Bayesian Learning

Gaussian Naive Bayes Classifier & Logistic Regression

Gaussian Naive Bayes Classifier

Bayes Theorem - Example

Sample Space for events A and B

A holds	T	T	F	F	T	F	T
B holds	T	F	T	F	T	F	F

P(A) = 4/7 P(B) = 3/ 7 P(B|A) = 2/4 P(A|B) = 2/3

Is Bayes Theorem correct?

P(B|A) = P(A|B) P(B) / P(A) = (2/3 * 3/7) / 4/7 = 2/4 ➔ CORRECT

P(A|B) = P(B|A) P(A) / P(B) = (2/4 * 4/7) / 3/7 = 2/3 ➔ CORRECT

Naive Bayes Classifier

- Practical Bayesian learning method is Naive Bayes Learner (**Naive Bayes Classifier).**
- The naive Bayes classifier applies to learning tasks where each instance ***x*** is described by a conjunction of attribute values and where the target function f(x) can take on any value from some finite set V.
- A set of training examples is provided, and a new instance is presented, described by the tuple of attribute values $\mathbf{(a_l, a_2 ... a_n)}$**.**
- The learner is asked to predict the target value (**classification**), for this new instance.

Naive Bayes Classifier

- The Bayesian approach to classifying the new instance is to assign the **most probable target value v_{MAP}**, given the attribute values (a_l, a_2 ... a_n) that describe the instance.

$$\mathbf{v_{MAP} = \underset{v_j \in V}{argmax}\, P(v_j | a_1, \dots, a_n)}$$

- By Bayes theorem:

$$\mathbf{v_{MAP} = \underset{v_j \in V}{argmax}\, \frac{P(a_1,\dots,a_n | v_j)\, P(v_j)}{P(a_1,\dots,a_n)}}$$

$$\mathbf{v_{MAP} = \underset{v_j \in V}{argmax}\, P(a_1, \dots, a_n | v_j)\, P(v_j)}$$

Naive Bayes Classifier

- It is easy to estimate each of the $\mathbf{P(v_j)}$ simply by counting the frequency with which each target value $\mathbf{v_j}$ occurs in the training data.
- However, estimating the different $\mathbf{P(a_l,a_2 \ldots a_n \mid v_j)}$ terms is not feasible unless we have a very, very large set of training data.
  - The problem is that the number of these terms is equal to the number of possible instances times the number of possible target values.
  - Therefore, we need to see every instance in the instance space many times in order to obtain reliable estimates.
- The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value.
- For a given the target value of the instance, the probability of observing conjunction $\mathbf{a_l,a_2 \ldots a_n}$ **,** is just the product of the probabilities for the individual attributes:

$$P(a_1, \ldots, a_n | v_j) = \prod_i P(a_i | v_j)$$

- **Naive Bayes classifier:** $\mathbf{v_{NB} = \underset{v_j \in V}{argmax}\ P(v_j) \ \prod_i P(a_i | v_j)}$

Naive Bayes in a Nutshell

Bayes rule:

$$P(Y = y_k | X_1 \ldots X_n) = \frac{P(Y = y_k)\, P(X_1 \ldots X_n | Y = y_k)}{\sum_j P(Y = y_j)\, P(X_1 \ldots X_n | Y = y_j)}$$

Assuming conditional independence among X_i's:

$$\mathbf{P(Y = y_k | X_1 \ldots X_n) = \frac{P(Y = y_k)\ \prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i | Y = y_j)}}$$

So, classification rule for $X^{new} = < X_1^{new} \ldots X_n^{new} >$ is:

$$\mathbf{Y^{new} \leftarrow \underset{y_k}{argmax}\ \ P(Y = y_k) \prod_i P(X_i^{new} | Y = y_k)}$$

Naive Bayes in a Nutshell

- Another way to view Naïve Bayes when Y is a Boolean attribute:

- Decision rule: is this quantity greater or less than 1?

$$\frac{P(Y = 1|X_1 \ldots X_n)}{P(Y = 0|X_1 \ldots X_n)} = \frac{P(Y = 1) \prod_i P(X_i|Y = 1)}{P(Y = 0) \prod_i P(X_i|Y = 0)}$$

Naive Bayes in a Nutshell

- **What if we have continuous X_i ?**

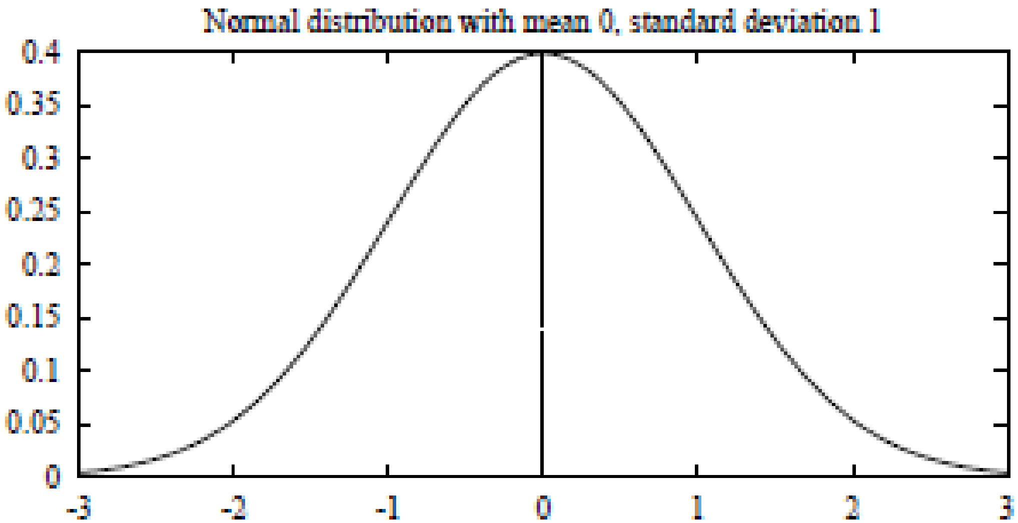
- Still we have:

$$\mathbf{P(Y = y_k | X_1 \dots X_n) = \frac{P(Y = y_k) \prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i | Y = y_j)}}$$

- Just need to decide how to represent $P(X_i \mid Y)$

- ***Common approach***: assume $P(X_i|Y=y_k)$ follows a ***Normal (Gaussian) distribution***

Gaussian Distribution
(also called "Normal Distribution")



A **Normal Distribution (Gaussian Distribution)** is a bell-shaped distribution defined by ***probability density function***

$$\mathbf{p(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}}$$

- A Normal distribution is fully determined by two parameters in the formula: μ and σ .
- If the random variable X follows a normal distribution:

  - The probability that X will fall into the interval (a, b) is $\int_a^b p(x)\, d(x)$

  - The expected, or ***mean value of X***, $E[X] = \mu$

  - The ***variance of X***, $Var(X) = \sigma^2$

  - The ***standard deviation of X***, $\sigma_x = \sigma$

Gaussian Naive Bayes (GNB)

Gaussian Naive Bayes (GNB) assumes

$$\mathbf{P(X_i = x|Y = y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}}\ e^{-\frac{1}{2}\left(\frac{x-\mu_{ik}}{\sigma_{ik}}\right)^2}}$$

where

- μ_{ik} is the **mean** of X_i values of the instances whose Y attribute is y_k.
- σ_{ik} is the **standard deviation** of X_i values of the instances whose Y attribute is y_k.

- Sometimes σ_{ik} and μ_{ik} are assumed as σ_i and μ_i
  - independent of Y

Gaussian Naive Bayes Algorithm

continuous X_i (but still discrete Y)

- **Train Gaussian Naive Bayes Classifier:** For each value y_k
  - Estimate $\mathbf{P(Y = y_k)}$:

    $\mathbf{P(Y = y_k)}$ **= (# of y_k examples) / (total # of examples)**

  - For each attribute X_i, in order to estimate $\mathbf{P(X_i | Y = y_k)}$:
    - Estimate class **conditional mean μ_{ik}** and **standard deviation σ_{ik}**

- **Classify (X^{new}):**

$$\mathbf{Y^{new} \leftarrow \underset{y_k}{argmax}\ P(Y = y_k) \prod_i P(X_i^{new} | Y = y_k)}$$

$$\mathbf{Y^{new} \leftarrow \underset{y_k}{argmax}\ P(Y = y_k) \prod_i \frac{1}{\sqrt{2\pi\sigma_{ik}^2}}\ e^{-\frac{1}{2}\left(\frac{X_i^{new} - \mu_{ik}}{\sigma_{ik}}\right)^2}}$$

Estimating Parameters:

- When the X_i are continuous we must choose some other way to represent the distributions $P(X_i|Y)$.

- We assume that for each possible discrete value y_k of Y, the distribution of each continuous X_i is Gaussian, and is defined by a mean and standard deviation specific to X_i and y_k.

- In order to train such a Naive Bayes classifier we must therefore estimate the **mean** and **variance** of each of these Gaussians:

$$\mu_{ik} = E[X_i|Y = y_k]$$

$$\sigma_{ik}^2 = E[(X_i - \mu_{ik})^2|Y = y_k]$$

Estimating Parameters:
Y discrete, X_i continuous

For each **k[th] class** ($Y=y_k$) and **i[th] feature** (X_i):

- $\boldsymbol{\mu_{ik}}$ is the **mean** of X_i values of the examples whose Y attribute is y_k.

$$\mu_{ik} = \frac{\textbf{summation of all } X_i \textbf{ values of } y_k \textbf{ examples}}{\textbf{\# of } y_k \textbf{ examples}}$$

- $\boldsymbol{\sigma^2_{ik}}$ is the **variance** of X_i values of the examples whose Y attribute is y_k.

$$\sigma^2_{ik} = \frac{\textbf{summation of all } (X_i^j - \mu_{ik})^2 \textbf{ for } X^j \textbf{ where } X^j \textbf{ is a } y_k \textbf{ example}}{\textbf{\# of } y_k \textbf{ examples}}$$

Estimating Parameters:

- How many parameters must we estimate for Gaussian Naive Bayes if Y has k possible values, and examples have n attributes?

➔ **2*n*k parameters** (***n*k mean values***, and ***n*k variances***)

$$\mathbf{P(X_i = x | Y = y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}}\ e^{-\frac{1}{2}\left(\frac{x-\mu_{ik}}{\sigma_{ik}}\right)^2}}$$

Logistic Regression

Logistic Regression

Idea:

- Naive Bayes allows computing P(Y|X) by learning P(Y) and P(X|Y)

- Why not learn P(Y|X) directly?

➔ **logistic regression**

Logistic Regression

- Consider learning f: X →Y, where
  - X is a vector of real-valued features, $< X_1 \ldots X_n >$
  - Y is boolean (**binary logistic regression**)
    - In general, Y is a discrete attribute and it can take $k \geq 2$ different values.
  - assume all X_i are conditionally independent given Y.
  - model $P(X_i \mid Y=y_k)$ as Gaussian

$$\mathbf{P(X_i|Y = y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}}\; e^{-\frac{1}{2}\left(\frac{X_i-\mu_{ik}}{\sigma_{ik}}\right)^2}}$$

  - model P(Y) as Bernoulli (π): summation of all possible probabilities is 1.

$$\mathbf{P(Y=1) = \pi \qquad P(Y=0) = 1-\pi}$$

- What do these assumptions imply about the form of P(Y|X)?

Logistic Regression

Derive P(Y|X) for Gaussian $P(X_i|Y=y_k)$ assuming $\sigma_{ik} = \sigma_i$

- In addition, assume variance is independent of class, i.e. $\boldsymbol{\sigma_{i0} = \sigma_{i1} = \sigma_i}$

$$P(Y=1|X) = \frac{P(Y=1)\,P(X|Y=1)}{P(Y=1)\,P(X|Y=1) + P(Y=0)\,P(X|Y=0)}$$

by Bayes theorem

$$P(Y=1|X) = \frac{1}{1 + \frac{P(Y=0)\,P(X|Y=0)}{P(Y=1)\,P(X|Y=1)}}$$

divide numerator and denumerator with same term

$$P(Y=1|X) = \frac{1}{1 + \exp\left(\ln \frac{P(Y=0)\,P(X|Y=0)}{P(Y=1)\,P(X|Y=1)}\right)}$$

$\exp(x)=e^x$ and $x = e^{\ln(x)}$

$$P(Y=1|X) = \frac{1}{1 + \exp\left(\ln \frac{P(Y=0)}{P(Y=1)} + \ln \frac{P(X|Y=0)}{P(X|Y=1)}\right)}$$

math fact

Logistic Regression

Derive P(Y|X) for Gaussian $P(X_i|Y=y_k)$ assuming $\sigma_{ik} = \sigma_i$

- $P(Y=1)=\pi$ and $P(Y=0)=1-\pi$ by modelling P(Y) as Bernoulli
- By independence assumption $\frac{P(X|Y=0)}{P(X|Y=1)} = \prod_i \frac{P(X_i|Y=0)}{P(X_i|Y=1)}$

$$P(Y=1|X) = \frac{1}{1 + \exp\left(\ln \frac{P(Y=0)}{P(Y=1)} + \ln \frac{P(X|Y=0)}{P(X|Y=1)}\right)}$$

$$P(Y=1|X) = \frac{1}{1 + \exp\left(\ln \frac{1-\pi}{\pi} + \ln \prod_i \frac{P(X_i|Y=0)}{P(X_i|Y=1)}\right)}$$

by assumptions above

$$P(Y=1|X) = \frac{1}{1 + \exp\left(\ln \frac{1-\pi}{\pi} + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)}\right)}$$

math fact

Logistic Regression

Derive P(Y|X) for Gaussian $P(X_i|Y=y_k)$ assuming $\sigma_{ik} = \sigma_i$

Since $\mathbf{P(X_i|Y = y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}}\, e^{-\frac{1}{2}\left(\frac{X_i-\mu_{ik}}{\sigma_{ik}}\right)^2}}$ and $\boldsymbol{\sigma_{i0} = \sigma_{i1} = \sigma_i}$

$$\ln\frac{1-\pi}{\pi} + \sum_i \ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)} = \ln\frac{1-\pi}{\pi} + \sum_i \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} + \sum_i \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i$$

$$P(Y=1|X) = \frac{1}{1 + \exp\left(\ln\frac{1-\pi}{\pi} + \sum_i \ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)}\right)}$$

$$\mathbf{P(Y=1|X) = \frac{1}{1 + \exp\left(w_0 + \sum_i w_i X_i\right)}}$$

Logistic Regression

Derive P(Y|X) for Gaussian $P(X_i|Y=y_k)$ assuming $\sigma_{ik} = \sigma_i$

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

implies

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

implies

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

implies

$$\ln \frac{P(Y=0|X)}{P(Y=1|X)} = w_0 + \sum_i w_i X_i$$

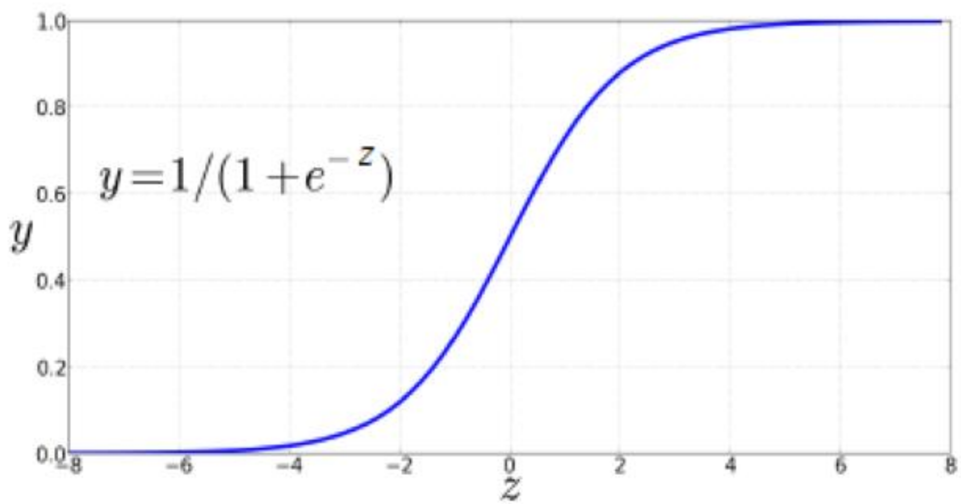
a linear classification rule

Logistic Regression
Logistic (sigmoid) function

- In Logistic Regression, P(Y|X) is assumed to be the following functional form which is a **sigmoid (logistic) function**.

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

- The **sigmoid function** $\mathbf{y} = \frac{\mathbf{1}}{\mathbf{1+exp(-z)}}$ takes a real value and maps it to the range [0,1].

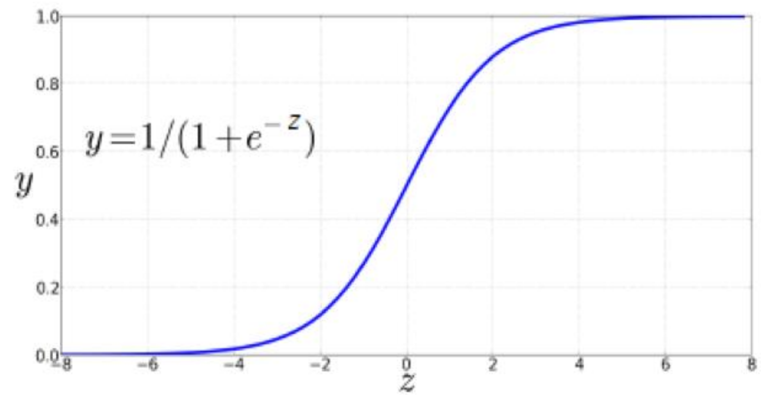


Logistic Regression is a Linear Classifier

- In Logistic Regression, P(Y|X) is assumed:

$$P(Y = 1|X) = \frac{1}{1 + \exp\left(w_0 + \sum_i w_i X_i\right)}$$

$$P(Y = 0|X) = \frac{\exp\left(w_0 + \sum_i w_i X_i\right)}{1 + \exp\left(w_0 + \sum_i w_i X_i\right)}$$



- **Decision Boundary:**

P(Y=0|X) > P(Y=1|X) ➔ Classification is 0 ➔ $\mathbf{w_0 + \sum_i w_i X_i > 0}$

P(Y=0|X) < P(Y=1|X) ➔ Classification is 1 ➔ $\mathbf{w_0 + \sum_i w_i X_i < 0}$

– $\mathbf{w_0 + \sum_i w_i X_i}$ **is a linear decision boundary.**

Logistic Regression for More Than 2 Classes

- In general case of logistic regression, we have M classes and $Y \in \{y_1, \ldots, y_M\}$

- for k < M

$$\mathbf{P(Y = y_k | X) = \frac{exp\,(\, w_{k0} + \sum_i w_{ki}\, X_i\,)}{1 \,+\, \sum_{j=1}^{M-1}\, exp\,(\, w_{j0} + \sum_i w_{ji}\, X_i\,)}}$$

- for k = M   (no weights for the last class)

$$\mathbf{P(Y = y_M | X) = \frac{1}{1 \,+\, \sum_{j=1}^{M-1}\, exp\,(\, w_{j0} + \sum_i w_{ji}\, X_i\,)}}$$

Training Logistic Regression

- We will focus on **binary logistic regression**.
- **Training Data:** We have **n training examples**: { <X^1,Y^1>, …, < X^n,Y^n > }
- **Attributes:** We have **d attributes:** We have to learn **weights W: w_0,w_1,…,w_d**
- We want to learn weights which produces maximum probability for the training data.

Maximum Likelihood Estimate for parameters W:

$$\mathbf{W_{MLE}} = \underset{\mathbf{W}}{\mathbf{argmax}}\ \mathbf{P}\left(\{\langle \mathbf{X^1}, \mathbf{Y^1} \rangle, \ldots, \langle \mathbf{X^n}, \mathbf{Y^n} \rangle \mid \mathbf{W} \right)$$

$$\mathbf{W_{MLE}} = \underset{\mathbf{W}}{\mathbf{argmax}} \prod_{j=1}^{n} \mathbf{P}\left(\langle \mathbf{X^j}, \mathbf{Y^j} \rangle \mid \mathbf{W} \right)$$

Training Logistic Regression

Maximum Conditional Likelihood Estimate

$$\mathbf{W}_{\mathbf{MLE}} = \underset{\mathbf{W}}{\mathbf{argmax}} \prod_{j=1}^{n} \mathbf{P}\left(\langle \mathbf{X}^{j}, \mathbf{Y}^{j} \rangle \mid \mathbf{W} \right)$$

data likelihood

- **Maximum Conditional Likelihood Estimate for parameters W:**

$$\mathbf{W}_{\mathbf{MCLE}} = \underset{\mathbf{W}}{\mathbf{argmax}} \prod_{j=1}^{n} \mathbf{P}\left(\mathbf{Y}^{j} \mid \mathbf{X}^{j}, \mathbf{W} \right)$$

conditional data likelihood

where

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

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

Training Logistic Regression

Expressing Conditional Log Likelihood

- Log value of conditional data likelihood l(W)

$$\mathbf{l(W) = \ln \prod_{j=1}^{n} P\left(Y^j \,\middle|\, X^j, W\right) = \sum_{j=1}^{n} \ln P\left(Y^j \,\middle|\, X^j, W\right)}$$

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

$$\mathbf{l(W) = \sum_{j=1}^{n} Y^j \ln P\left(Y^j = 1 \middle| X^j, W\right) + (1 - Y^j) \ln P\left(Y^j = 0 \middle| X^j, W\right)}$$

- Y can take only values 0 or 1, so only one of the two terms in the expression will be non-zero for any given Y^j

$$\mathbf{l(W) = \sum_{j=1}^{n} Y^j \ln \frac{P\left(Y^j=1 \middle| X^j, W\right)}{P\left(Y^j=0 \middle| X^j, W\right)} + \ln P\left(Y^j = 0 \middle| X^j, W\right)}$$

$$\mathbf{l(W) = \sum_{j=1}^{n} Y^j \left(w_0 + \sum_i w_i X_i^j\right) - \ln(1 + \exp(w_0 + \sum_i w_i X_i^j))}$$

Training Logistic Regression

Maximizing Conditional Log Likelihood

$$\mathbf{l(W) = \sum_{j=1}^{n} Y^j \left(w_0 + \sum_i w_i X_i^j\right) - ln(1 + exp(w_0 + \sum_i w_i X_i^j))}$$

Bad News:

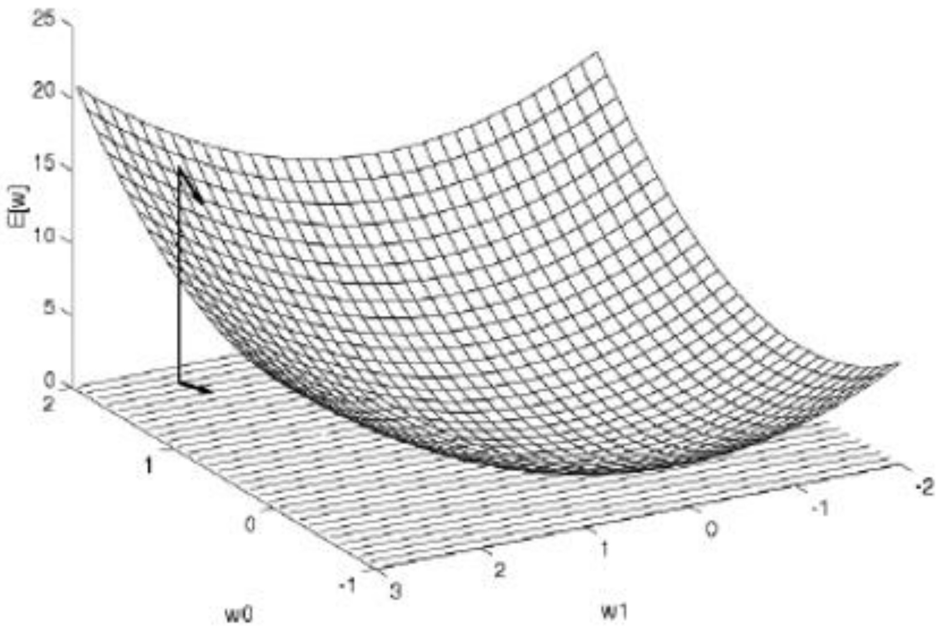
- Unfortunately, there is no closed form solution to maximizing l(W) with respect to W.

Good News:

- l(W) is **concave** function of W. **Concave** functions are easy to optimize.
- Therefore, one common approach is to use **gradient ascent**
- maximum of a concave function = minimum of a convex function

  Gradient Ascent (concave) / Gradient Descent (convex)

Gradient Descent



Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Gradient Descent

Batch gradient: use error $E_D(\mathbf{w})$ over entire training set D

Do until satisfied:

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

gradient descent

Stochastic gradient: use error $E_d(\mathbf{w})$ 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: $\nabla E_d(\mathbf{w}) = \left[\frac{\partial E_d(\mathbf{w})}{\partial w_0} \ldots \frac{\partial E_d(\mathbf{w})}{\partial w_n}\right]$
3. Update the vector of parameters: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla E_d(\mathbf{w})$

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

Stochastic can be much faster when D is very large

Intermediate approach: use error over subsets of D

Maximizing Conditional Log Likelihood
Gradient Ascent

$$l(W) = \sum_{j=1}^{n} Y^j \left(w_0 + \sum_i w_i X_i^j\right) - \ln(1 + \exp(w_0 + \sum_i w_i X_i^j))$$

Gradient $= \left[\frac{\partial l(W)}{\partial w_0}, \dots, \frac{\partial l(W)}{\partial w_n}\right]$

Update Rule:

$$w_i = w_i + \eta \frac{\partial l(W)}{\partial w_i}$$

gradient ascent

learning rate

Maximizing Conditional Log Likelihood
Gradient Ascent

$$l(W) = \sum_{j=1}^{n} Y^j \left(w_0 + \sum_i w_i X_i^j\right) - \ln(1 + \exp(w_0 + \sum_i w_i X_i^j))$$

$$\frac{\partial l(W)}{\partial w_i} = \sum_{j=1}^{n} Y^j X_i^j - X_i^j \frac{\exp(w_0 + \sum_i w_i X_i^j)}{1 + \exp(w_0 + \sum_i w_i X_i^j)}$$

$$\frac{\partial l(W)}{\partial w_i} = \sum_{j=1}^{n} X_i^j (Y^j - P(Y^j = 1 | X^j, W))$$

Gradient ascent algorithm: iterate until change < ε
For all i, repeat

$$w_i = w_i + \eta \sum_{j=1}^{n} X_i^j (Y^j - P(Y^j = 1 | X^j, W))$$

G.Naïve Bayes vs. Logistic Regression

Recall two assumptions deriving form of LR from GNBayes:

1. X_i conditionally independent of X_k given Y
2. $P(X_i \mid Y = y_k) = N(\mu_{ik}, \sigma_i)$, ← not $N(\mu_{ik}, \sigma_{ik})$

Consider three learning methods:

- GNB (assumption 1 only) -- decision surface can be non-linear
- GNB2 (assumption 1 and 2) – decision surface linear
- LR -- decision surface linear, trained without assumption 1.

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

- Both (1) and (2) are satisfied: LR = GNB2 = GNB
- (1) is satisfied, but not (2) : GNB > GNB2, GNB > LR, LR > GNB2
- Neither (1) nor (2) is satisfied: GNB>GNB2, LR > GNB2, LR><GNB

G.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 than GNB2 because 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