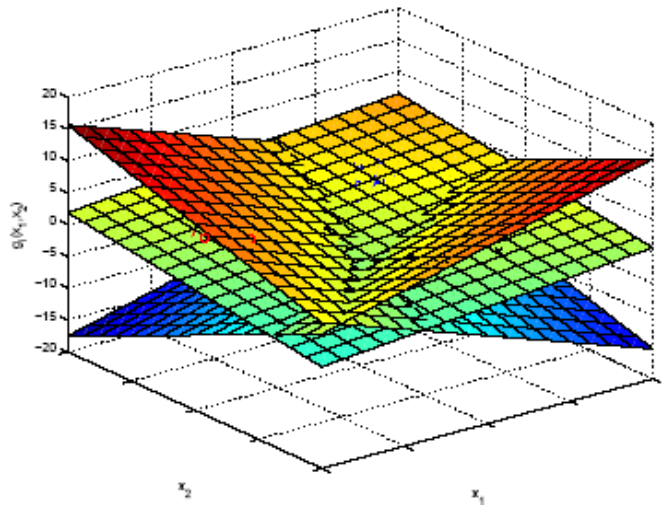
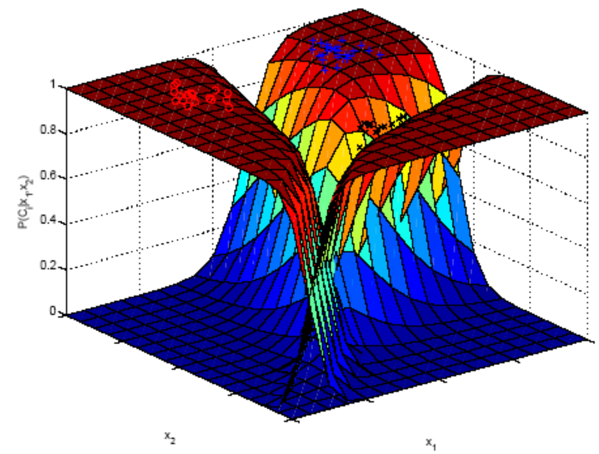
Introduction to Machine Learning





Week 2: Logistic Regression

Iasonas Kokkinos

i.kokkinos@cs.ucl.ac.uk

University College London

Training criterion for logistic regression

Training set: $\{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^N, y^N)\}, \mathbf{x} \in \mathbb{R}^D, y \in \{0, 1\}$

$$L(\mathbf{w}) = -\sum_{i=1}^{N} y^i \log g(\mathbf{w}^T \mathbf{x}^i) + (1 - y^i) \log(1 - g(\mathbf{w}^T \mathbf{x}^i))$$

‘Cross-entropy’, or ‘log loss’

Q1: How does this behave?

Short answer: much better

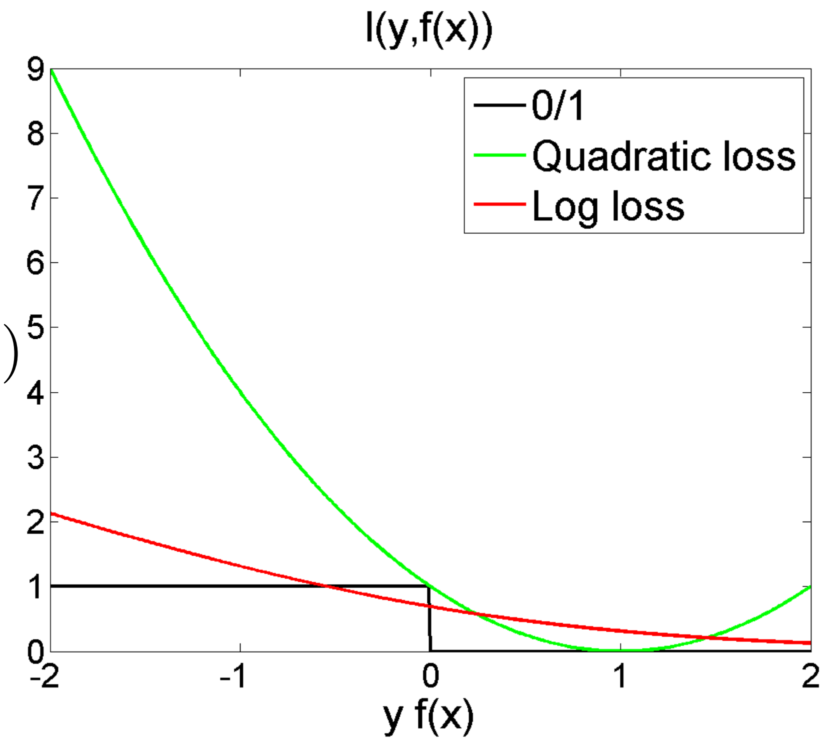
Log loss:

$$l(y, f_{\mathbf{w}}(\mathbf{x})) = \log(1 + \exp(-y f_{\mathbf{w}}(\mathbf{x})))$$

Quadratic loss:

$$l(y, f_{\mathbf{w}}(\mathbf{x})) = (1 - f_{\mathbf{w}}(\mathbf{x}))^2$$

where: $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$



Q2: How to optimize it with respect to w?

Lecture outline

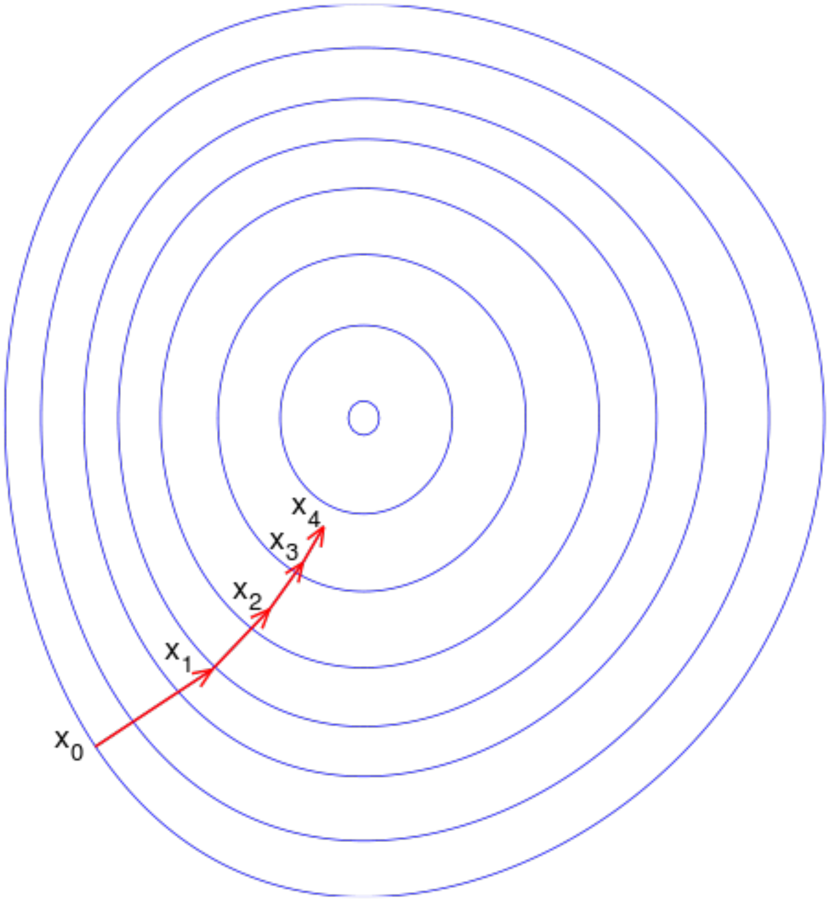
Recap & problems of linear regression

Logistic Regression

Training criterion formulation

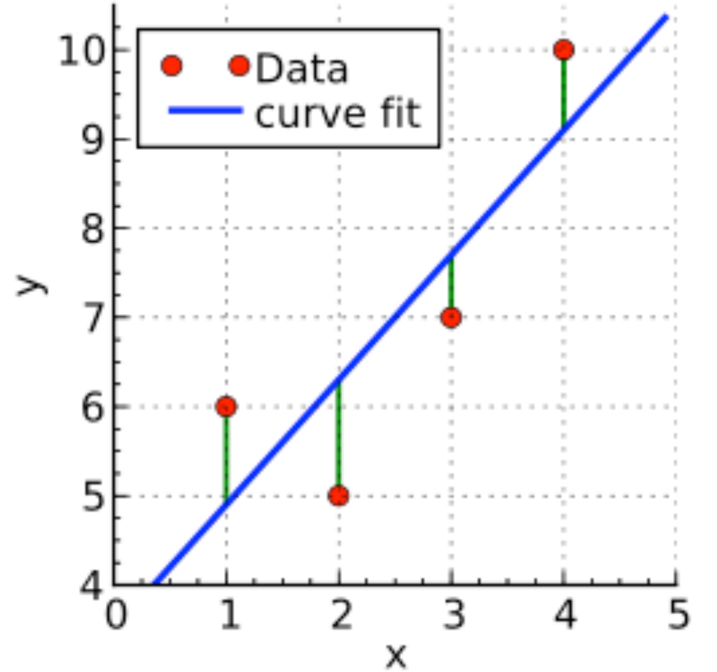
Interpretation

Optimization



Recap: Sum of squared errors criterion

$$y^i = \mathbf{w}^T \mathbf{x}^i + \epsilon^i$$



Loss function: sum of squared errors

$$L(\mathbf{w}) = \sum_{i=1}^{N} (\epsilon^i)^2$$

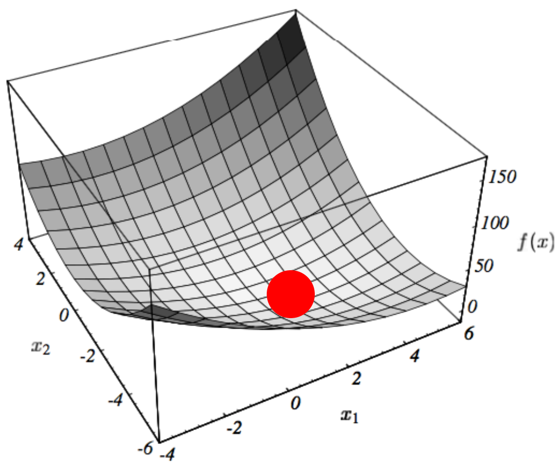
Expressed as a function of two variables:

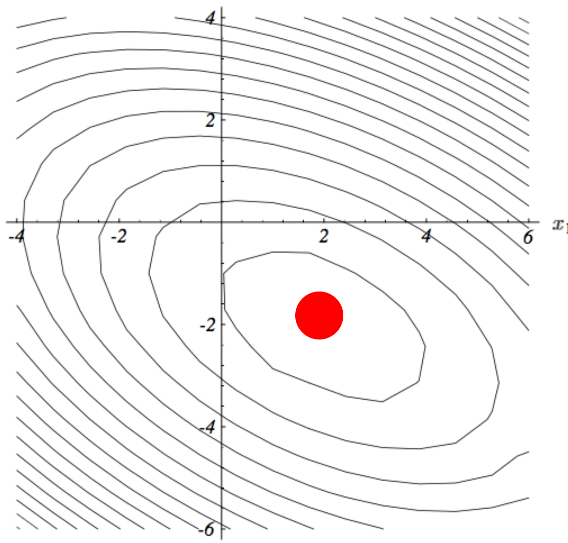
$$L(w_0, w_1) = \sum_{i=1}^{N} \left[y^i - \left(w_0 x_0^i + w_1 x_1^i \right) \right]^2$$

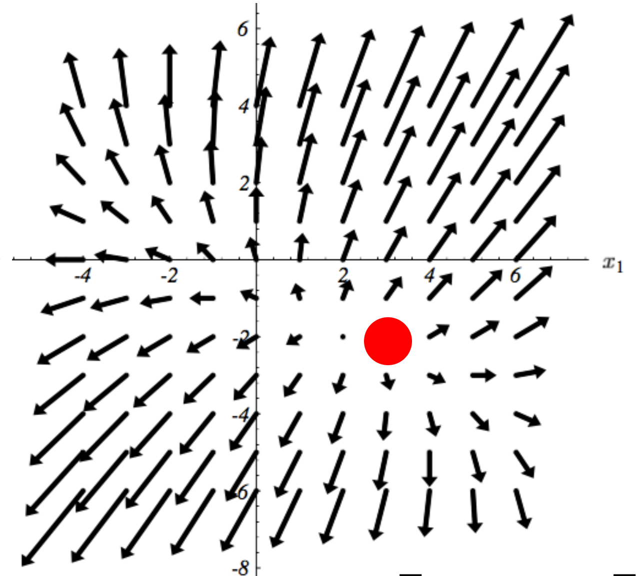
Question: what is the best (or least bad) value of w?

Answer: least squares

Gradient-based optimization







$$f(\mathbf{x})$$

2D function graph

$$f(\mathbf{x}) = c$$

isocontours

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

gradient field

at minimum of function: $\nabla f(\mathbf{x}) = \mathbf{0}$

Recap: condition for optimum

$$\frac{\partial L(w_0, w_1)}{\partial w_0} = 0 \quad \Leftrightarrow$$

$$\sum_{i=1}^{N} y^i x_0^i = w_0 \sum_{i=1}^{N} x_0^i x_0^i + w_1 \sum_{i=1}^{N} x_1^i x_0^i$$

$$\frac{\partial L(w_0, w_1)}{\partial w_1} = 0 \quad \Leftrightarrow$$

$$\sum_{i=1}^{N} y^i x_1^i = w_0 \sum_{i=1}^{N} x_0^i x_1^i + w_1 \sum_{i=1}^{N} x_1^i x_1^i$$

2 linear equations, 2 unknowns

Least squares solution, in vector form

$$
\begin{aligned}
L(\mathbf{w}) &= \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \\
&= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \\
&= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w}
\end{aligned}
$$

Condition for minimum:

$$\nabla L(\mathbf{w}^*) = \mathbf{0}$$

$$-2\mathbf{X}^T \mathbf{y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w}^* = \mathbf{0}$$

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

Gradient of cross-entropy los

$$L(\mathbf{w}) = -\sum_{i=1}^{N} y^i \log g(\mathbf{w}^T\mathbf{x}^i) + (1-y^i)\log(1-g(\mathbf{w}^T\mathbf{x}^i))$$

$$\frac{\partial L(\mathbf{w})}{\partial w_k} = -\sum_{i=1}^{N}\left[y^i \frac{1}{g(\mathbf{w}^T\mathbf{x}^i)}\frac{\partial g(\mathbf{w}^T\mathbf{x}^i)}{\partial w_k} + (1-y^i)\frac{1}{1-g(\mathbf{w}^T\mathbf{x}^i)}\left(-\frac{\partial g(\mathbf{w}^T\mathbf{x}^i)}{\partial w_k}\right)\right]$$

Fact: $g(x) = \frac{1}{1+\exp(-x)} \rightarrow \frac{dg}{dx} = g(x)(1-g(x))$

$$= -\sum_{i=1}^{N}\left[y^i \frac{1}{g(\mathbf{w}^T\mathbf{x}^i)} - (1-y^i)\frac{1}{1-g(\mathbf{w}^T\mathbf{x}^i)}\right] g(\mathbf{w}^T\mathbf{x}^i)(1-g(\mathbf{w}^T\mathbf{x}^i))\frac{\partial \mathbf{w}^T\mathbf{x}^i}{\partial w_k}$$

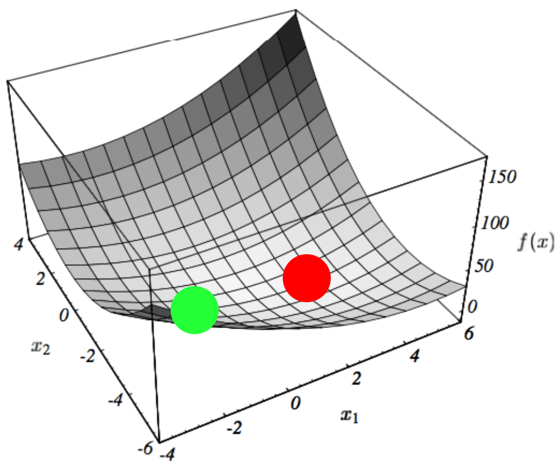
$$= -\sum_{i=1}^{N}\left[y^i(1-g(\mathbf{w}^T\mathbf{x}^i)) - (1-y^i)g(\mathbf{w}^T\mathbf{x}^i)\right] x_k^i$$

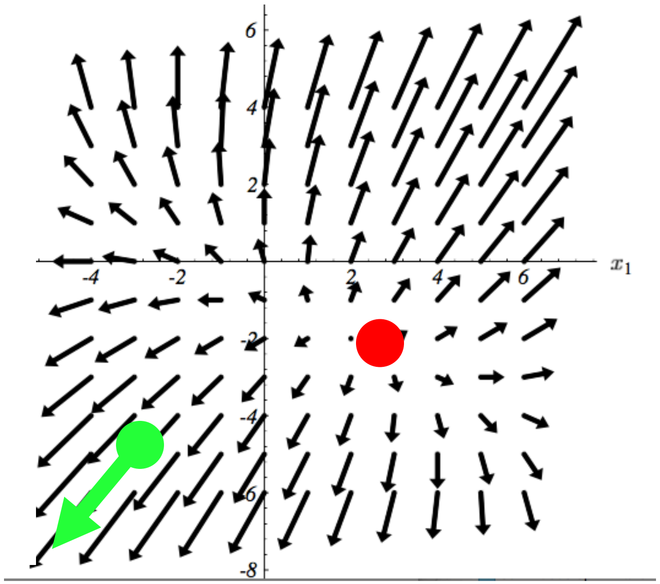
$$= -\sum_{i=1}^{N}\left[y^i - g(\mathbf{w}^T\mathbf{x}^i)\right] \mathbf{x}_k^i$$

$$\nabla L(\mathbf{w}^*) = \mathbf{0}$$

Nonlinear system of equations!!

Gradient-based minimization

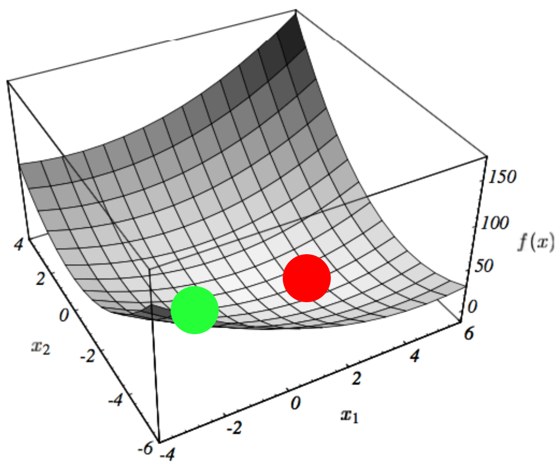


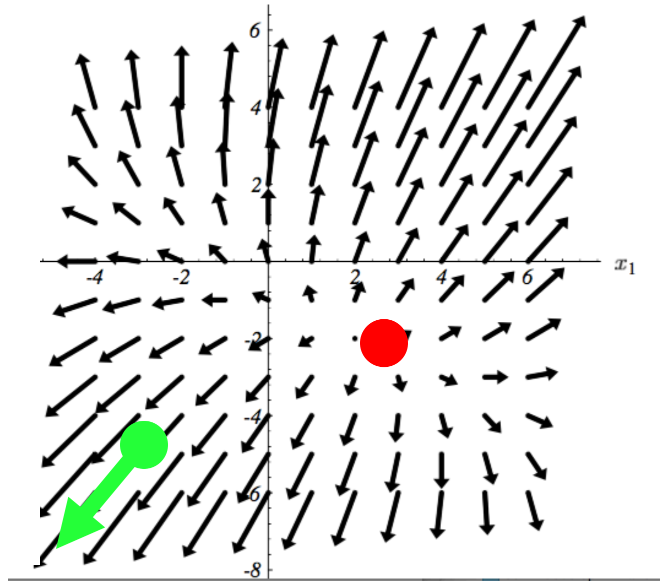


$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Fact: gradient at any point gives direction of fastest increase

Gradient-based minimization



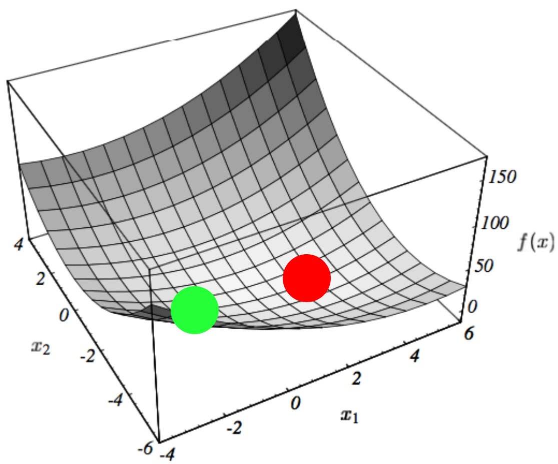


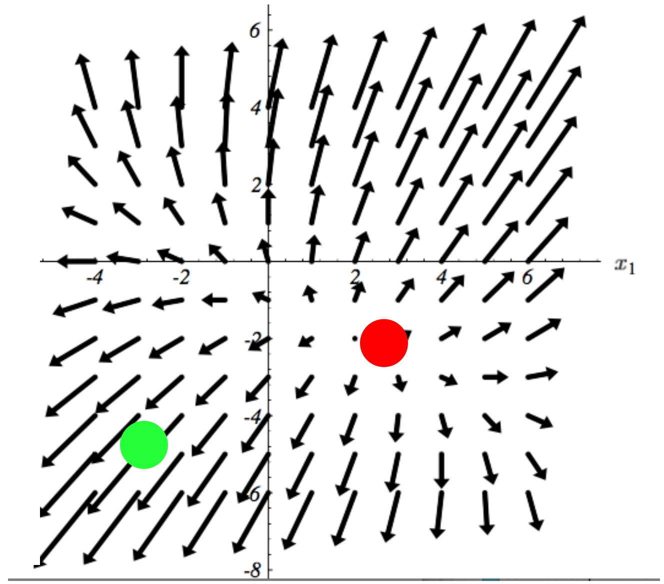
$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Fact: gradient at any point gives direction of fastest increase

Idea: start at a point and move in the direction opposite to the gradient

Gradient-based minimization



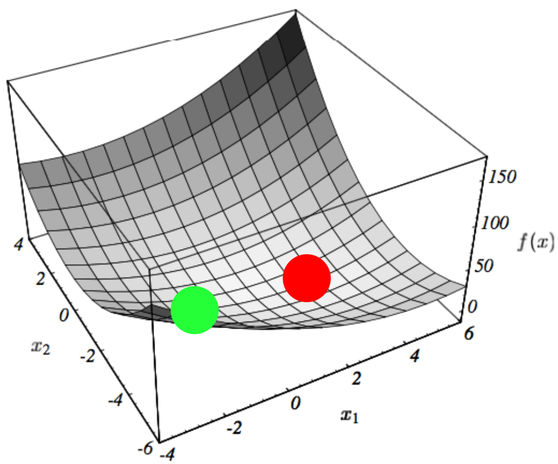


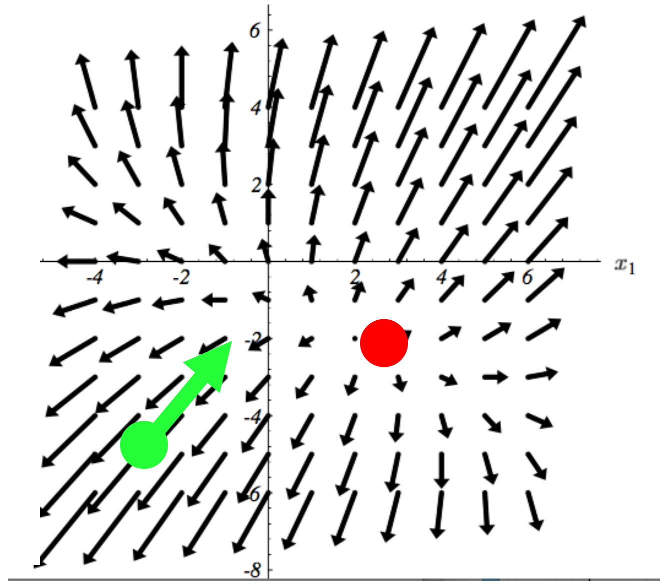
$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Fact: gradient at any point gives direction of fastest increase

Idea: start at a point and move in the direction opposite to the gradient

Gradient-based minimization



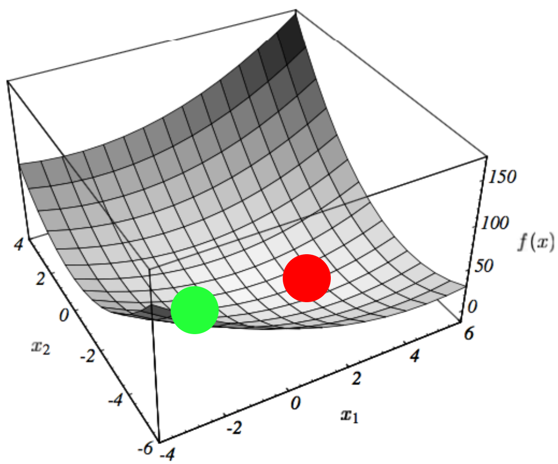


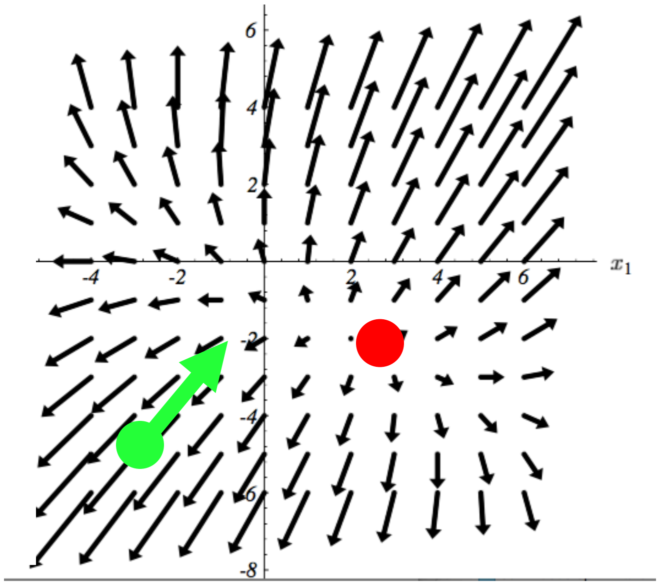
$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Fact: gradient at any point gives direction of fastest increase

Idea: start at a point and move in the direction opposite to the gradient

Gradient-based minimization





$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

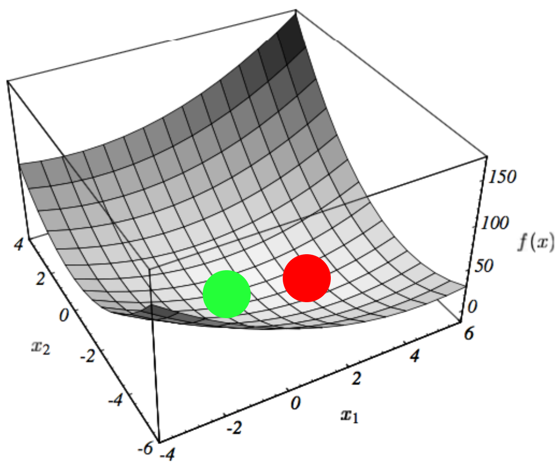
Fact: gradient at any point gives direction of fastest increase

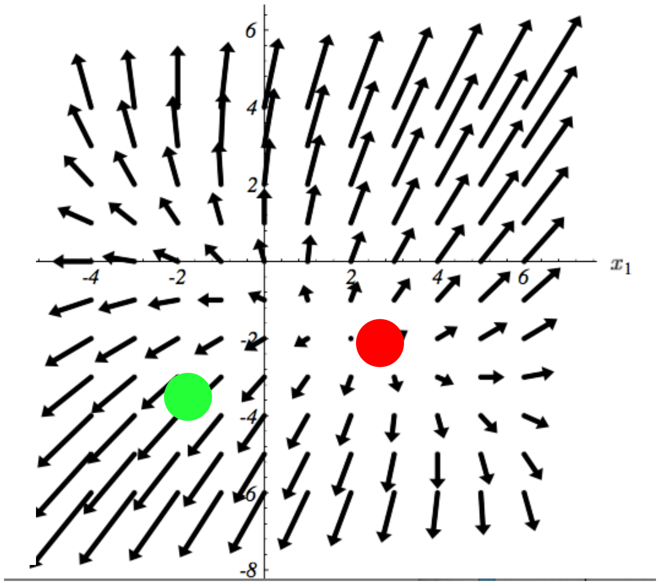
Idea: start at a point and move in the direction opposite to the gradient

Initialize: $\mathbf{x}_0$

Update: $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$ **i=0**

Gradient-based minimization

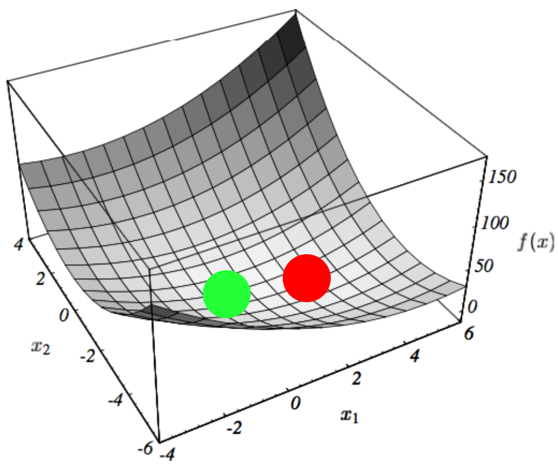


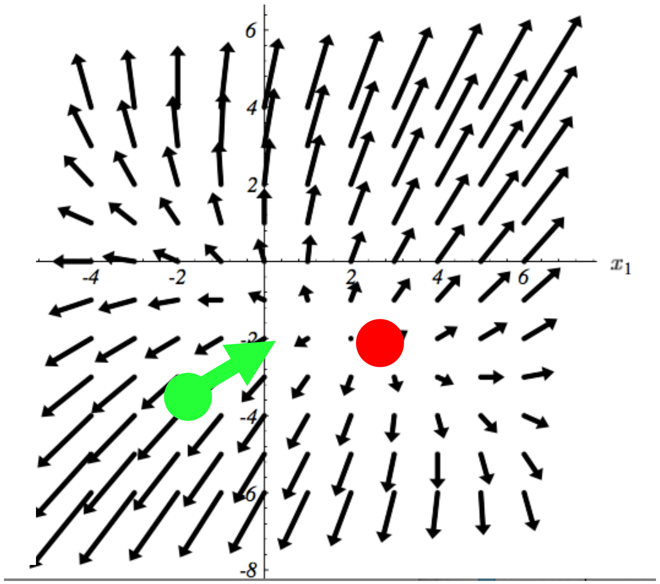


$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Update: $$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$$ **i=1**

Gradient-based minimization

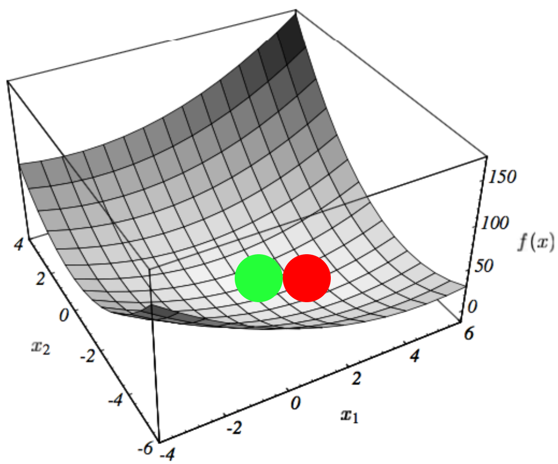


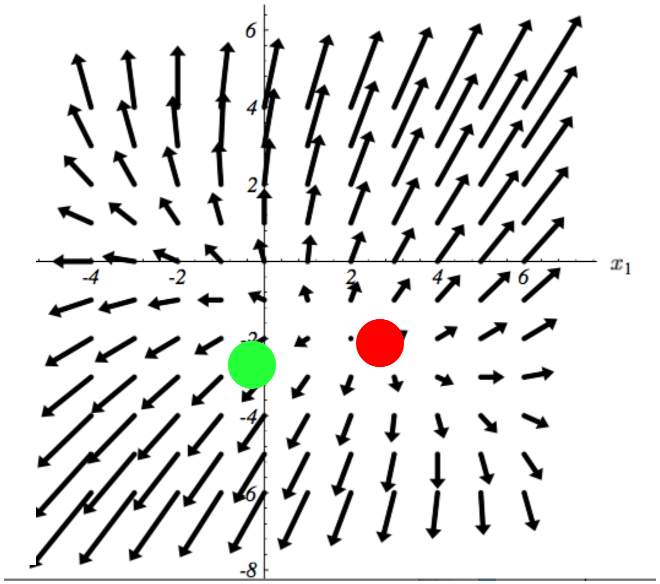


$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Update: $$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$$ **i=1**

Gradient-based minimization



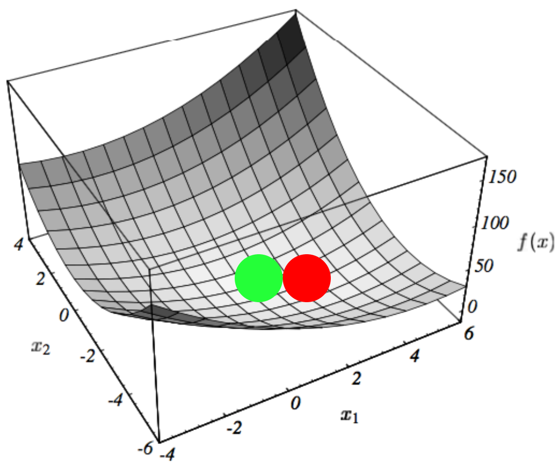


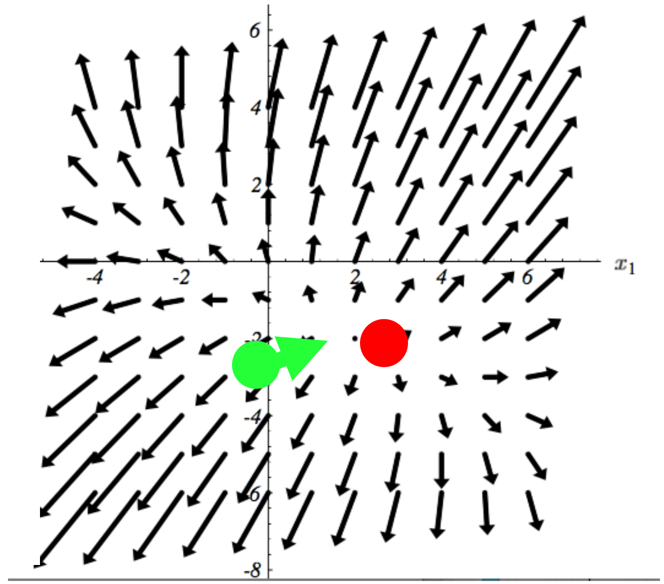
$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Update: $$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$$

i=2

Gradient-based minimization

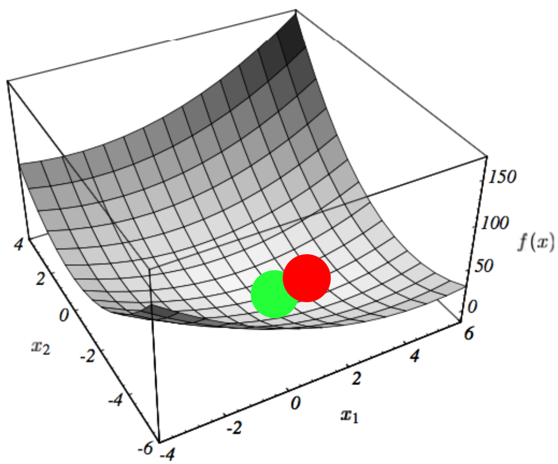


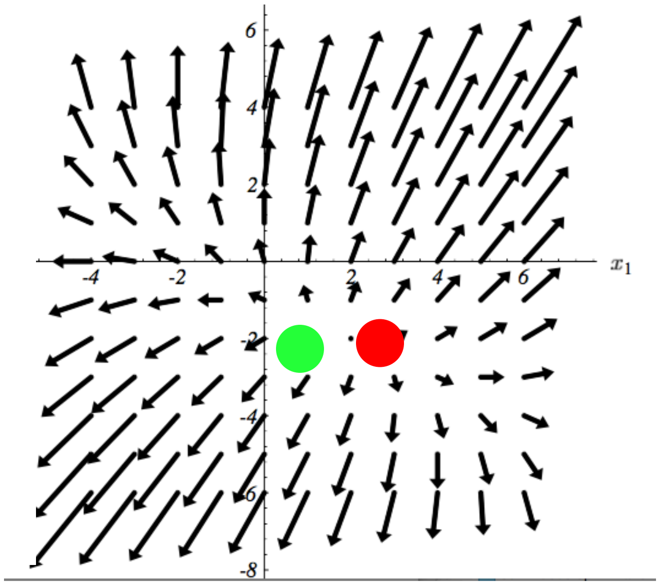


$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Update: $$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$$ **i=2**

Gradient-based minimization

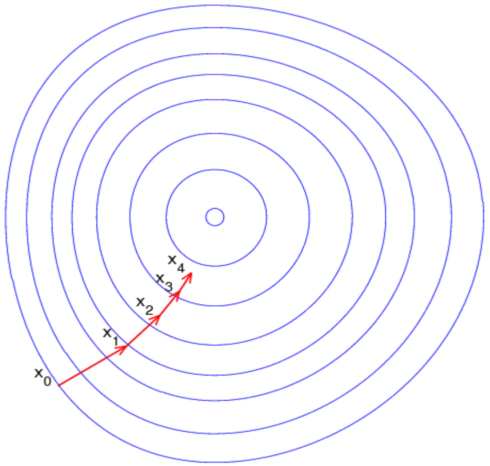




$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Update: $$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$$ **i=3**

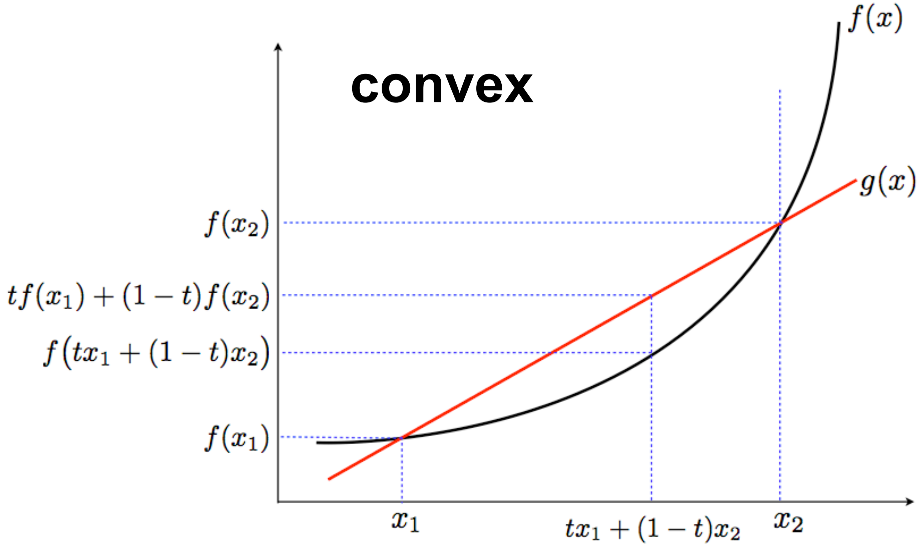
Gradient descent minimization method

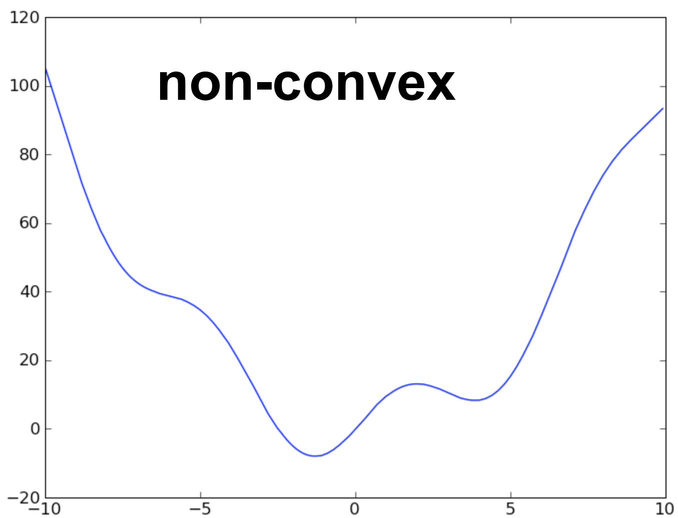


Initialize: $\mathbf{x}_0$

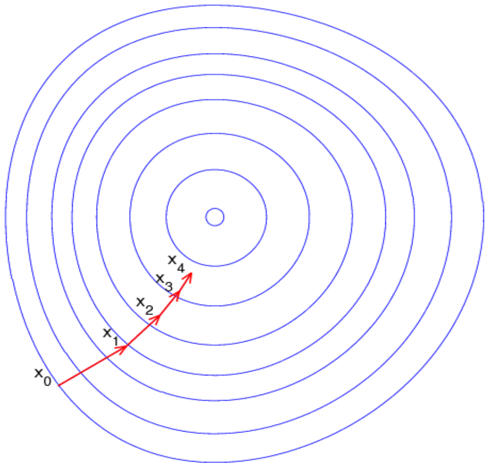
Update: $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$

We can always make it converge for a convex function





Problems of gradient descent



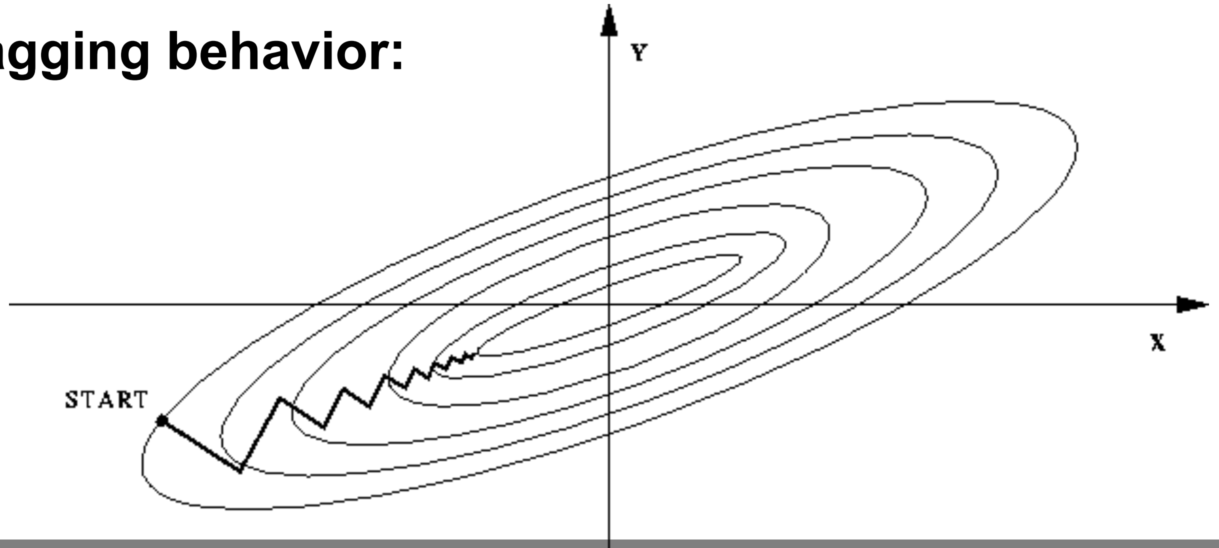
Step-size selection:

Initialize: $\mathbf{x}_0$

Update: $\mathbf{x}_{i+1} = \mathbf{x}_i - \boxed{\alpha} \nabla f(\mathbf{x}_i)$

How to set this?

Zig-zagging behavior:



Thought experiment: least squares

Sum of squared errors minimization:

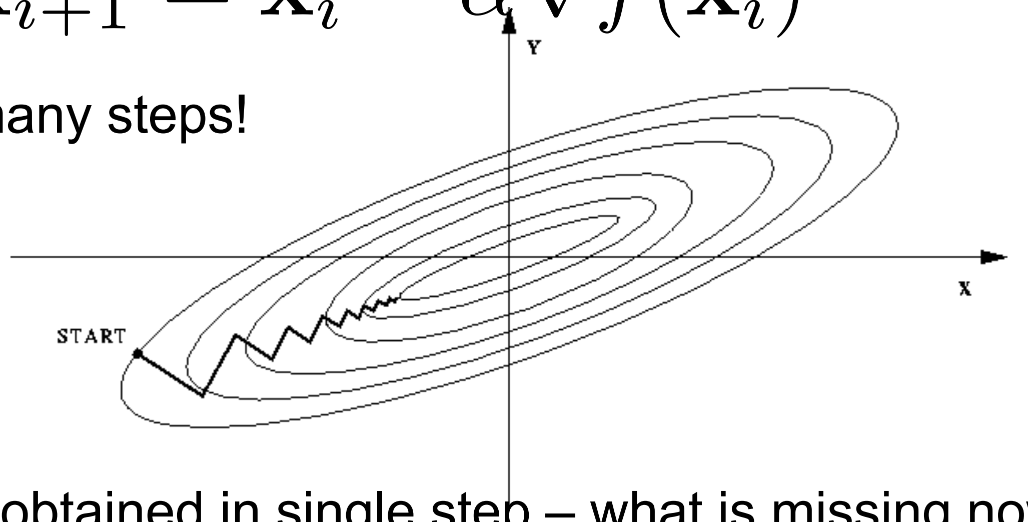
$$f(\mathbf{x}) = (\mathbf{y} - \mathbf{D}\mathbf{x})^T(\mathbf{y} - \mathbf{D}\mathbf{x})$$

Gradient descent:

Initialize: $\mathbf{x}_0$

Update: $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x}_i)$

May require many steps!



We know solution can be obtained in single step – what is missing now?

Least squares solution, in vector form

$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}$$

$$= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w}$$

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

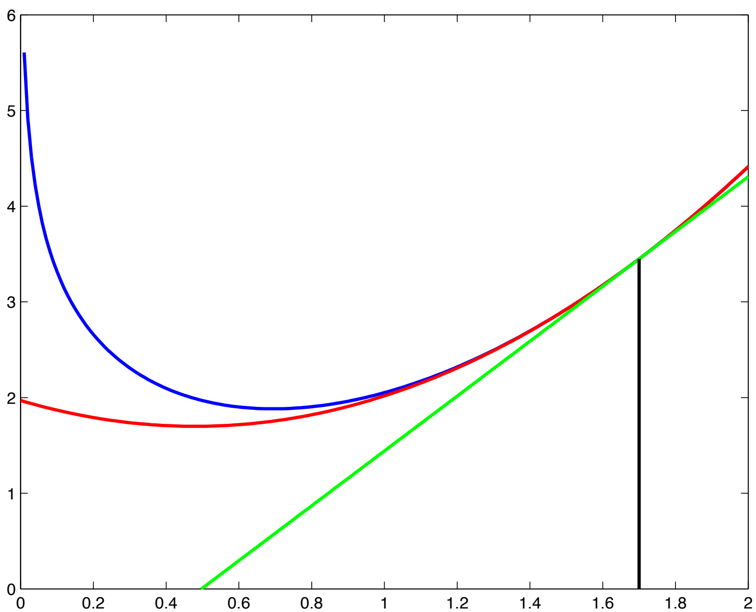
Second-order methods

First- order Taylor series approximation:

$$f(x) \simeq f(a) + (x-a)f'(a) + e(x)$$

Second-order Taylor series approximation:

$$f(x) = f(a) + (x-a)f'(a) + \frac{1}{2}(x-a)^2 f''(a) + e(x)$$



blue:

$$f(x) = x^2 - \log(b) + \exp(\frac{x}{20})$$

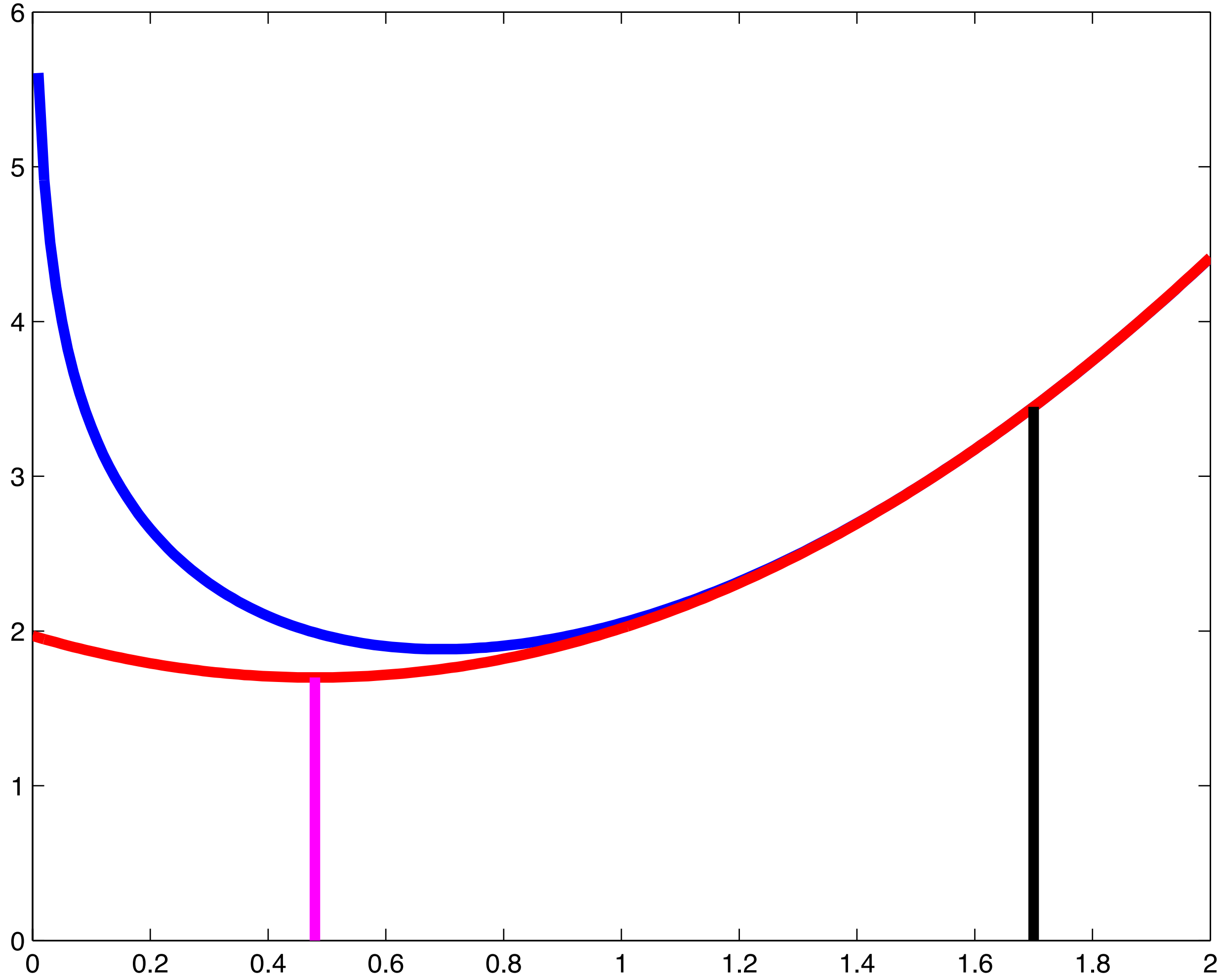
green: linear approximation

$$l(x) = f(a) + (x-a)f'(a)$$

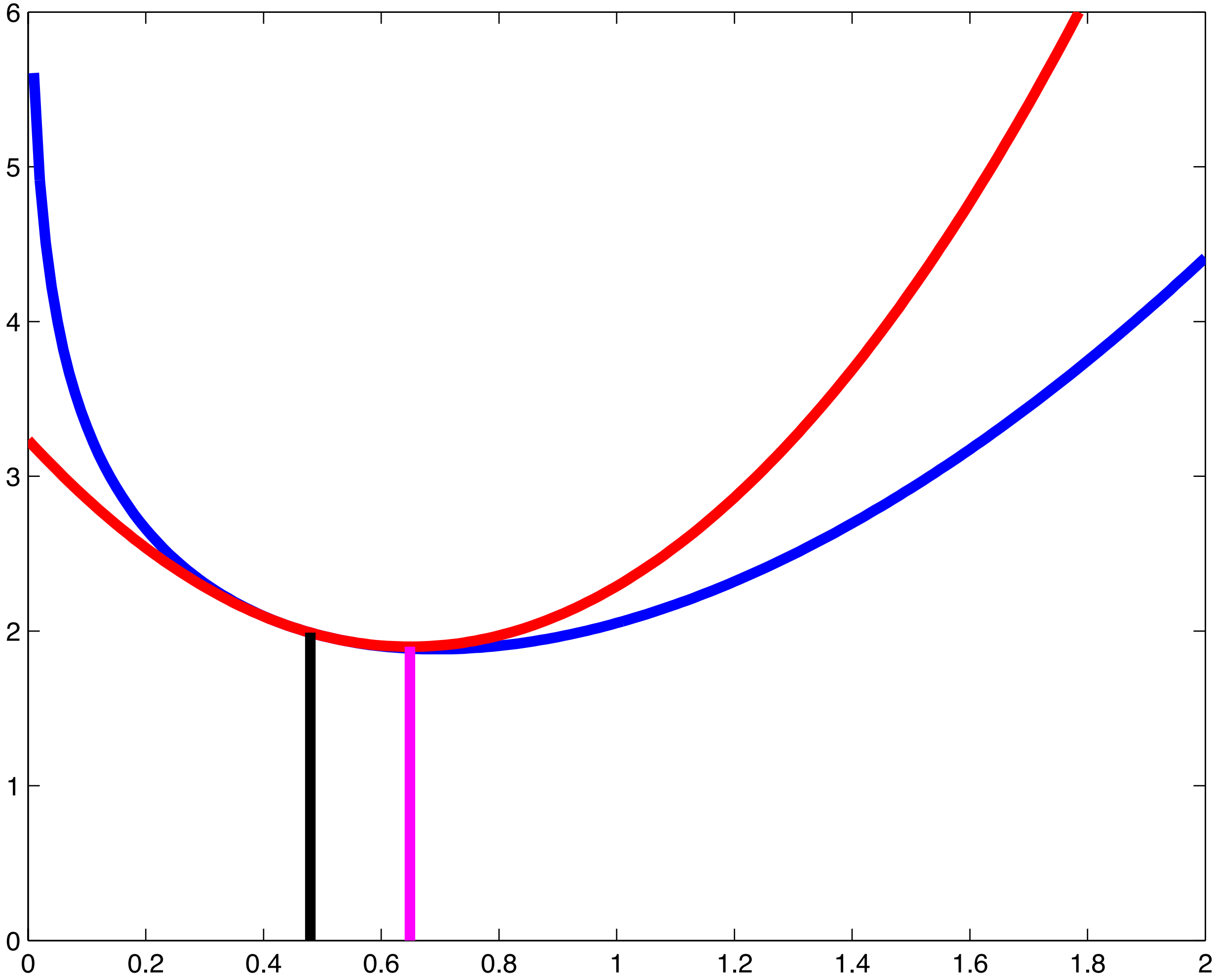
red: quadratic approximation

$$q(x) = f(a) + (x-a)f'(a) + \frac{1}{2}(x-a)^2 f''(a)$$

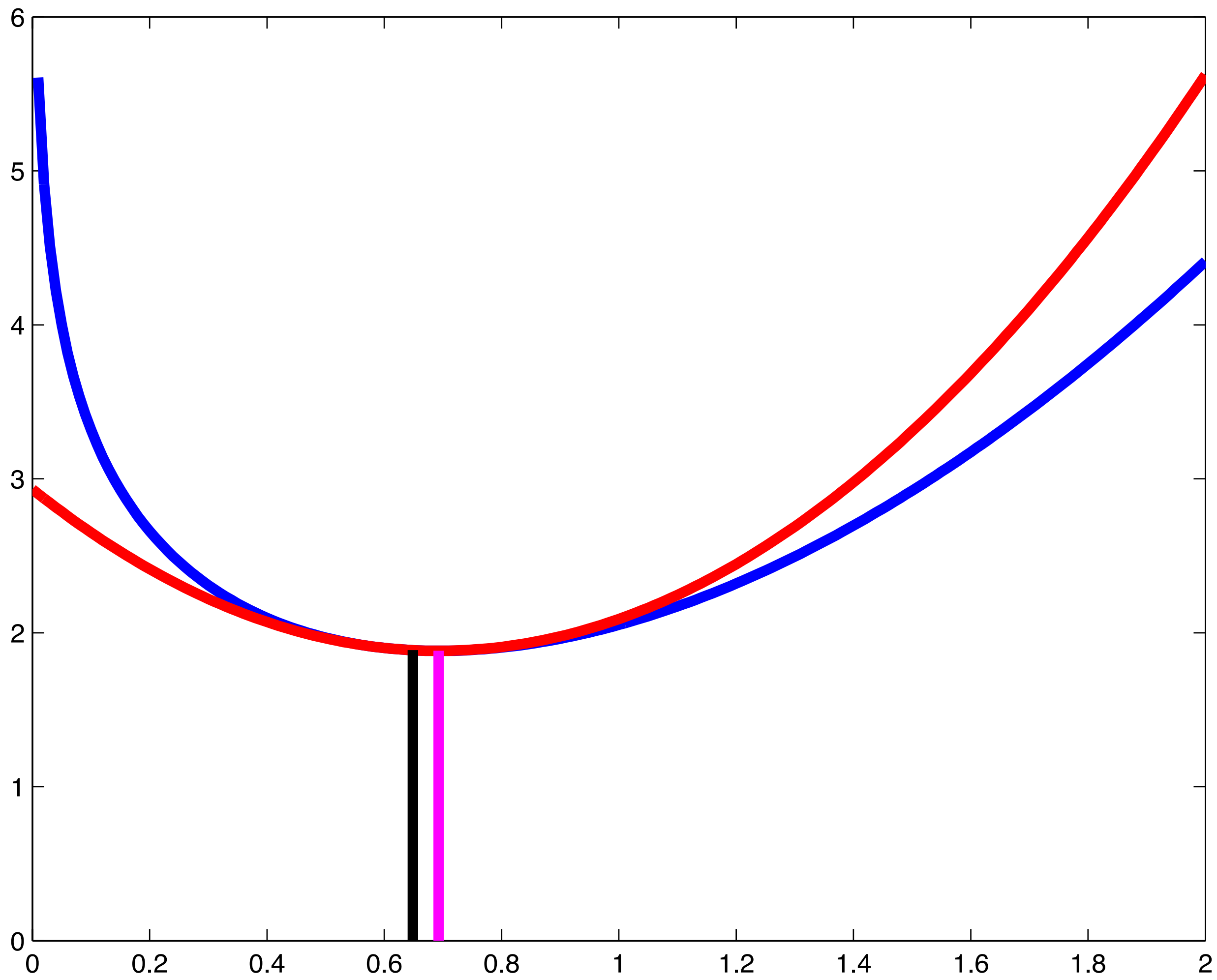
Second-order minimization, 1D



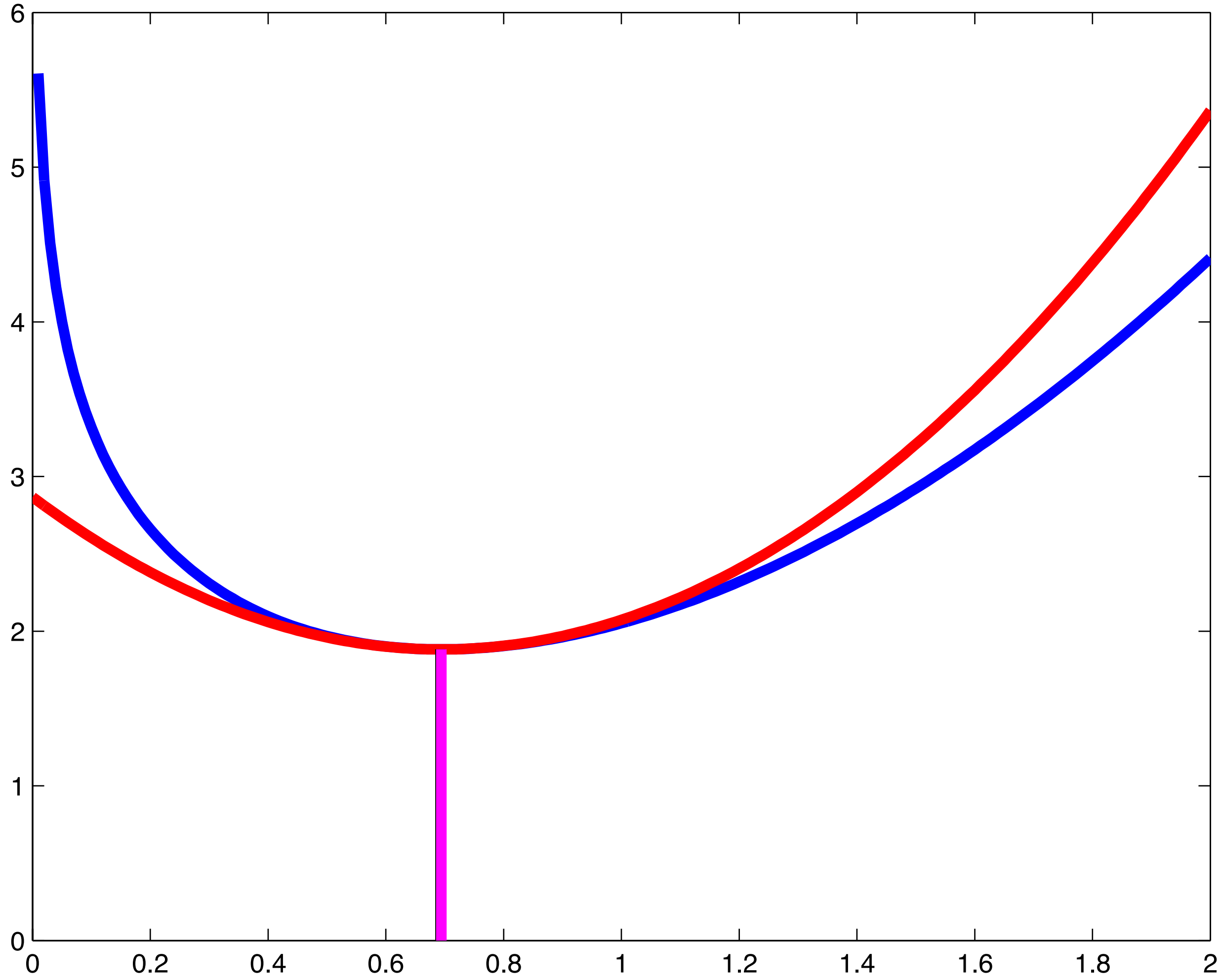
Second-order minimization, 1D



Second-order minimization, 1D



Second-order minimization, 1D



Second-order minimization, 1D

Start from some inititial position, x_0

At any point, form quadratic approximation:

$$f(x) \simeq q(x) = f(x_i) + (x - x_i)f'(x_i) + \frac{1}{2}(x - x_i)^2 f''(x_i)$$

Condition for minimum of quadratic approximation:

$$q'(x) = 0 \rightarrow f'(x_i) + (x - x_i)f''(x_i) = 0$$

Set point in next iteration to be at the minimum of present approximation

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$$

Until update is too small

Note: f" sets the update rate α as the inverse of curvature

Second-order methods, multivariate case

First- order Taylor series approximation:

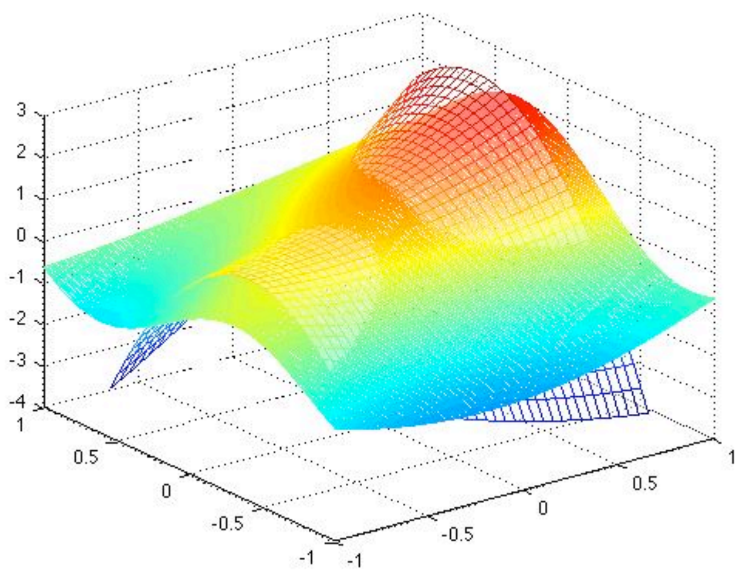
$$f(\mathbf{x}) \simeq f(\mathbf{x}_i) + (\mathbf{x} - \mathbf{x}_i)^T \nabla f(\mathbf{x}_i)$$

Second-order Taylor series approximation:

$$f(\mathbf{x}) \simeq f(\mathbf{x}_i) + (\mathbf{x} - \mathbf{x}_i)^T \nabla f(\mathbf{x}_i) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_i)^T \mathbf{H}(\mathbf{x} - \mathbf{x}_i)$$

$$\doteq q(\mathbf{x})$$

$$H_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$



Second-order minimization, 1D

Start from some inititial position, x_0

At any point, form quadratic approximation:

$$f(x) \simeq q(x) = f(x_i) + (x - x_i)f'(x_i) + \frac{1}{2}(x - x_i)^2 f''(x_i)$$

Condition for minimum of quadratic approximation:

$$q'(x) = 0 \rightarrow f'(x_i) + (x - x_i)f''(x_i) = 0$$

Set point in next iteration to be at the minimum of present approximation

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$$

Until update is too small

Second-order minimization, N-D (Newton-Raphson)

Start from some inititial position, $\mathbf{x}_0$

At any point, form quadratic approximation:

$$f(\mathbf{x}) \simeq q(\mathbf{x}) = f(\mathbf{x}_i) + (\mathbf{x} - \mathbf{x}_i)^T \nabla f(\mathbf{x}_i) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_i)^T \mathbf{H}(\mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i)$$

Condition for minimum of quadratic approximation:

$$\nabla q(\mathbf{x}) = 0 \rightarrow \nabla f(\mathbf{x}_i) + (\mathbf{x} - \mathbf{x}_i)^T \mathbf{H}(\mathbf{x}_i) = 0$$

Set point in next iteration to be at the minimum of present approximation

$$\mathbf{x}_{i+1} = \mathbf{x}_i - (\mathbf{H}(\mathbf{x}_i))^{-1} \nabla f(\mathbf{x}_i)$$

Until update is too small

Newton-Raphson for Logistic Regression

Gradient: $$\frac{\partial L(\mathbf{w})}{\partial w_k} = -\sum_{i=1}^{N}\left[y^i - g(\mathbf{w}^T\mathbf{x}^i)\right]\mathbf{x}_k^i$$

Hessian:

$$\frac{\partial^2 L(\mathbf{w})}{\partial w_k \partial w_j} = \frac{\partial\left(-\sum_{i=1}^{N}\left[y^i - g(\mathbf{w}^T\mathbf{x}^i)\right]\mathbf{x}_k^i\right)}{\partial w_j}$$

$$= \sum_{i=1}^{N}\mathbf{x}_k^i\frac{\partial g(\mathbf{w}^T\mathbf{x}^i)}{\partial w_j} = \sum_{i=1}^{N}\mathbf{x}_k^i g(\mathbf{w}^T\mathbf{x}^i)(1-g(\mathbf{w}^T\mathbf{x}^i))\mathbf{x}_j^i$$

Summation- and matrix-based expressions

$$H_{k,j} = \frac{\partial^2 L(\mathbf{w})}{\partial w_k \partial w_j} = \sum_{i=1}^{N} \mathbf{x}_k^i g(\mathbf{w}^T \mathbf{x}^i)(1 - g(\mathbf{w}^T \mathbf{x}^i)\mathbf{x}_j^i$$

Matrix version of same result:

$$H(\mathbf{w}) = \mathbf{X}^T \mathbf{R} \mathbf{X}, \quad R_{i,i} = g(\mathbf{w}^T \mathbf{x}^i)(1 - g(\mathbf{w}^T \mathbf{x}^i))$$