Today:
• Logistic regression
• Generative/Discriminative classifiers

Readings: (see class website)

Required:
• Mitchell: “Naïve Bayes and Logistic Regression”

Optional
• Ng & Jordan

Logistic Regression

Idea:
• Naïve Bayes allows computing $P(Y|X)$ by learning $\hat{P}(Y)$ and $\hat{P}(X|Y)$

• Why not learn $\hat{P}(Y|X)$ directly?
Consider learning \( f: X \rightarrow Y \), where
- \( X \) is a vector of real-valued features, \(<X_1, \ldots, X_n>\)
- \( Y \) is boolean
- assume all \( X_i \) are conditionally independent given \( Y \)
- model \( P(X_i | Y = y_k) \) as Gaussian \( N(\mu_{ik}, \sigma_i) \)
- model \( P(Y) \) as Bernoulli (\( \pi \))

What does that imply about the form of \( P(Y | X) \)?

\[
P(Y = 1 | X = <X_1, \ldots, X_n>) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}
\]

Derive form for \( P(Y | X) \) for continuous \( X_i \)

\[
P(Y = 1 | X) = \frac{P(Y = 1)P(X | Y = 1)}{P(Y = 1)P(X | Y = 1) + P(Y = 0)P(X | Y = 0)}
\]

\[
= \frac{1}{1 + \frac{P(Y = 0)P(X | Y = 0)\exp(\ln \frac{P(Y = 0)P(X | Y = 0)}{P(Y = 1)P(X | Y = 1)})}{1 + \exp(\ln \frac{P(Y = 0)P(X | Y = 0)}{P(Y = 1)P(X | Y = 1)})} + \sum_i \ln \frac{P(X_i | Y = 0)}{P(X_i | Y = 1)}
\]

\[
P(x | y_i) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_i^2}}
\]

\[
P(Y = 1 | X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}
\]
Very convenient!

\[ P(Y = 1|X = < X_1, \ldots, X_n >) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)} \]

implies

\[ P(Y = 0|X = < X_1, \ldots, X_n >) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)} \]

implies

\[ \frac{P(Y = 0|X)}{P(Y = 1|X)} = \exp(w_0 + \sum_i w_i X_i) \]

implies

\[ \ln \frac{P(Y = 0|X)}{P(Y = 1|X)} = w_0 + \sum_i w_i X_i \]
\[ \ln \frac{P(Y = 0|X)}{P(Y = 1|X)} = w_0 + \sum_i w_i X_i \]
Logistic regression more generally

- Logistic regression when $Y$ not boolean (but still discrete-valued).
- Now $y \in \{y_1, y_2, ..., y_R\}$: learn $R-1$ sets of weights

$$P(Y = y_k|X) = \frac{\exp(w_{k0} + \sum_{i=1}^{n} w_{ki}X_i)}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji}X_i)}$$

for $k < R$

$$P(Y = y_R|X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji}X_i)}$$

Training Logistic Regression: MCLE

- we have $L$ training examples: $\{<X_1^1, Y_1^1>, ..., <X_L^L, Y_L^L>\}$
- maximum likelihood estimate for parameters $W$

$$W_{MLE} = \arg \max_W P(<X_1^1, Y_1^1> \ldots <X_L^L, Y_L^L > | W)$$

$$= \arg \max_W \prod_i P(<X_i^i, Y_i^i> | W)$$

- maximum conditional likelihood estimate

$$\arg \max_W \prod_i P(Y_i^i | X_i^i, W)$$
Training Logistic Regression: MCLE

• Choose parameters $W = \langle w_0, \ldots w_n \rangle$ to maximize conditional likelihood of training data

  where
  \[ P(Y = 0|X, W) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)} \]
  \[ P(Y = 1|X, W) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)} \]

• Training data $D = \{ \langle X^1, Y^1 \rangle, \ldots \langle X^L, Y^L \rangle \}$
• Data likelihood $= \prod P(X^l, Y^l|W)$
• Data conditional likelihood $= \prod P(Y^l|X^l, W)$

\[ W_{MCLE} = \arg \max_W \prod_l P(Y^l|W, X^l) \]

Expressing Conditional Log Likelihood

\[ l(W) \equiv \ln \prod_l P(Y^l|X^l, W) = \sum_l \ln P(Y^l|X^l, W) \]

\[ P(Y = 0|X, W) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)} \]
\[ P(Y = 1|X, W) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)} \]

\[ l(W) = \sum_l Y^l \ln P(Y^l = 1|X^l, W) + (1 - Y^l) \ln P(Y^l = 0|X^l, W) \]
\[ = \sum_l Y^l \ln \frac{P(Y^l = 1|X^l, W)}{P(Y^l = 0|X^l, W)} + \ln P(Y^l = 0|X^l, W) \]
\[ = \sum_l Y^l (w_0 + \sum_i w_i X_i^l) - \ln (1 + \exp(w_0 + \sum_i w_i X_i^l)) \]
Maximizing Conditional Log Likelihood

\[ P(Y = 0|X, W) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)} \]

\[ P(Y = 1|X, W) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)} \]

\[
\ell(W) \equiv \ln \prod_l P(Y^l|X^l, W) \\
= \sum_l Y^l(w_0 + \sum_i w_i X_i^l) - \ln(1 + \exp(w_0 + \sum_i w_i X_i^l))
\]

Good news: \( \ell(W) \) is concave function of \( W \)

Bad news: no closed-form solution to maximize \( \ell(W) \)

Gradient Descent

\[ \nabla E[w] = \begin{bmatrix} \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_n} \end{bmatrix} \]

Training rule:

\[ \Delta w = -\eta \nabla E[w] \]

i.e.,

\[ \Delta w_i = -\eta \frac{\partial E}{\partial w_i} \]
Gradient Descent:

**Batch gradient**: use error $E_D(w)$ over entire training set $D$

Do until satisfied:

1. Compute the gradient $\nabla E_D(w) = \left[ \frac{\partial E_D(w)}{\partial w_0}, \ldots, \frac{\partial E_D(w)}{\partial w_n} \right]$
2. Update the vector of parameters: $w \leftarrow w - \eta \nabla E_D(w)$

**Stochastic gradient**: use error $E_d(w)$ over single examples $d \in D$

Do until satisfied:

1. Choose (with replacement) a random training example $d \in D$
2. Compute the gradient just for $d$: $\nabla E_d(w) = \left[ \frac{\partial E_d(w)}{\partial w_0}, \ldots, \frac{\partial E_d(w)}{\partial w_n} \right]$
3. Update the vector of parameters: $w \leftarrow w - \eta \nabla E_d(w)$

Stochastic approximates Batch arbitrarily closely as $\eta \to 0$

Stochastic can be much faster when $D$ is very large

Intermediate approach: use error over subsets of $D$

Maximize Conditional Log Likelihood:

**Gradient Ascent**

$$l(W) \equiv \ln \prod_l P(Y^l | X^l, W)$$

$$= \sum_l Y^l(w_0 + \sum_i w_i X_i^l) - \ln(1 + \exp(w_0 + \sum_i w_i X_i^l))$$

$$\frac{\partial l(W)}{\partial w_i} = \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$
Maximize Conditional Log Likelihood: Gradient Ascent

\[ l(W) = \ln \prod_l P(Y^l | X^l, W) \]
\[ = \sum_l Y^l(w_0 + \sum_i w_i X_i^l) - \ln(1 + \exp(w_0 + \sum_i w_i X_i^l)) \]

\[ \frac{\partial l(W)}{\partial w_i} = \sum_l X_i^l(Y^l - \hat{P}(Y^l = 1 | X^l, W)) \]

Gradient ascent algorithm: iterate until change < \( \varepsilon \)
For all \( i \), repeat
\[ w_i \leftarrow w_i + \eta \sum_l X_i^l(Y^l - \hat{P}(Y^l = 1 | X^l, W)) \]

That’s all for M(C)LE. How about MAP?

- One common approach is to define priors on \( W \)
  - Normal distribution, zero mean, identity covariance
- Helps avoid very large weights and overfitting
- MAP estimate

\[ W \leftarrow \arg \max_W \ln P(W) \prod_l P(Y^l | X^l, W) \]

- Let’s assume Gaussian prior: \( W \sim N(0, \sigma) \)
MLE vs MAP

- Maximum conditional likelihood estimate
  \[ W \leftarrow \arg \max_W \ln \prod_l P(Y^l|X^l, W) \]
  \[ w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W)) \]

- Maximum a posteriori estimate with prior \( W \sim N(0, \sigma I) \)
  \[ W \leftarrow \arg \max_W \ln [P(W) \prod_l P(Y^l|X^l, W)] \]
  \[ w_i \leftarrow w_i - \eta \lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W)) \]

MAP estimates and Regularization

- Maximum a posteriori estimate with prior \( W \sim N(0, \sigma I) \)
  \[ W \leftarrow \arg \max_W \ln [P(W) \prod_l P(Y^l|X^l, W)] \]
  \[ w_i \leftarrow w_i - \eta \lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W)) \]

called a “regularization” term
- helps reduce overfitting, especially when training data is sparse
- keep weights nearer to zero (if \( P(W) \) is zero mean Gaussian prior), or whatever the prior suggests
- used very frequently in Logistic Regression
The Bottom Line

• Consider learning \( f: X \rightarrow Y \), where
  • \( X \) is a vector of real-valued features, \(< X_1 \ldots X_n >\)
  • \( Y \) is boolean
  • assume all \( X_i \) are conditionally independent given \( Y \)
  • model \( P(X_i \mid Y = y_k) \) as Gaussian \( \text{N}(\mu_{ik}, \sigma_i) \)
  • model \( P(Y) \) as Bernoulli \((\pi)\)

• Then \( P(Y|X) \) is of this form, and we can directly estimate \(W\)

\[
P(Y = 1|X = < X_1, \ldots X_n >) = \frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}
\]

• Furthermore, same holds if the \( X_i \) are boolean
  • trying proving that to yourself

Generative vs. Discriminative Classifiers

Training classifiers involves estimating \( f: X \rightarrow Y \), or \( P(Y|X) \)

Generative classifiers (e.g., Naïve Bayes)
• Assume some functional form for \( P(X|Y) \), \( P(X) \)
• Estimate parameters of \( P(X|Y) \), \( P(X) \) directly from training data
• Use Bayes rule to calculate \( P(Y|X=x_i) \)

Discriminative classifiers (e.g., Logistic regression)

• Assume some functional form for \( P(Y|X) \)
• Estimate parameters of \( P(Y|X) \) directly from training data
Use Naïve Bayes or Logistic Regression?

Consider

• Restrictiveness of modeling assumptions

• Rate of convergence (in amount of training data) toward asymptotic hypothesis

Naïve Bayes vs Logistic Regression

Consider $Y$ boolean, $X_i$ continuous, $X=<X_1 ... X_n>$

Number of parameters to estimate:

• NB:

  \[
  P(Y) \
  P(x_i|Y=1) \approx \frac{2N_i x_i}{N} \quad \text{and} \quad N \quad \text{and} \quad 2
  \]

  \[
  \sigma_i \approx \sigma_i
  \]

• LR:

  \[
  \sum \quad \text{and} \quad 1
  \]

\[
P(Y=0|X,W) = \frac{1}{1 + \exp(w_0 + \sum_i w_iX_i)}
\]

\[
P(Y=1|X,W) = \frac{\exp(w_0 + \sum_i w_iX_i)}{1 + \exp(w_0 + \sum_i w_iX_i)}
\]
Naïve Bayes vs Logistic Regression

Consider Y boolean, Xᵢ continuous, X=<X₁ ... Xₙ>

Number of parameters:
- NB: 4n +1
- LR: n+1

Estimation method:
- NB parameter estimates are uncoupled
- LR parameter estimates are coupled

Recall two assumptions deriving from LR from GNBayes:
1. Xᵢ conditionally independent of Xₖ given Y ✔
2. P(Xᵢ | Y = y_k) = N(µₖᵢk,σᵢk) ↯ not N(µₖᵢk,σₖᵢk)

Consider three learning methods:
- GNB (assumption 1 only)
- GNB2 (assumption 1 and 2)
- LR

Which method works better if we have *infinite* training data, and…
- Both (1) and (2) are satisfied
- Neither (1) nor (2) is satisfied
- (1) is satisfied, but not (2)
G. Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

Recall two assumptions deriving form of LR from GNBayes:
1. \( X_i \) conditionally independent of \( X_k \) given \( Y \)
2. \( P(X_i \mid Y = y_k) = \mathcal{N}(\mu_{ik}, \sigma_{ik}) \), \( \not\sim \mathcal{N}(\mu_{ik}, \sigma_{ik}) \)

Consider three learning methods:
• GNB (assumption 1 only)
• GNB2 (assumption 1 and 2)
• LR

Which method works better if we have *infinite* training data, and...

• Both (1) and (2) are satisfied
• Neither (1) nor (2) is satisfied
• (1) is satisfied, but not (2)
G.Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

What if we have only finite training data?

They converge at different rates to their asymptotic (= data) error

Let \( \epsilon_{A,n} \) refer to expected error of learning algorithm A after \( n \) training examples

Let \( d \) be the number of features: \( \langle X_1 \ldots X_d \rangle \)

\[
\epsilon_{LR,n} \leq \epsilon_{LR,\infty} + O\left(\frac{\sqrt{d}}{n}\right)
\]

\[
\epsilon_{GNB,n} \leq \epsilon_{GNB,\infty} + O\left(\frac{\log d}{n}\right)
\]

So, GNB requires \( n = O(\log d) \) to converge, but LR requires \( n = O(d) \)

Some experiments from UCI data sets

[Ng & Jordan, 2002]
Naïve Bayes vs. Logistic Regression

The bottom line:

GNB2 and LR both use linear decision surfaces, GNB need not

Given infinite data, LR is better or equal to GNB2 because \textit{training procedure} does not make assumptions 1 or 2 (though our derivation of the form of $P(Y|X)$ did).

But GNB2 converges more quickly to its perhaps-less-accurate asymptotic error

And GNB is both more biased (assumption1) and less (no assumption 2) than LR, so either might beat the other

What you should know:

• Logistic regression
  – Functional form follows from Naïve Bayes assumptions
    • For Gaussian Naïve Bayes assuming variance $\sigma_{i,k} = \sigma_i$
    • For discrete-valued Naïve Bayes too
  – But training procedure picks parameters without making conditional independence assumption
  – MLE training: pick $W$ to maximize $P(Y \mid X, W)$
  – MAP training: pick $W$ to maximize $P(W \mid X,Y)$
    • ‘regularization’
    • helps reduce overfitting

• Gradient ascent/descent
  – General approach when closed-form solutions unavailable

• Generative vs. Discriminative classifiers
  – Bias vs. variance tradeoff