CS6375: Machine Learning

Gautam Kunapuli

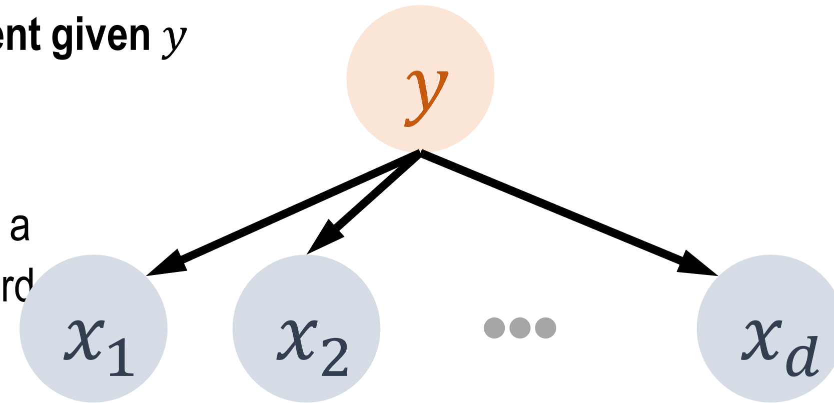
Logistic Regression

THE UNIVERSITY OF TEXAS AT DALLAS
Erik Jonsson School of Engineering and Computer Science

Naïve Bayes Classifier for Continuous Features

Naïve Bayes assumption: **features are conditionally independent given y**

$$P(x_1, x_2, \dots, x_d | y) = \prod_{j=1}^{d} P(x_j | y)$$



For a **continuous feature** x_j, we can represent $P(x_j | y = c)$ with a Gaussian distribution, with parameters μ_{jc} (mean) and σ_{jc} (standard deviation)

$$P(x_j | y = c) = \frac{1}{\sqrt{2\pi\sigma_{jc}^2}} \exp\left(-\frac{(x_j - \mu_{jc})^2}{2\sigma_{jc}^2}\right)$$

That is, if there are $c = 1, \dots, C$ classes, we model each conditional distribution with its **own** Gaussian distribution.

We can use **Maximum Likelihood Estimation** as before to estimate the parameters of the distribution:

$$\mu_{jc} = \frac{1}{\sum_{i=1}^{n} I(y_i = c)} \sum_{i=1}^{n} x_{ij} \cdot I(y_i = c)$$

$$\sigma_{jc} = \frac{1}{\sum_{i=1}^{n} I(y_i = c)} \sum_{i=1}^{n} (x_{ij} - \mu_{jc})^2 \cdot I(y_i = c)$$

where i indexes training examples, j indexes features and c indexes class labels.

$I(y_i = c)$ is an indicator function that simply indicates if its argument is true or not, that is:

$$I(y_i = c) = \begin{cases} 1, & \text{if } y_i = c \\ 0, & \text{otherwise} \end{cases}$$

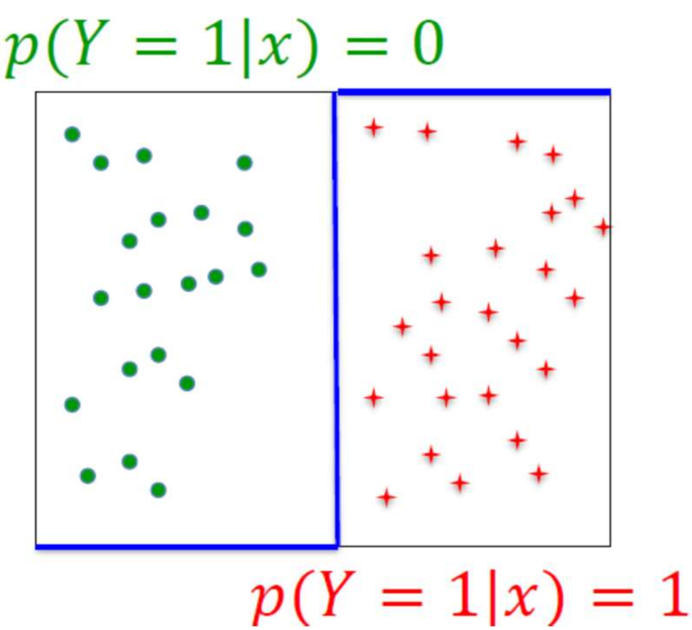
Logistic Regression

Misclassification Minimization:

Learn $p(y|\boldsymbol{x})$ directly from the data

- Assume a functional form, (e.g., a linear classifier $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$) such that
    - $p(y = 1|\ \boldsymbol{x}) = 1$ on one side and
    - $p(y = 1|\ \boldsymbol{x}) = 0$ on the other side

    that is $p(y = -1|\ \boldsymbol{x}) = 1 - p(y = 1|\ \boldsymbol{x}) = 1$
- Not differentiable
- Makes it difficult to learn
- Can't handle noisy labels

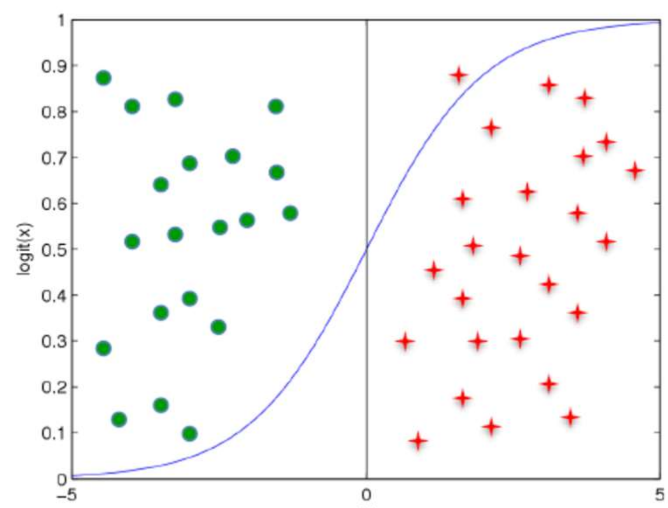


Logistic Loss Function:

Learn or $p(y|\boldsymbol{x})$ directly from the data

- Assume a functional form, (e.g., a linear classifier $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$) such that
- $p(y = -1|\ \boldsymbol{x}) = \frac{1}{1+\exp\,(\boldsymbol{w}^T\boldsymbol{x}+b)}$ on one side and
- $p(y = 1|\ \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}^T\boldsymbol{x}+b)}{1+\exp\,(\boldsymbol{w}^T\boldsymbol{x}+b)}$ on the other side

  that is $p(y = -1|\ \boldsymbol{x}) = 1 - p(y = 1|\ \boldsymbol{x})$
- Differentiable
- Easy to learn
- Handles noisy labels naturally

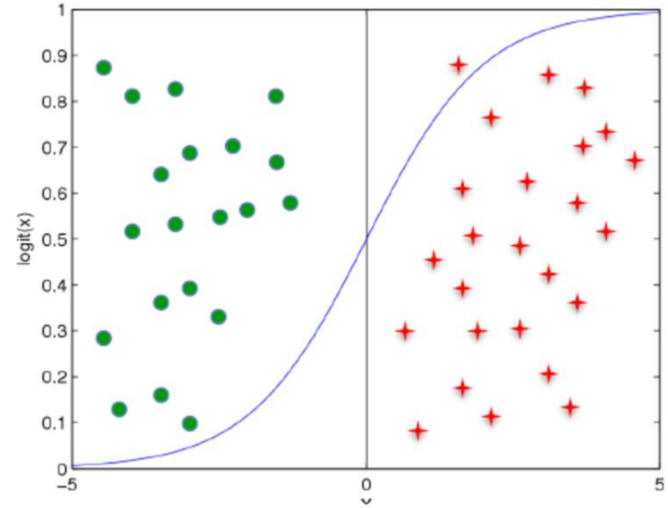


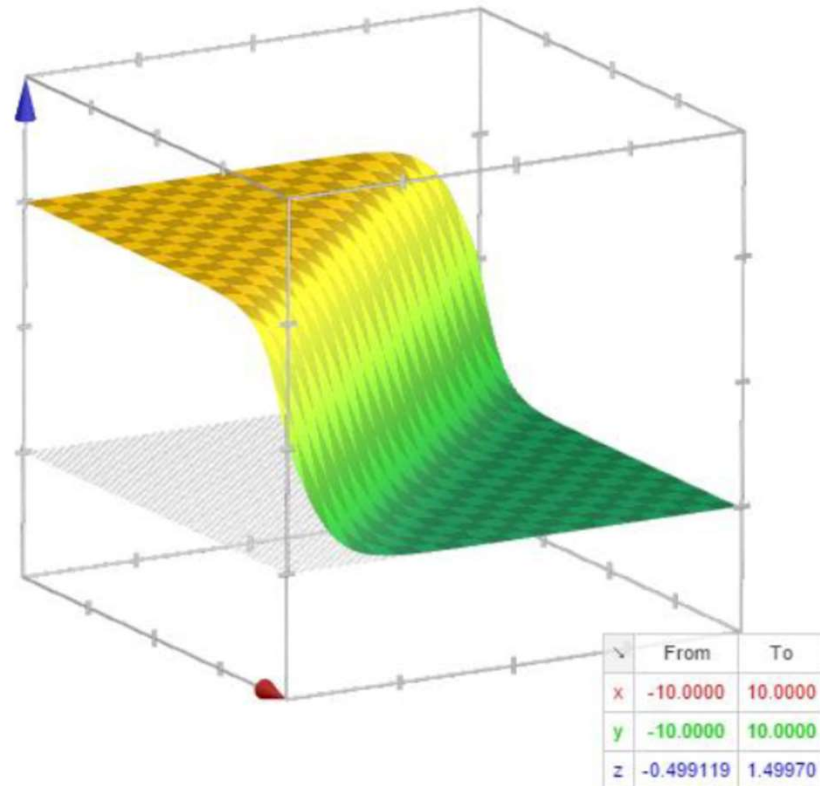
The Logistic Function

A linear function $y = f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ has a range from $[-\infty, \infty]$. The **logistic function** transforms this range to a probability $[0, 1]$.

Given some $\boldsymbol{w}$ and b, we can classify a new point $\boldsymbol{x}$ by **assigning the labels** as follows:

- $y = 1$, if $p(y = 1|\,\boldsymbol{x}) > p(y = -1|\boldsymbol{x})$ and
- $y = -1$, otherwise



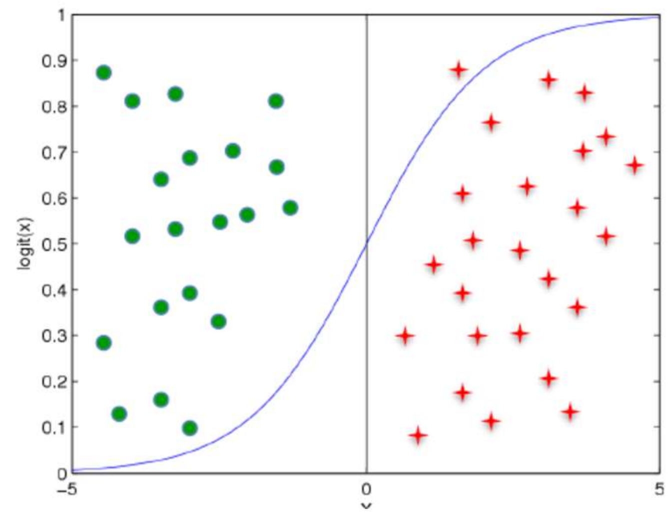


The Logistic Function

A linear function $y = f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ has a range from $[-\infty, \infty]$. The **logistic function** transforms this range to a probability $[0, 1]$.

Given some $\boldsymbol{w}$ and b, we can classify a new point $\boldsymbol{x}$ by assigning the labels as follows:

- $y = 1$, if $p(y = 1|\ \boldsymbol{x}) > p(y = -1|\boldsymbol{x})$ and
- $y = -1$, otherwise

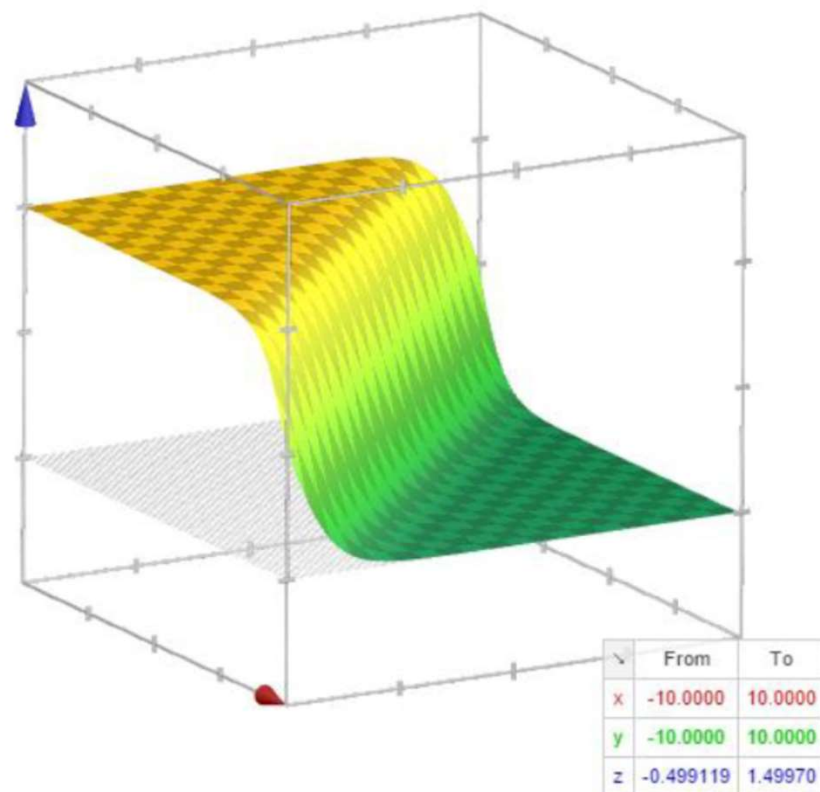


Logistic regression **implements a linear classifier**, that is:

- $y = 1$, if $\boldsymbol{w}^T\boldsymbol{x} + b > 0$ and
- $y = -1$, otherwise

We can show this by showing the **log-odds of a training example** belonging to class $y = 1$ are:

$$\log \frac{p(y = 1|\ \boldsymbol{x})}{p(y = -1|\ \boldsymbol{x})} = \boldsymbol{w}^T\boldsymbol{x} + b$$



Formulating Logistic Regression

Since we are fitting a **conditional probability distribution**, we no longer minimize the loss on the training data. Instead, we are interested in **finding the distribution** h that is **most likely** given the training data.

Let $S = (\boldsymbol{x}_i, y_i)_{i=1}^n$ be the training data set (sample). Our goal is to find h to maximize $P(h|S)$:

The distribution $P(S|h)$ is called the **likelihood function**. The **log likelihood** is frequently used as the objective function for learning.
The h that maximizes the likelihood on the training data is called the **maximum likelihood estimator**.

$$\arg\max_h P(h \mid S) = \arg\max_h \frac{P(S \mid h)P(h)}{P(S)}$$

*by **Bayes' Rule***

$$= \arg\max_h P(S \mid h)P(h)$$

because $P(S)$ doesn't depend on h

$$= \arg\max_h P(S \mid h)$$

*assuming $P(h)$ is **uniform***

$$= \arg\max_h \log P(S \mid h)$$

*because log is **monotonic***

In our framework, we **assume** that each training example is **identically and independently distributed (i.i.d.)**

$$\log P(S|h) = \log \prod_{i=1}^{n} P(\boldsymbol{x}_i, y_i|h) = \sum_{i=1}^{n} \log P(\boldsymbol{x}_i, y_i|h)$$

this shows that the log likelihood of a data set is the sum of the log likelihoods of the individual training examples in the data set

Learning the Weights

*to make everything simpler, **assume** that the labels are $y = 1$ (for positive examples) and $y = 0$ (for negative examples, instead of $y = -1$)*

$$\max_{\boldsymbol{w},b} \log P(S|\boldsymbol{w}, b)$$

$$= \max_{\boldsymbol{w},b} \log \prod_{i=1}^{n} P(\boldsymbol{x}_i, y_i|\boldsymbol{w}, b)$$

$$= \max_{\boldsymbol{w},b} \sum_{i=1}^{n} \log P(\boldsymbol{x}_i, y_i|\boldsymbol{w}, b)$$

$$= \max_{\boldsymbol{w},b} \sum_{i=1}^{n} \log \left[P(y_i|\boldsymbol{x}_i, \boldsymbol{w}, b)\, P(\boldsymbol{x}_i|\boldsymbol{w}, b)\right]$$

*by **Bayes' Rule***

$$= \max_{\boldsymbol{w},b} \sum_{i=1}^{n} \log P(y_i|\boldsymbol{x}_i, \boldsymbol{w}, b)$$

because $P(\boldsymbol{x}_i|\boldsymbol{w}, b)$ doesn't depend on $\boldsymbol{w}$ and b

$$= \max_{\boldsymbol{w},b} \sum_{i=1}^{n} y_i \log P(y_i = 1|\boldsymbol{x}_i, \boldsymbol{w}, b) + (1 - y_i) \log P(y_i = 0|\boldsymbol{x}_i, \boldsymbol{w}, b)$$

if $y_i = 1$ the log likelihood is $\log p(y = 1|\, \boldsymbol{x})$
if $y_i = 0$ the log likelihood is $\log p(y = 0|\, \boldsymbol{x})$

$$= \max_{\boldsymbol{w},b} \sum_{i=1}^{n} y_i \log \frac{P(y_i = 1|\boldsymbol{x}_i, \boldsymbol{w}, b)}{P(y_i = 0|\boldsymbol{x}_i, \boldsymbol{w}, b)} + \log P(y_i = 0|\boldsymbol{x}_i, \boldsymbol{w}, b)$$

Learning the Weights

*to make everything simpler, **assume** that the labels are $y = 1$ (for positive examples) and $y = 0$ (for negative examples, instead of $y = -1$)*

$$\max_{\boldsymbol{w},b} \sum_{i=1}^{n} y_i \log \frac{P(y_i = 1|\boldsymbol{x}_i, \boldsymbol{w}, b)}{P(y_i = 0|\boldsymbol{x}_i, \boldsymbol{w}, b)} + \log P(y_i = 0|\boldsymbol{x}_i, \boldsymbol{w}, b)$$
$$= \max_{\boldsymbol{w},b} \sum_{i=1}^{n} y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - \log(1 + \exp(\boldsymbol{w}^T\boldsymbol{x}_i + b))$$
$$= \max_{\boldsymbol{w},b} L(\boldsymbol{w}, b)$$
find parameters $\boldsymbol{w}, b$ to maximize the conditional log-likelihood

no closed-form solution!

$$\nabla_{\boldsymbol{w}} L = \sum_{i=1}^{n} (y_i - p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{w})) \cdot \boldsymbol{x}_i$$
$$\nabla_b L = \sum_{i=1}^{n} (y_i - p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{w}))$$

*gradient depends $y_i - p(y_i = 1|\boldsymbol{x}_i)$, the difference between the **true label** and the **predicted probability***
- *if $y_i = 1$ (positive example), the gradient pushes $p(y_i = 1|\boldsymbol{x}_i)$ closer to 1 (hopefully resulting in a high probability of $y_i = 1$)*
- *if $y_i = 0$ (negative example), the gradient pushes $p(y_i = 1|\boldsymbol{x}_i)$ closer to 0 (hopefully resulting in a low probability of $y_i = 1$, which is a high probability of $y_i = 0$)*

Learning the Weights

*to make everything simpler, **assume** that the labels are $y = 1$ (for positive examples) and $y = 0$ (for negative examples, instead of $y = -1$)*

$$\max_{\boldsymbol{w},b} \sum_{i=1}^{n} y_i \log \frac{P(y_i = 1|\boldsymbol{x}_i, \boldsymbol{w}, b)}{P(y_i = 0|\boldsymbol{x}_i, \boldsymbol{w}, b)} + \log P(y_i = 0|\boldsymbol{x}_i, \boldsymbol{w}, b)$$
$$= \max_{\boldsymbol{w},b} \sum_{i=1}^{n} y_i (\boldsymbol{w}^T \boldsymbol{x}_i + b) - \log(1 + \exp(\boldsymbol{w}^T \boldsymbol{x}_i + b))$$
$$= \max_{\boldsymbol{w},b} L(\boldsymbol{w}, b)$$

find parameters $\boldsymbol{w}, b$ to maximize the conditional log-likelihood

(Batch) Gradient Ascent for Logistic Regression

Initialize: $w = w_0, b = b_0, t = 0$

Iterate until convergence

Compute updates:

$$w_{t+1} = w_t + \eta_t \nabla_w L(f(\boldsymbol{x}), y)$$
$$b_{t+1} = b_t + \eta_t \nabla_b L(f(\boldsymbol{x}), y)$$

Check for convergence

Continue to next iteration: $t = t + 1$

$$\nabla_{\boldsymbol{w}} L = \sum_{i=1}^{n} (y_i - p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{w})) \cdot \boldsymbol{x}_i$$

$$\nabla_b L = \sum_{i=1}^{n} (y_i - p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{w}))$$

*gradient depends $y_i - p(y_i = 1|\boldsymbol{x}_i)$, the difference between the **true label** and the **predicted probability***

- *if $y_i = 1$ (positive example), the gradient pushes $p(y_i = 1|\boldsymbol{x}_i)$ closer to 1 (hopefully resulting in a high probability of $y_i = 1$)*
- *if $y_i = 0$ (negative example), the gradient pushes $p(y_i = 1|\boldsymbol{x}_i)$ closer to 0 (hopefully resulting in a low probability of $y_i = 1$, which is a high probability of $y_i = 0$)*

- Batch algorithm: use all the data points together; $L(\boldsymbol{w}, b)$ is a **concave function**, so **gradient ascent** (because maximization) will converge to a **global minimum**
- Online algorithm: use (small chunks or) one data point at a time; leads to **stochastic gradient descent**, which is **highly efficient**

Priors and Regularization

Overfitting the training data is possible in Logistic Regression, especially when data is very high dimensional and training data is sparse; can be avoided by adding a **prior**, which leads to a penalized log-likelihood

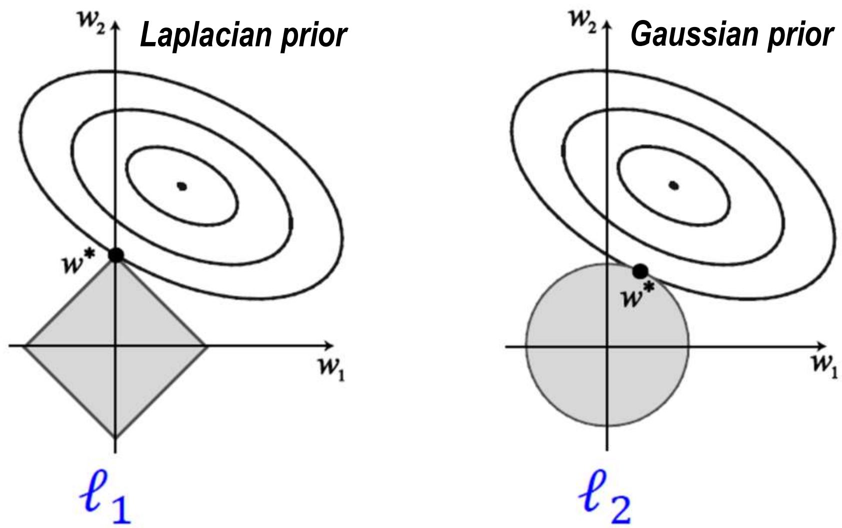
$$\max_{\boldsymbol{w},b} \sum_{i=1}^{n} \log P(y_i|\boldsymbol{x}_i, \boldsymbol{w}, b) + \lambda \log \mathrm{P}(\boldsymbol{w})$$

Consider a **prior distribution** on the weights to prevent overfitting

- assume weights from a normal distribution with zero mean, identity covariance: $\mathrm{P}(\boldsymbol{w}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|\boldsymbol{w}\|^2}{2\sigma^2}\right)$
- **maximizing** $\mathrm{P}(\boldsymbol{w})$ pushes weights to zero, which **minimizes the complexity** of the resulting classifier; this in turn also helps avoid large weights and overfitting
- taking the logarithm gives us $\log \mathrm{P}(\boldsymbol{w}) = -\|\boldsymbol{w}\|^2 + \mathrm{const}$

$$\max_{\boldsymbol{w},b} \sum_{i=1}^{n} y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - \log(1 + \exp(\boldsymbol{w}^T\boldsymbol{x}_i + b)) \; - \frac{\lambda}{2} \|\boldsymbol{w}\|^2$$

- can **also** be solved by **gradient ascent**
  - batch algorithm still has global optimal solution
  - online algorithm with one/few data points at a time will lead to a stochastic gradient descent algorithm
- **regularization parameter:** $\lambda > 0$



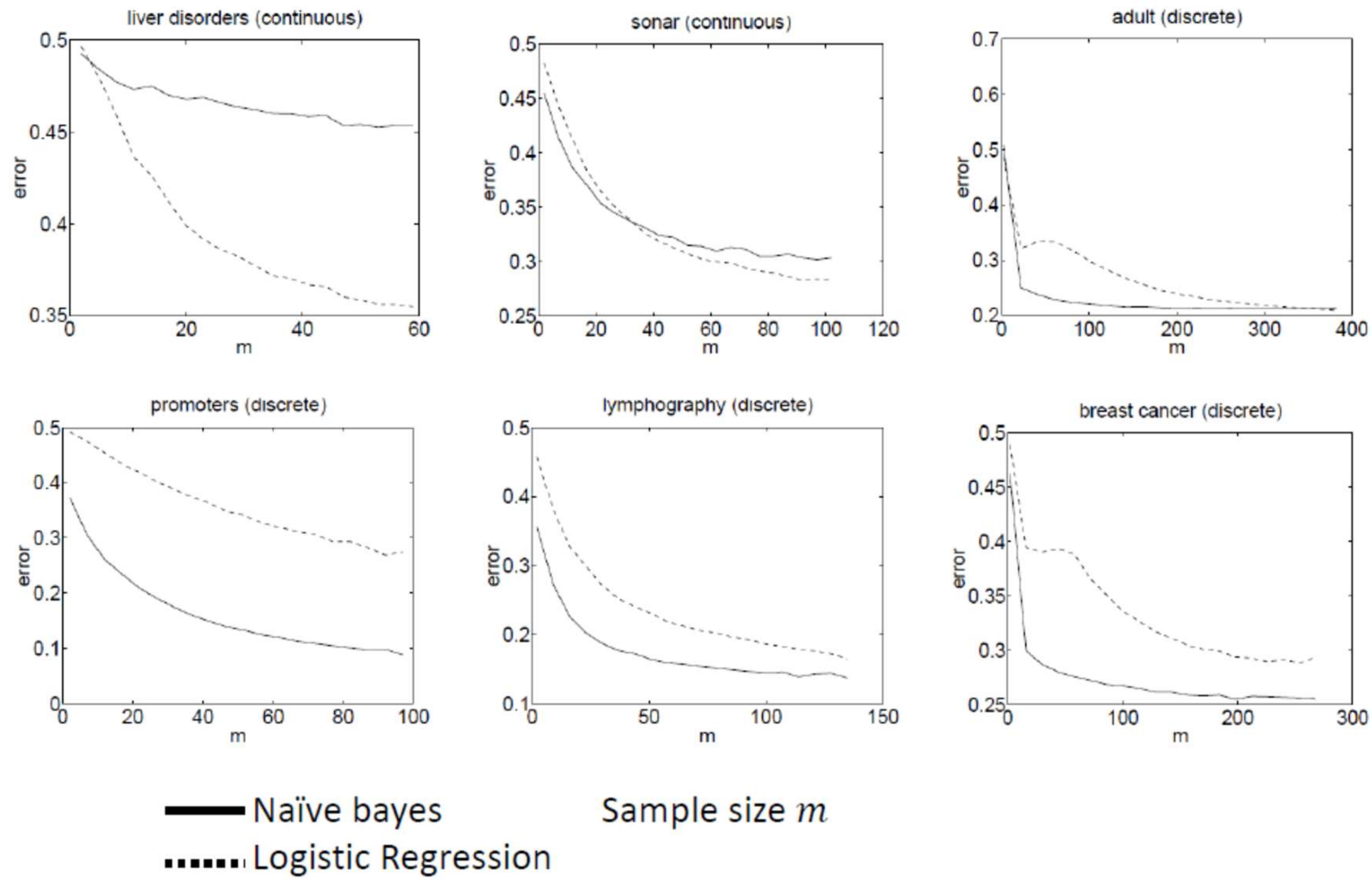
different priors can lead to different regularization functions

Naïve Bayes vs. Logistic Regression

Non-asymptotic analysis (for Gaussian NB)

Size of training data to get close to infinite data solution, (m= # of attributes/features in X)

- **Naïve Bayes** needs $O(\log m)$ samples
  *NB converges quickly to its (perhaps less helpful) asymptotic estimates; makes very strong **independence assumptions***
- **Logistic Regression** needs $O(m)$ samples
  LR converges more slowly but makes no independence assumptions



Multiclass Logistic Regression

Let there be C classes: $c = 1, \ldots, C$. Choose class C to be the **reference class** and represent each of the other classes as a logistic function with respect to class C:

$$\log \frac{p(y = 1 | \boldsymbol{x})}{p(y = C | \boldsymbol{x})} = \boldsymbol{w}_1^T \boldsymbol{x} + b_1 \qquad p(y = 1 | \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_1^T \boldsymbol{x} + b_1)}{1 + \sum_{c=1}^{C-1} \exp(\boldsymbol{w}_c^T \boldsymbol{x} + b_c)}$$

$$\vdots$$

$$\log \frac{p(y = c | \boldsymbol{x})}{p(y = C | \boldsymbol{x})} = \boldsymbol{w}_c^T \boldsymbol{x} + b_c \qquad p(y = c | \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_c^T \boldsymbol{x} + b_c)}{1 + \sum_{c=1}^{C-1} \exp(\boldsymbol{w}_c^T \boldsymbol{x} + b_c)}$$

$$\vdots$$

$$\log \frac{p(y = C-1 | \boldsymbol{x})}{p(y = C | \boldsymbol{x})} = \boldsymbol{w}_{C-1}^T \boldsymbol{x} + b_{C-1} \qquad p(y = C-1 | \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_{C-1}^T \boldsymbol{x} + b_{C-1})}{1 + \sum_{c=1}^{C-1} \exp(\boldsymbol{w}_c^T \boldsymbol{x} + b_c)}$$

$$p(y = C | \boldsymbol{x}) = \frac{1}{1 + \sum_{c=1}^{C-1} \exp(\boldsymbol{w}_c^T \boldsymbol{x} + b_c)}$$

Gradient ascent can be applied **simultaneously** to train all the weight vectors and bias constants.