropy error function}

$$\mathcal{E}(b, w) = -\sum_{n} \{y_n \log \sigma(b + w^T x_n) + (1 - y_n) \log[1 - \sigma(b + w^T x_n)]\}$$
Shorthand notation

*This is for convenience*

- Append 1 to $\mathbf{x}$
  \[
  \mathbf{x} \leftarrow [1 \ x_1 \ x_2 \ \cdots \ x_D]
  \]
- Append $b$ to $\mathbf{w}$
  \[
  \mathbf{w} \leftarrow [b \ w_1 \ w_2 \ \cdots \ w_D]
  \]
- Cross-entropy is then
  \[
  \mathcal{E}(\mathbf{w}) = -\sum_n \{y_n \log \sigma(\mathbf{w}^T \mathbf{x}_n) + (1 - y_n) \log[1 - \sigma(\mathbf{w}^T \mathbf{x}_n)]\}
  \]

*NB.* We are not using the $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{w}}$ (as in several textbooks) for cosmetic reasons.
How to find the optimal parameters for logistic regression?

We will minimize the error function

\[ E(w) = -\sum_n \{ y_n \log \sigma(w^T x_n) + (1 - y_n) \log[1 - \sigma(w^T x_n)] \} \]

However, this function is complex and we cannot find the simple solution as we did in Naive Bayes. So we need to use numerical methods.

- Numerical methods are messier, in contrast to cleaner analytic solutions.
- In practice, we often have to tune a few optimization parameters — patience is necessary.
An overview of numerical methods

We describe two

- Gradient descent (our focus in lecture): simple, especially effective for large-scale problems
- Newton method: classical and powerful method

Gradient descent is often referred to as an *first-order* method as it requires only to compute the gradients (i.e., the first-order derivative) of the function.

In contrast, Newton method is often referred as to an *second-order* method.
Example: \( \min f(\theta) = 0.5(\theta_1^2 - \theta_2)^2 + 0.5(\theta_1 - 1)^2 \)

- We compute the gradients
  \[
  \frac{\partial f}{\partial \theta_1} = 2(\theta_1^2 - \theta_2)\theta_1 + \theta_1 - 1 \quad (5)
  \]
  \[
  \frac{\partial f}{\partial \theta_2} = -(\theta_1^2 - \theta_2) \quad (6)
  \]

- Use the following \textit{iterative} procedure for \textit{gradient descent}
  1. Initialize \( \theta_1^{(0)} \) and \( \theta_2^{(0)} \), and \( t = 0 \)
  2. do
     \[
     \theta_1^{(t+1)} \leftarrow \theta_1^{(t)} - \eta \left[ 2(\theta_1^{(t)})^2 - \theta_2^{(t)} \right] \theta_1^{(t)} + \theta_1^{(t)} - 1 \quad (7)
     \]
     \[
     \theta_2^{(t+1)} \leftarrow \theta_2^{(t)} - \eta \left[ -(\theta_1^{(t)})^2 - \theta_2^{(t)} \right] \quad (8)
     \]
     \[
     t \leftarrow t + 1 \quad (9)
     \]
  3. until \( f(\theta^{(t)}) \) does not change much
Gradient descent

**General form for minimizing** $f(\theta)$

$$\theta^{t+1} \leftarrow \theta - \eta \frac{\partial f}{\partial \theta}$$

**Remarks**

- $\eta$ is often called *step size* – literally, how far our update will go along the direction of the negative gradient.
- Note that this is for *minimizing* a function, hence the subtraction ($-\eta$).
- With a *suitable* choice of $\eta$, the iterative procedure converges to a stationary point where
  $$\frac{\partial f}{\partial \theta} = 0$$
- A stationary point is only necessary for being the minimum.
Seeing in action

Choose the right $\eta$ is important

small $\eta$ is too slow?
Seeing in action

Choose the right $\eta$ is important

small $\eta$ is too slow? 

large $\eta$ is too unstable?
How do we do this for logistic regression?

**Simple fact: derivatives of $\sigma(a)$**

\[
\frac{d \sigma(a)}{d a} = \frac{d}{d a} \left( \frac{1}{1 + e^{-a}} \right) = \frac{-(1 + e^{-a})'}{(1 + e^{-a})^2}
\]
How do we do this for logistic regression?

**Simple fact: derivatives of $\sigma(a)$**

\[
\frac{d \sigma(a)}{d a} = \frac{d}{d a} \left( \frac{1}{1 + e^{-a}} \right) = \frac{-(1 + e^{-a})'}{(1 + e^{-a})^2}
\]

\[
= \frac{e^a}{(1 + e^{-a})^2} = \frac{1}{1 + e^{-a}} \left( 1 - \frac{1}{1 + e^{-a}} \right)
\]
How do we do this for logistic regression?

Simple fact: derivatives of $\sigma(a)$

$$
\frac{d \sigma(a)}{d a} = \frac{d}{d a} \left( \frac{1}{1 + e^{-a}} \right) = \frac{-(1 + e^{-a})'}{(1 + e^{-a})^2} = \frac{e^a}{(1 + e^{-a})^2} = \frac{1}{1 + e^{-a}} \left( 1 - \frac{1}{1 + e^{-a}} \right) = \sigma(a)[1 - \sigma(a)]
$$
Gradients of the cross-entropy error function

**Gradients**

\[ \frac{\partial E(w)}{\partial w} = - \sum_n \left\{ y_n [1 - \sigma(w^T x_n)] x_n - (1 - y_n) \sigma(w^T x_n)] x_n \right\} \quad (10) \]

\[ = \sum_n \left\{ \sigma(w^T x_n) - y_n \right\} x_n \quad (11) \]

**Remarks**

- \( e_n = \left\{ \sigma(w^T x_n) - y_n \right\} \) is called error for the \( n \)th training sample.
- Stationary point (in this case, the optimum):

\[ \sum_n \sigma(w^T x_n) x_n = \sum_n x_n y_n \]

**Intuition:** on average, the error is zero.
Numerical optimization

Gradient descent

- Choose a proper step size $\eta > 0$
Numerical optimization

Gradient descent

- Choose a proper step size $\eta > 0$
- Iteratively update the parameters following the negative gradient to minimize the error function

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \sum_n \{ \sigma(\mathbf{w}^T \mathbf{x}_n) - y_n \} \mathbf{x}_n$$

Remarks

- The step size needs to be chosen carefully to ensure convergence.
- The step size can be adaptive (i.e. varying from iteration to iteration). For example, we can use techniques such as line search
- There is a variant called stochastic gradient descent, also popularly used (later in this semester).
Intuition for Newton method

Approximate the true function with an easy-to-solve optimization problem
Approximation

**Taylor expansion of the cross-entropy function**

\[ E(w) \approx E(w^{(t)}) + (w - w^{(t)})^T \nabla E(w^{(t)}) + \frac{1}{2} (w - w^{(t)})^T H^{(t)} (w - w^{(t)}) \]

where

- \( \nabla E(w^{(t)}) \) is the gradient
- \( H^{(t)} \) is the Hessian matrix evaluated at \( w^{(t)} \)

**Example: a scalar function**

\[ \sin(\theta) \approx \sin(0) + \theta \cos(\theta = 0) + \frac{1}{2} \theta^2 [- \sin(\theta = 0)] = \theta \]

where \( \nabla \sin(\theta) = \cos(\theta) \) and \( H = \nabla \cos(\theta) = - \sin(\theta) \)
So what is the Hessian matrix?

The matrix of second-order derivatives

\[ H = \frac{\partial^2 \mathcal{E}(w)}{\partial w w^T} \]

In other words,

\[ H_{ij} = \frac{\partial}{\partial w_j} \left( \frac{\partial \mathcal{E}(w)}{\partial w_i} \right) \]

So the Hessian matrix is \( \mathbb{R}^{D \times D} \), where \( w \in \mathbb{R}^D \).
Optimizing the approximation

Minimize the approximation

$$\mathcal{E}(w) \approx \mathcal{E}(w^{(t)}) + (w - w^{(t)})^T \nabla \mathcal{E}(w^{(t)}) + \frac{1}{2}(w - w^{(t)})^T H^{(t)}(w - w^{(t)})$$

and use the solution as the new estimate of the parameters

$$w^{(t+1)} \leftarrow \min_w (w - w^{(t)})^T \nabla \mathcal{E}(w^{(t)}) + \frac{1}{2}(w - w^{(t)})^T H^{(t)}(w - w^{(t)})$$
Optimizing the approximation

Minimize the approximation

\[
E(w) \approx E(w^{(t)}) + (w - w^{(t)})^T \nabla E(w^{(t)}) + \frac{1}{2} (w - w^{(t)})^T H^{(t)} (w - w^{(t)})
\]

and use the solution as the new estimate of the parameters

\[
w^{(t+1)} \leftarrow \min_w (w - w^{(t)})^T \nabla E(w^{(t)}) + \frac{1}{2} (w - w^{(t)})^T H^{(t)} (w - w^{(t)})
\]

The quadratic function minimization has a \textit{closed} form, thus, we have

\[
w^{(t+1)} \leftarrow w^{(t)} - \left( H^{(t)} \right)^{-1} \nabla E(w^{(t)})
\]

i.e., the Newton method.
Contrast gradient descent and Newton method

**Similar**

Both are iterative procedures.

**Difference**

- Newton method requires second-order derivatives.
- Newton method does not have the magic $\eta$ to be set.
Other important things about Hessian

Our cross-entropy error function is convex

\[
\frac{\partial E(w)}{\partial w} = \sum_n \{\sigma(w^T x_n) - y_n\} x_n
\]

\[\Rightarrow H = \frac{\partial^2 E(w)}{\partial w^T w} = \text{homework}\]
Other important things about Hessian

Our cross-entropy error function is convex

\[
\frac{\partial E(w)}{\partial w} = \sum_n \{\sigma(w^T x_n) - y_n\} x_n \tag{12}
\]

\[
H = \frac{\partial^2 E(w)}{\partial w\partial w^T} = \text{homework} \tag{13}
\]

For any vector \( v \),

\[
v^T Hv = \text{homework} \geq 0
\]

Thus, positive definite. Thus, the cross-entropy error function is convex, with only one global optimum.
Good about Newton method

Fast!

Suppose we want to minimize $f(x) = x^2 + 2x$ and we have its current estimate at $x^{(t)} \neq -1$. So what is the next estimate?

$$x^{(t+1)} \leftarrow x^{(t)} - [f''(x)]^{-1} f'(x) = x^{(t)} - \frac{1}{2}(2x^{(t)} + 2) = -1$$

Namely, the next step (of iteration) immediately tells us the global optimum! (In optimization, this is called *superlinear convergence rate*).

In general, the better our approximation, the fast the Newton method is in solving our optimization problem.
Bad about Newton method

Not scalable!

- Computing and inverting Hessian matrix can be very expensive for large-scale problems where the dimensionally $D$ is very large.
- Newton method does not guarantee convergence if your starting point is far away from the optimum

*NB. There are fixes and alternatives, such as Quasi-Newton/Quasi-second order method.*
Generative versus discriminative: two different modeling paradigms

Naive Bayes and logistic regression highlight the differences:

- Setup
  Suppose the training data is from a joint probabilistic model $p(x, y)$
- Differences in specifying models
  - the generative approach requires we specify the model for the joint distribution (such as Naive Bayes), and thus, maximize the \textit{joint} likelihood $\sum_n \log p(x_n, y_n)$
  - the discriminative approach (discriminative) requires only specifying a model for the conditional distribution (such as logistic regression), and thus, maximize the \textit{conditional} likelihood $\sum_n \log p(y_n|x_n)$
Generative versus discriminative: two different modeling paradigms

Naive Bayes and logistic regression highlight the differences:

- **Setup**
  Suppose the training data is from a joint probabilistic model $p(x, y)$

- **Differences in specifying models**
  - the generative approach requires we specify the model for the joint distribution (such as Naive Bayes), and thus, maximize the **joint** likelihood $\sum_n \log p(x_n, y_n)$
  - the discriminative approach (discriminative) requires only specifying a model for the conditional distribution (such as logistic regression), and thus, maximize the **conditional** likelihood $\sum_n \log p(y_n | x_n)$

- **Differences in computation**
  - Sometimes, modeling by discriminative approach is easier
  - Sometimes, parameter estimation by generative approach is easier