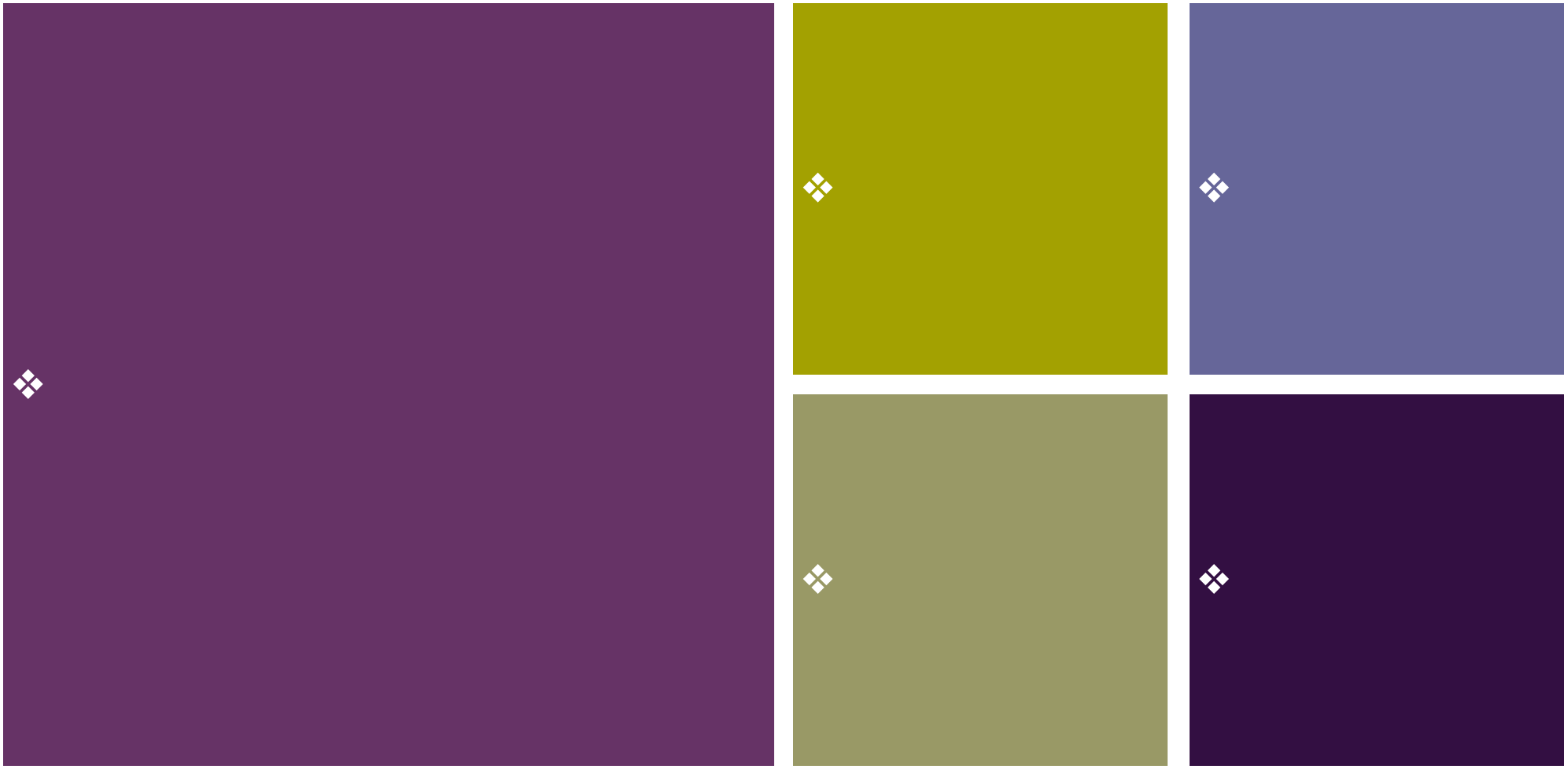


Pattern Recognition and Image Analysis

Dr. Manal Helal – Fall 2015

Lecture 4

Solving Posteriors (Deterministic)

Naive Bayes

Naïve Bayes and Statistical Independency

Naïve Bayes is

- still widely (and successfully) used
- often outperforming much more advanced classifiers
- appropriate in the presence of high dimensional features (curse of dimensionality)
- also called “Idiot’s Bayes”

Naïve Bayes and Statistical Independency (cont.)

For the class dependent pdf we can do the following factorization:

$$
\begin{aligned}
p(\boldsymbol{x}|y) &= p(x_1, x_2, \ldots, x_d|y) \\
&= p(x_1|y)p(x_2, x_3, \ldots, x_d|y, x_1) \\
&= p(x_1|y)p(x_2|y, x_1)p(x_3, x_4, \ldots, x_d|y, x_1, x_2) \\
&= p(x_1|y)\prod_{i=2}^{d} p(x_i|y, x_1, \ldots, x_{i-1})
\end{aligned}
$$

Naïve Bayes and Statistical Independency (cont.)

- The Naïve Bayes classifier makes a very strong – so to call naïve – independency assumption.
- All d components of the feature vector $\boldsymbol{x}$ are assumed to be mutually independent.
- This independency assumption implies:

$$p(\boldsymbol{x}|y) = \prod_{i=1}^{d} p(x_i|y)$$

Naïve Bayes and Statistical Independency (cont.)

The decision rule of naïve Bayes reads as follows:

$$
\begin{aligned}
y^* &= \operatorname*{argmax}_y p(y|\boldsymbol{x}) \\
&= \operatorname*{argmax}_y p(y)p(\boldsymbol{x}|y) \\
&= \operatorname*{argmax}_y p(y) \prod_{i=1}^{d} p(x_i|y)
\end{aligned}
$$

An Example: Naïve Bayes and Gaussians

Example

Assume the 100–dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^{100}$ belonging to class y is normally distributed and all components are *mutually dependent*:

$$\begin{aligned} \boldsymbol{\mu}_y &\in \mathbb{R}^{100} \\ \boldsymbol{\Sigma} &= \boldsymbol{\Sigma}^T \in \mathbb{R}^{100\times 100} \end{aligned}$$

The total number of parameters to be estimated for each class is

$$100 + 100 \cdot (100 + 1)/2 = 5150.$$

An Example: Naïve Bayes and Gaussians (cont.)

Example cont.

Assume the 100–dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^{100}$ belonging to class y is normally distributed and all components are *mutually independent*.

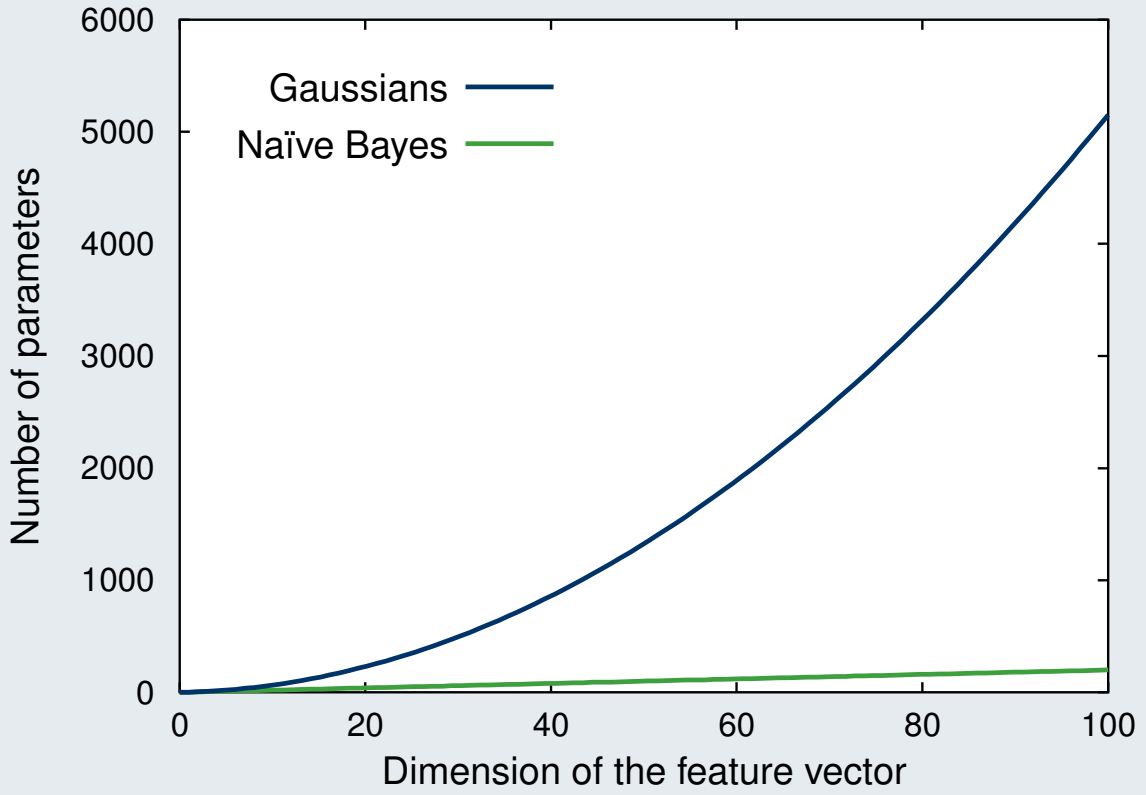
$$p(\boldsymbol{x}|y) = \prod_{i=1}^{100} p(x_i|y) \quad = \quad \prod_{i=1}^{100} \mathcal{N}(x_i; \mu_i, \sigma_i^2).$$

For each component $i = \{1, 2, 3, \ldots, 100\}$ we have to estimate mean $\mu_i \in \mathbb{R}$ and variance $\sigma_i^2 \in \mathbb{R}$. The total number of parameters to be estimated for each class is

$$100 + 100 = 200.$$

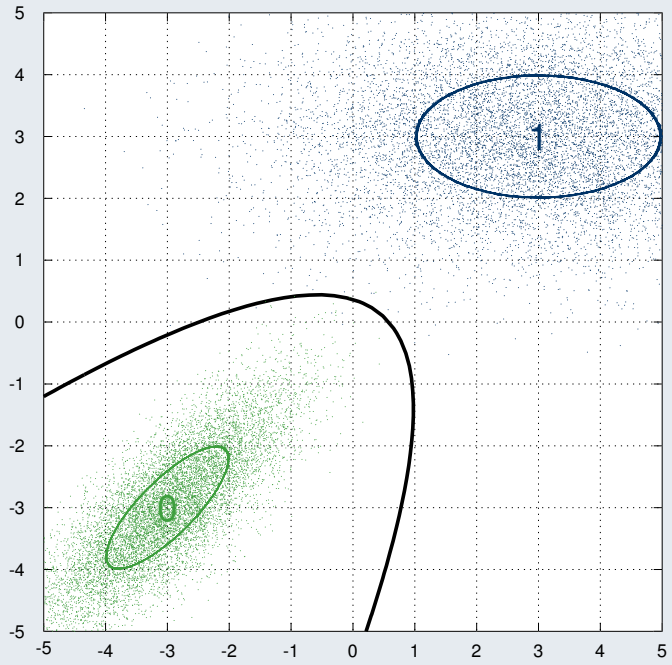
An Example: Naïve Bayes and Gaussians (cont.)

Example cont.



An Example: Naïve Bayes and Gaussians (cont.)

Example cont.



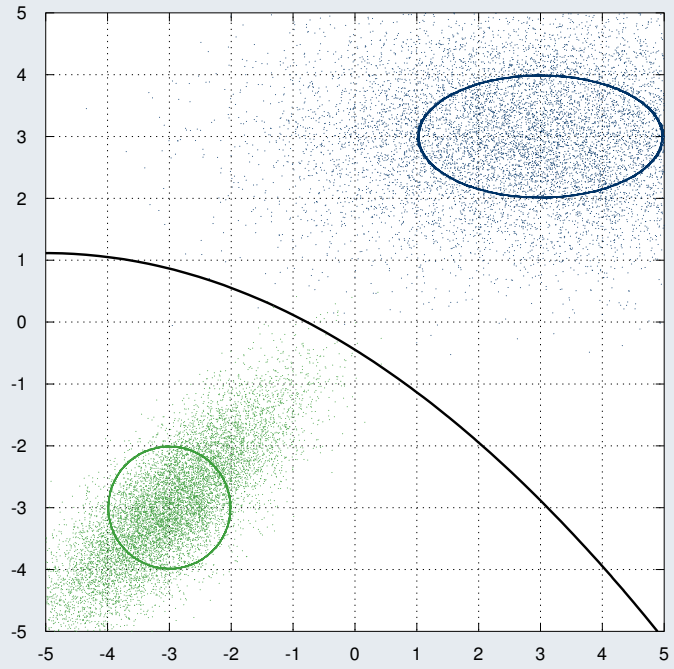
$p(y = 0) = 0.5$

$p(y = 1) = 0.5$

Fig.: Quadratic decision boundary that considers statistical dependency

An Example: Naïve Bayes and Gaussians (cont.)

Example cont.



$p(y = 0) = 0.5$

$p(y = 1) = 0.5$

Fig.: Quadratic decision boundary assuming independency of x_1 and x_2

Naïve Bayes

Let us consider the logit transform

$$\begin{aligned}
\log \frac{p(y=0|\boldsymbol{x})}{p(y=1|\boldsymbol{x})} &= \log \frac{p(y=0)p(\boldsymbol{x}|y=0)}{p(y=1)p(\boldsymbol{x}|y=1)} \\
&= \log \frac{p(y=0)}{p(y=1)} + \log \frac{p(\boldsymbol{x}|y=0)}{p(\boldsymbol{x}|y=1)} \\
&= \log \frac{p(y=0)}{p(y=1)} + \log \frac{\prod_{i=1}^{d} p(x_i|y=0)}{\prod_{i=1}^{d} p(x_i|y=1)} \\
&= \underbrace{\alpha_0 + \sum_{i=1}^{d} \alpha_{0,i}(x_i)}_{\text{generalized additive model}}
\end{aligned}$$

Naïve Bayes (cont.)

Is there anything between Bayes and Naïve Bayes?

Naïve Bayes (cont.)

There are multiple techniques to beat the curse of dimensionality, for example:

- Reduction of the parameter space
  - Introduction of independency assumptions
    (from complete dependency to mutual independency)
  - Parameter tying
- Reduction of the dimension of the feature vectors

Naïve Bayes (cont.)

First order dependency

$$
\begin{aligned}
p(\boldsymbol{x}|y) &= p(x_1, x_2, \ldots, x_d|y) \\
&= p(x_1|y)p(x_2, x_3, \ldots, x_d|y, x_1) \\
&= p(x_1|y)p(x_2|y, x_1)p(x_3, x_4, \ldots, x_d|y, x_1, x_2) \\
&= p(x_1|y)\prod_{i=2}^{d} p(x_i|y, x_{i-1})
\end{aligned}
$$

Naïve Bayes (cont.)

Example

First order dependency in a Gaussian random vector can be identified through the covariance matrix $\mathbf{\Sigma}$. It has the following structure:

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_{1,1} & \sigma_{2,1} & 0 & 0 & \cdots & 0 & 0 \\ \sigma_{2,1} & \sigma_{2,2} & \sigma_{3,2} & 0 & \cdots & 0 & 0 \\ 0 & \sigma_{3,2} & \sigma_{3,3} & \sigma_{4,3} & \cdots & 0 & 0 \\ 0 & 0 & \sigma_{4,3} & \sigma_{4,4} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \sigma_{d,d-1} \\ 0 & 0 & 0 & 0 & \cdots & \sigma_{d,d-1} & \sigma_{d,d} \end{pmatrix}$$

Naïve Bayes (cont.)

Example

First order dependency in Gaussian random vector with tied diagonal elements, i. e. $\sigma_{i,i} = \sigma$:

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma & \sigma_{2,1} & 0 & 0 & \cdots & 0 & 0 \\ \sigma_{2,1} & \sigma & \sigma_{3,2} & 0 & \cdots & 0 & 0 \\ 0 & \sigma_{3,2} & \sigma & \sigma_{4,3} & \cdots & 0 & 0 \\ 0 & 0 & \sigma_{4,3} & \sigma & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \sigma_{d,d-1} \\ 0 & 0 & 0 & 0 & \cdots & \sigma_{d,d-1} & \sigma \end{pmatrix}$$

Maximum Likelihood Estimation

Parameters Estimation

Parameterization

- Until now, $F(\boldsymbol{x})$ was some arbitrary function in $\boldsymbol{x}$

  Example: $F(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\alpha}^T \boldsymbol{x} + \alpha_0$ with components defined by Gaussian distributions

- We can express a nonlinear $F(\boldsymbol{x})$ as a scalar product by lifting $\boldsymbol{x}$ into a higher dimensional space:
  Given

$$\boldsymbol{x} = (x_1, x_2)^T \in \mathbb{R}^2, \ \boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \ \boldsymbol{\alpha} = (\alpha_1, \alpha_2)^T, \ \alpha_0 ,$$

  then

$$F(\boldsymbol{x}) = a_{11}x_1^2 + (a_{12} + a_{21})x_1x_2 + a_{22}x_2^2 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_0 .$$

- Rewrite $F(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}'$ with $\boldsymbol{\theta}, \boldsymbol{x}' \in \mathbb{R}^6$:

$$\boldsymbol{\theta} = (a_{11}, a_{12} + a_{21}, a_{22}, \alpha_1, \alpha_2, \alpha_0)^T$$
$$\boldsymbol{x}' = (x_1^2, x_1x_2, x_2^2, x_1, x_2, 1)^T$$

Parameterization (cont.)

Definition

We write the parameterized logistic function in the following:

$$g(\boldsymbol{\theta}^T \boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$$

where $\boldsymbol{\theta}, \boldsymbol{x}$ are the lifted parameters of the original decision function F (if it was not already linear).

Log-Likelihood Function

- Let us assume the posteriors are given by

$$\begin{aligned} p(y=0|\boldsymbol{x}) &= 1 - g(\boldsymbol{\theta}^T \boldsymbol{x}) \\ p(y=1|\boldsymbol{x}) &= g(\boldsymbol{\theta}^T \boldsymbol{x}) \end{aligned}$$

  where $g(\boldsymbol{\theta}^T \boldsymbol{x})$ is the sigmoid function parameterized in $\boldsymbol{\theta}$.

- The parameter vector $\boldsymbol{\theta}$ has to be estimated from a set S of m training samples:

$$S = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), (\boldsymbol{x}_3, y_3), \ldots, (\boldsymbol{x}_m, y_m)\} \quad .$$

- Method of choice: Maximum Likelihood Estimation

Log-Likelihood Function (cont.)

Before we work on the formulas of the ML-estimator, we rewrite the posteriors using Bernoulli probability:

$$p(y|\boldsymbol{x}) \quad = \quad g(\boldsymbol{\theta}^T \boldsymbol{x})^y (1 - g(\boldsymbol{\theta}^T \boldsymbol{x}))^{1-y}$$

which shows the great benefit of the chosen notation for class numbers.

Log-Likelihood Function (cont.)

Now we can compute the log-likelihood function
(assuming that the training samples are mutually independent):

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= \log\left(\prod_{i=1}^{m} p(y_i|\boldsymbol{x}_i)\right) \\
&= \sum_{i=1}^{m} \log\left(g(\boldsymbol{\theta}^T\boldsymbol{x}_i)^{y_i}\left(1-g(\boldsymbol{\theta}^T\boldsymbol{x}_i)\right)^{1-y_i}\right) \\
&= \sum_{i=1}^{m} \left(y_i \log g(\boldsymbol{\theta}^T\boldsymbol{x}_i) + (1-y_i)\log\left(1-g(\boldsymbol{\theta}^T\boldsymbol{x}_i)\right)\right)
\end{aligned}
$$

Log-Likelihood Function (cont.)

Simplification of the log-likelihood function:

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= \sum_{i=1}^{m} \left(y_i \log g(\boldsymbol{\theta}^T \boldsymbol{x}_i) + (1 - y_i) \log \left(1 - g(\boldsymbol{\theta}^T \boldsymbol{x}_i)\right) \right) \\
&= \sum_{i=1}^{m} \left(y_i \log \frac{e^{\boldsymbol{\theta}^T \boldsymbol{x}_i}}{1 + e^{\boldsymbol{\theta}^T \boldsymbol{x}_i}} + (1 - y_i) \log \frac{1}{1 + e^{\boldsymbol{\theta}^T \boldsymbol{x}_i}} \right) \\
&= \sum_{i=1}^{m} \left(y_i \boldsymbol{\theta}^T \boldsymbol{x}_i + \log \frac{1}{1 + e^{\boldsymbol{\theta}^T \boldsymbol{x}_i}} \right) \\
&= \sum_{i=1}^{m} \left(y_i \boldsymbol{\theta}^T \boldsymbol{x}_i + \log \left(1 - g(\boldsymbol{\theta}^T \boldsymbol{x}_i)\right) \right)
\end{aligned}
$$

Maximization of the log-likelihood function

- The log-likelihood function is concave.
- We use the ☞ Newton-Raphson algorithm to solve the unconstrained optimization problem:

  For the $(k+1)$-st iteration step, we get:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \left(\frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \mathcal{L}\left(\boldsymbol{\theta}^{(k)}\right) \right)^{-1} \frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}^{(k)}\right)$$

Note: If you write the Newton-Raphson iteration in matrix form, you will end up with a weighted least squares iteration scheme.

Newton-Raphson Iteration

Taylor's Theorem:

Approximation of a k-times differentiable function $f(x)$ around a given point x_0:

$$f(x_0+h) = f(x_0)+f'(x_0)h+\frac{f''(x_0)}{2!}h^2+\ldots+\frac{f^{(k)}(x_0)}{k!}h^k+r_k(x_0+h)h^k, \quad \lim_{h\to 0} r_k(x_0+h) = 0$$

Second order Taylor polynomial:

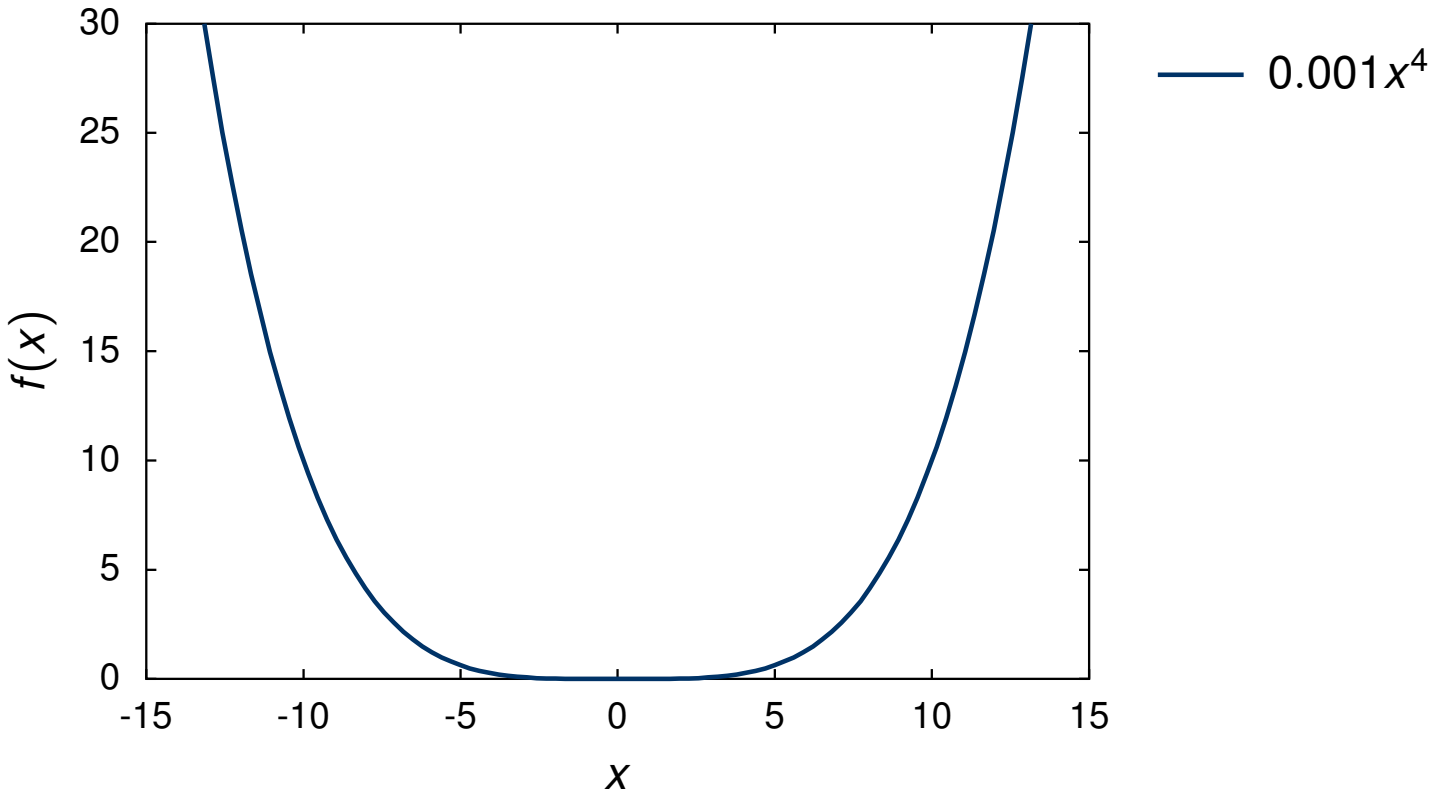
$$f(x_0+h) \approx f(x_0)+f'(x_0)h+\frac{1}{2}f''(x_0)h^2$$

Newton-Raphson Iteration (cont.)

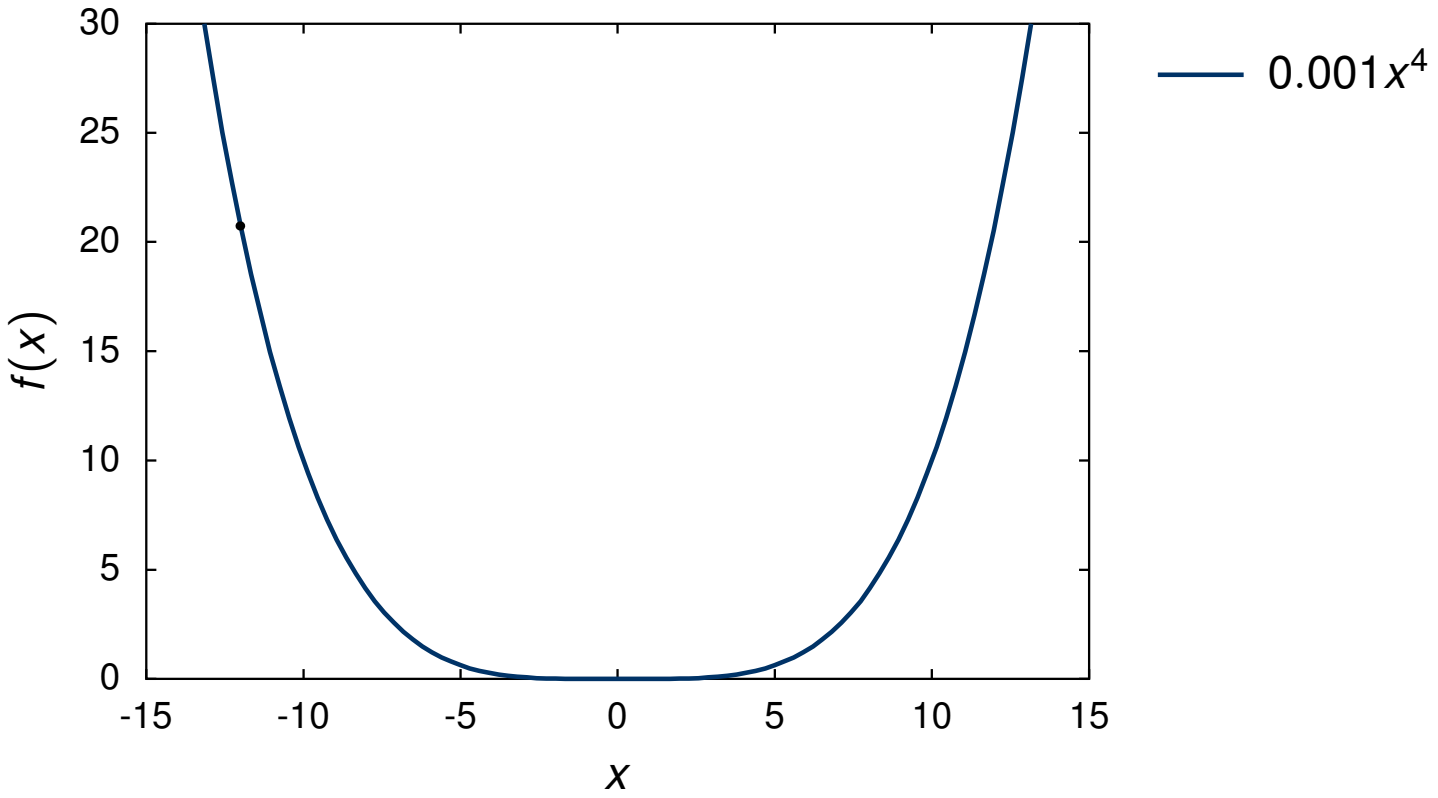
Extremum:

$$
\begin{aligned}
f'(x_0 + h) &= f'(x_0) + f''(x_0)h \stackrel{!}{=} 0 \\
\hat{h} &= -\frac{f'(x_0)}{f''(x_0)} \\
x_1 = x_0 + \hat{h} &= x_0 - \frac{f'(x_0)}{f''(x_0)}
\end{aligned}
$$

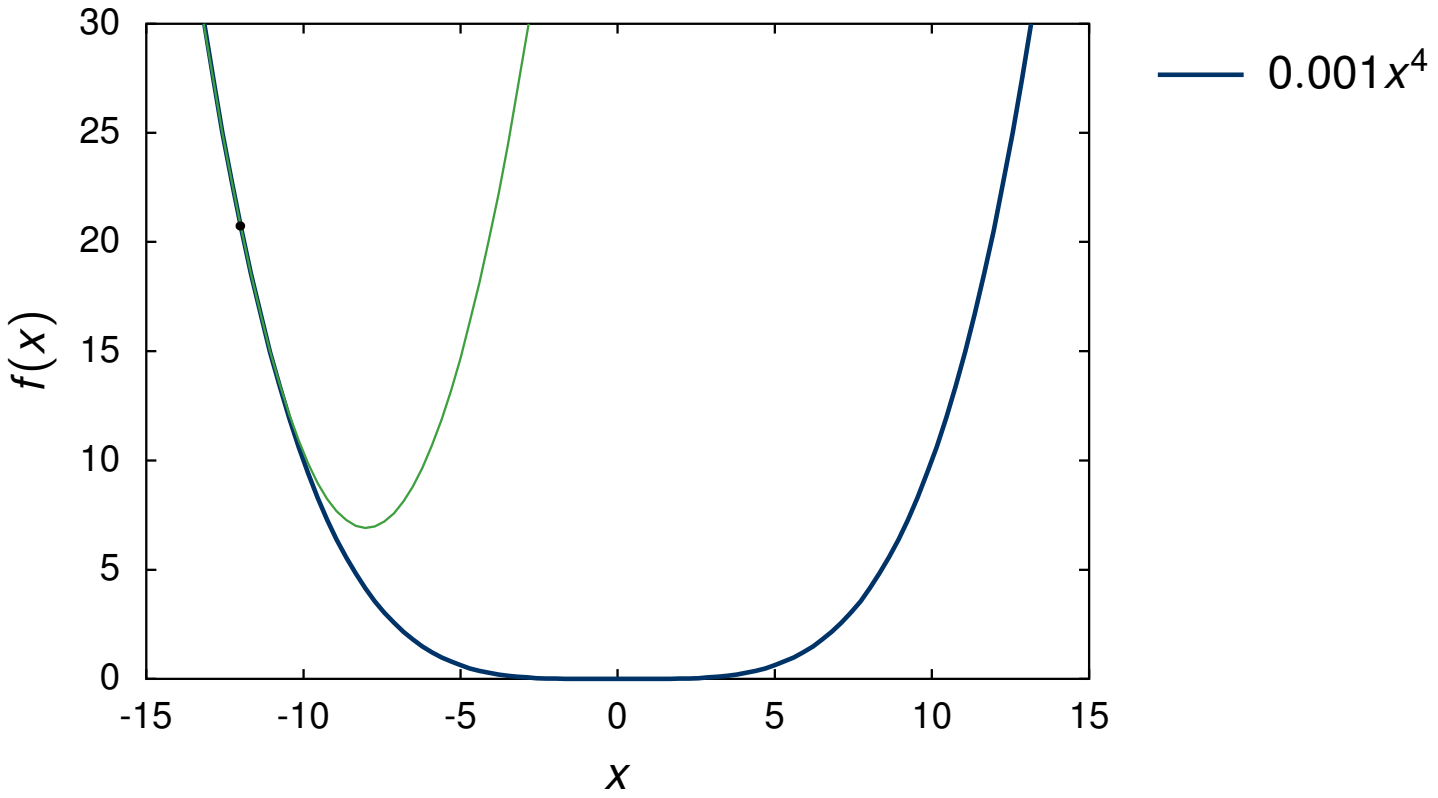
Newton-Raphson Iteration (cont.)



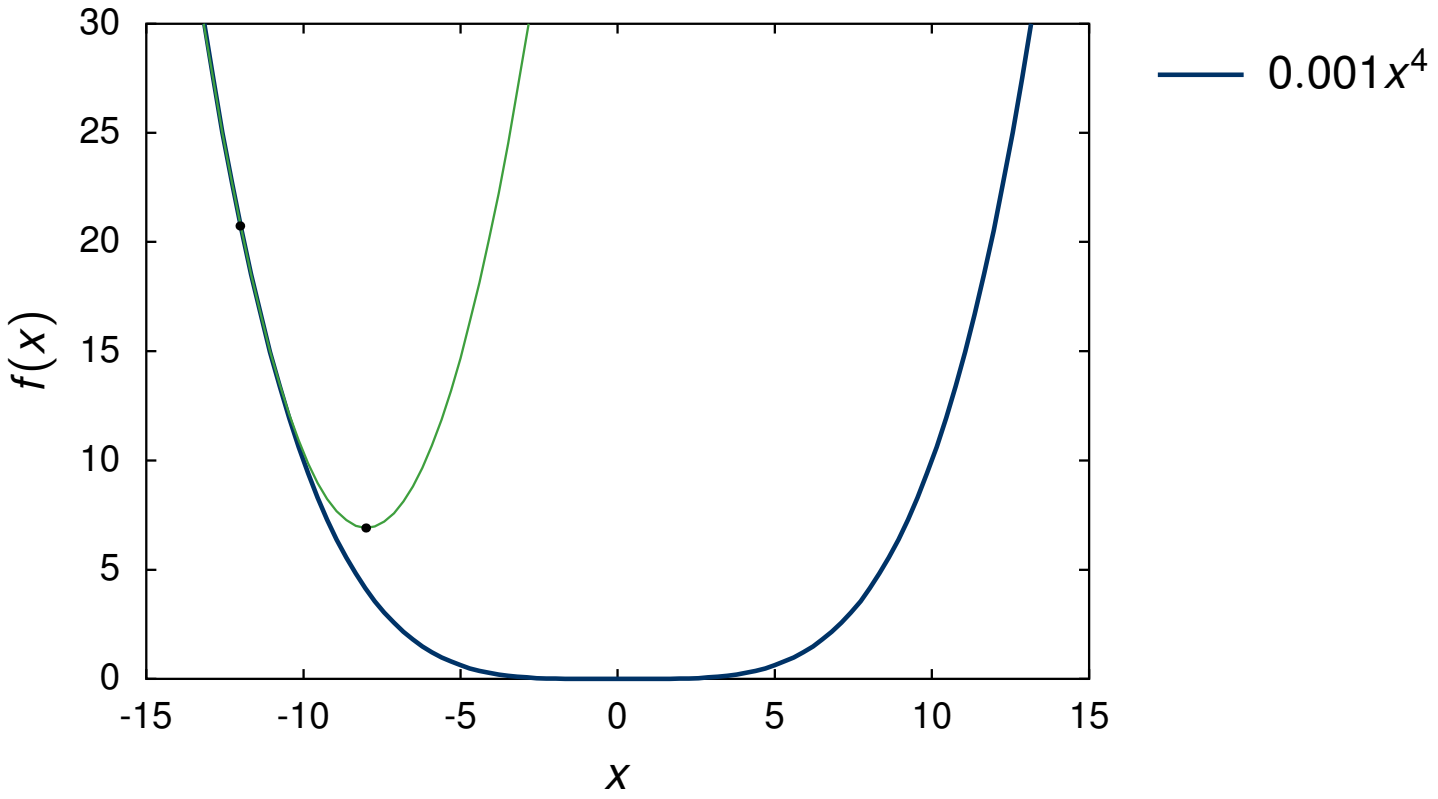
Newton-Raphson Iteration (cont.)



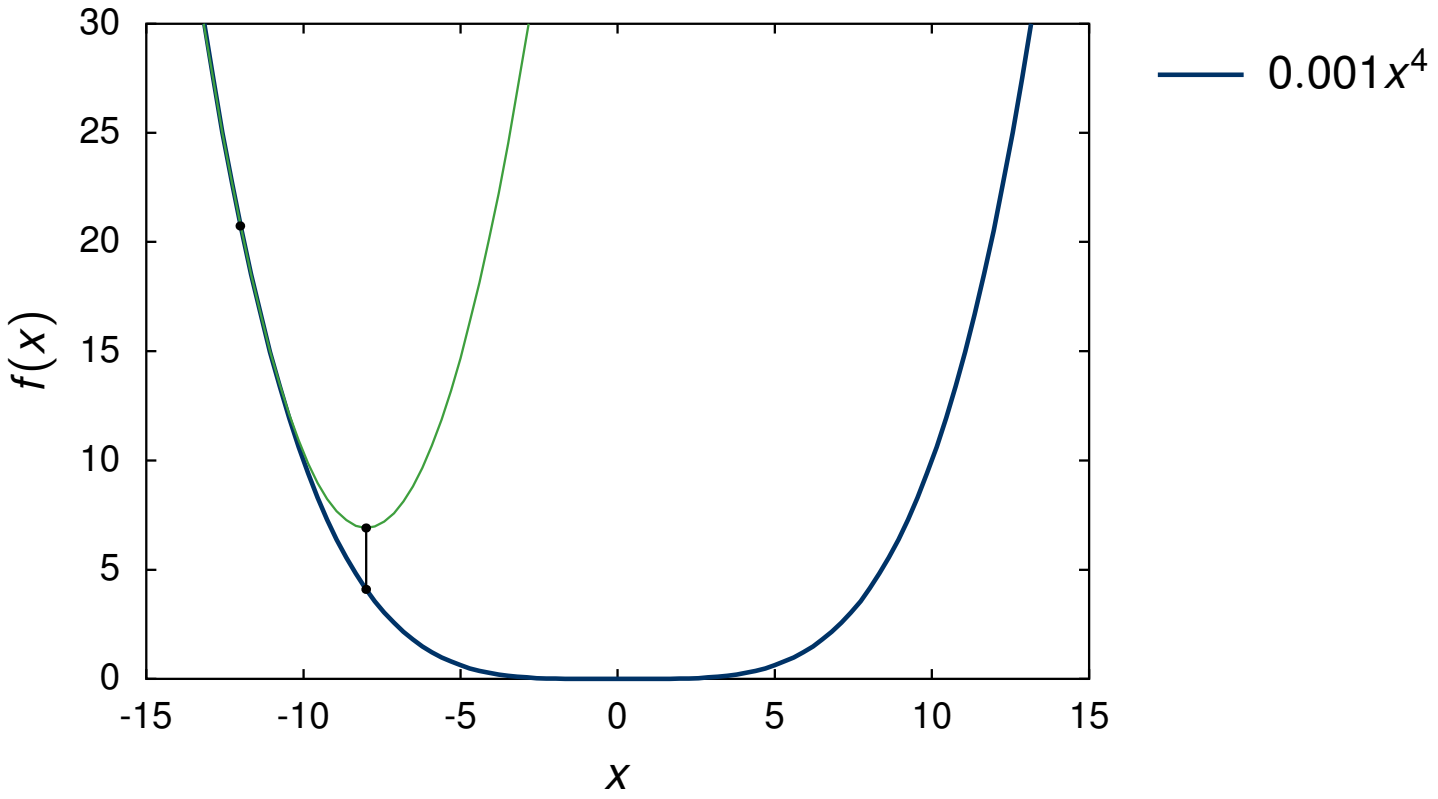
Newton-Raphson Iteration (cont.)



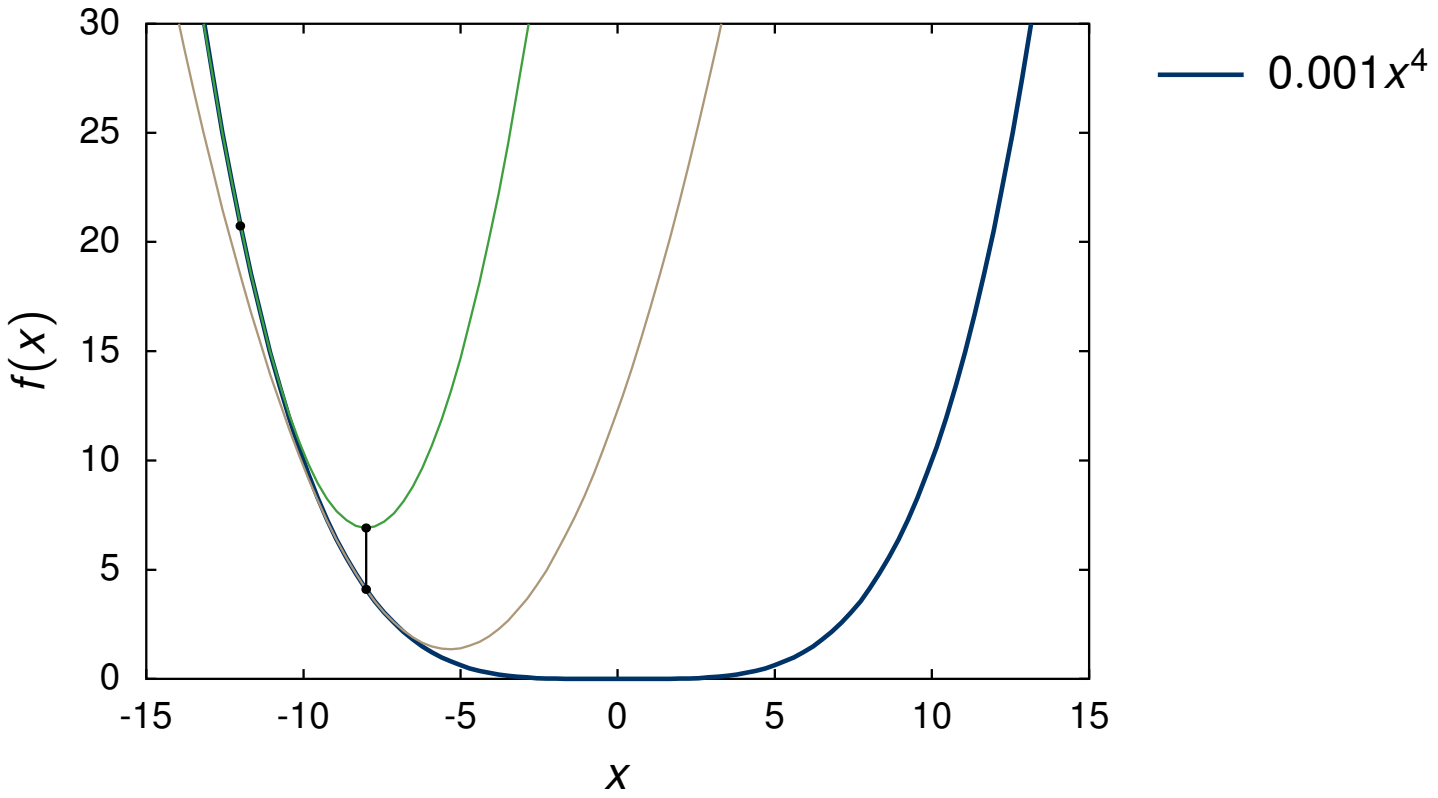
Newton-Raphson Iteration (cont.)



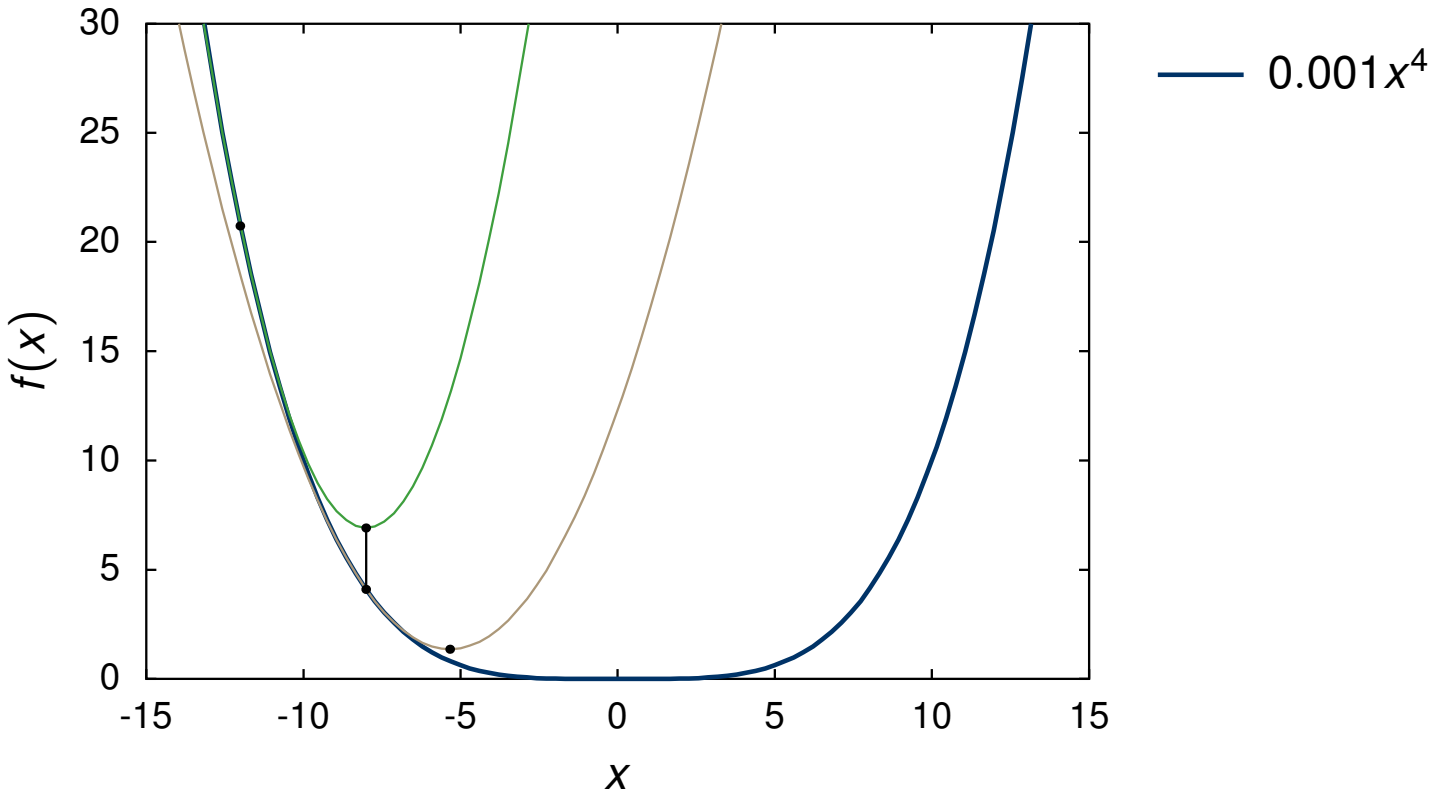
Newton-Raphson Iteration (cont.)



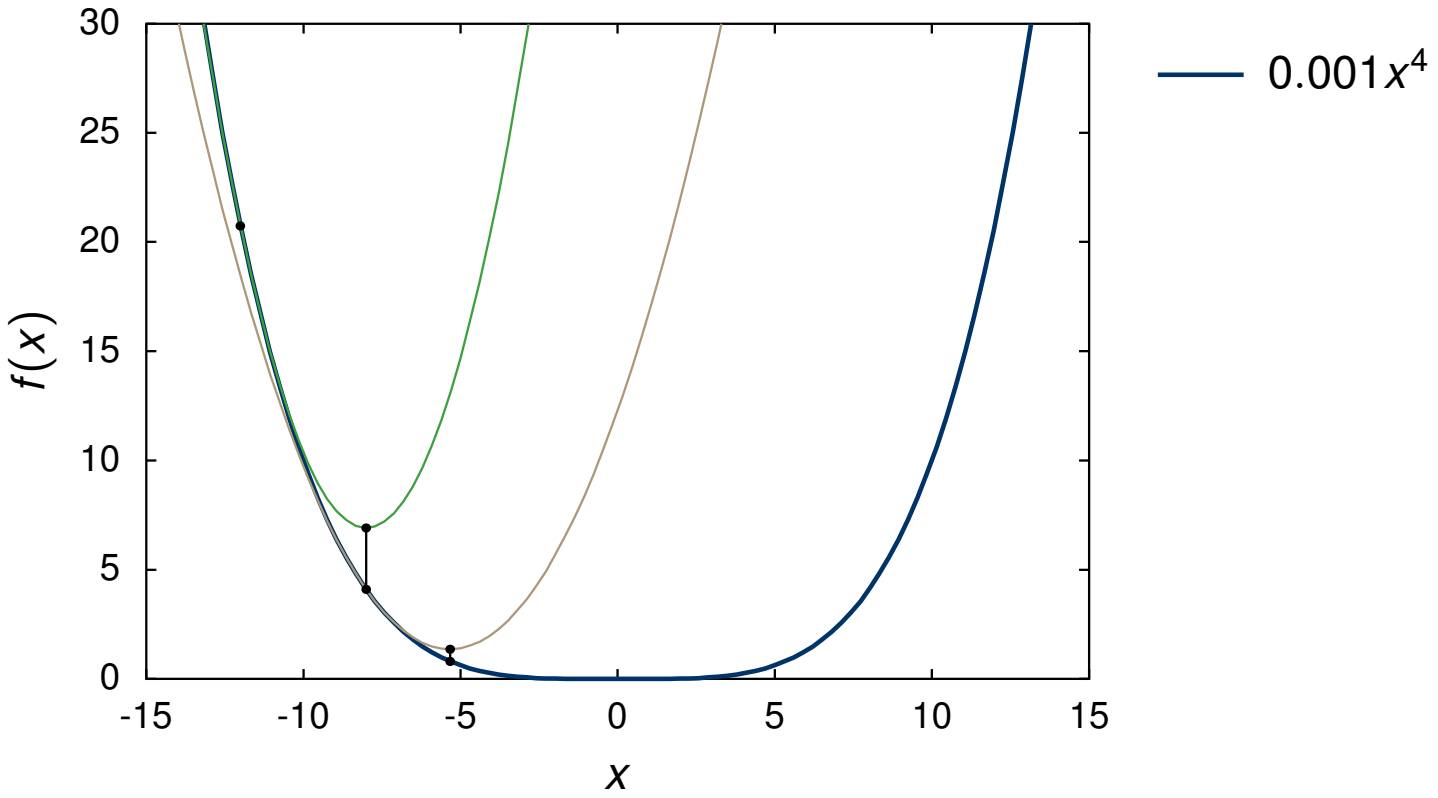
Newton-Raphson Iteration (cont.)



Newton-Raphson Iteration (cont.)



Newton-Raphson Iteration (cont.)



Gradient of the Log-Likelihood Function

The gradient:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\mathcal{L}(\boldsymbol{\theta}) &= \frac{\partial}{\partial\theta_j}\left(\sum_{i=1}^{m}\left(y_i\boldsymbol{\theta}^T\boldsymbol{x}_i + \log\left(1-g(\boldsymbol{\theta}^T\boldsymbol{x}_i)\right)\right)\right)\\ &= \sum_{i=1}^{m}\left(y_i x_{i,j} - \frac{1}{1-g(\boldsymbol{\theta}^T\boldsymbol{x}_i)}\frac{\partial}{\partial\theta_j}g(\boldsymbol{\theta}^T\boldsymbol{x}_i)\right)\end{aligned}$$

Now we use the derivative of the sigmoid function and get

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\mathcal{L}(\boldsymbol{\theta}) &= \sum_{i=1}^{m}\left(y_i x_{i,j} - \frac{1}{1-g(\boldsymbol{\theta}^T\boldsymbol{x}_i)}g(\boldsymbol{\theta}^T\boldsymbol{x}_i)\left(1-g(\boldsymbol{\theta}^T\boldsymbol{x}_i)\right)x_{i,j}\right)\\ &= \sum_{i=1}^{m}\left(y_i - g(\boldsymbol{\theta}^T\boldsymbol{x}_i)\right)x_{i,j}\end{aligned}$$

where $x_{i,j}$ is the j-th component of the i-th training feature vector.

Gradient of the Log-Likelihood Function (cont.)

Finally, we have a quite simple gradient:

$$\frac{\partial}{\partial \theta_j} \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \left(y_i - g(\boldsymbol{\theta}^T \boldsymbol{x}_i) \right) x_{i,j}$$

where $x_{i,j}$ is the j–th component of the i–th training feature vector.

Or in vector notation:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \left(y_i - g(\boldsymbol{\theta}^T \boldsymbol{x}_i) \right) \boldsymbol{x}_i$$

Hessian of the Log-Likelihood Function

- The Newton-Raphson algorithm requires the Hessian matrix.
- Remember the derivative of the sigmoid function!

$$\frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \mathcal{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{m} g(\boldsymbol{\theta}^T \boldsymbol{x}_i) \left(1 - g(\boldsymbol{\theta}^T \boldsymbol{x}_i)\right) \boldsymbol{x}_i \boldsymbol{x}_i^T$$

Example

Comprehensive Questions (cont.)

- How can a nonlinear function be written as a scalar product?
- What is the objective function for the ML-estimation of the logistic regression parameters?
- What is the difference between a gradient descent and Newton-Raphson numerical optimization scheme?
- What is the parameter update rule for the logistic regression parameters using the Newton-Raphson scheme?

Comprehensive Questions

- What is the assumption of Naïve Bayes?
- How does the assumption affect the class dependent pdf?
- What is the structure of the covariance matrix of normal-distributed classes in Naïve Bayes?
- How can Naïve Bayes be extended to first-order statistical dependencies?