University of Amsterdam
Informatics Institute

AMLAB Amsterdam Machine Learning Lab

# Machine Learning 1

Lecture 7.4 - Supervised Learning
Classification - Logistic Regression: Newton-Raphson Optimization

*Erik Bekkers*

*(Bishop 4.3.3)*

*Slide credits: Patrick Forré and Rianne van den Berg*

Image credit: Kirillm | Getty Images

# Logistic Regression for Two Classes

- Given: Dataset $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)^T$ with binary targets

  $\mathbf{t} = (t_1, ..., t_N)^T$ with $t_n \in \{\mathcal{C}_1, \mathcal{C}_2\} = \{1, 0\}$

- Conditional likelihood function:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(\mathbf{t}_n|\mathbf{x}_n, \mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}$$

$$y_n = p(C_1|\phi_n) = \sigma(w^T \phi_n) \qquad \boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$$

- Maximizing the conditional likelihood/minimizing the cross-entropy

$$E(\mathbf{w}) = -\ln p(\mathbf{t}, \mathbf{X}, \mathbf{w}) = -\sum_{n=1}^{N} t_n \ln y_n + (1 - t_n)\ln(1 - y_n)$$

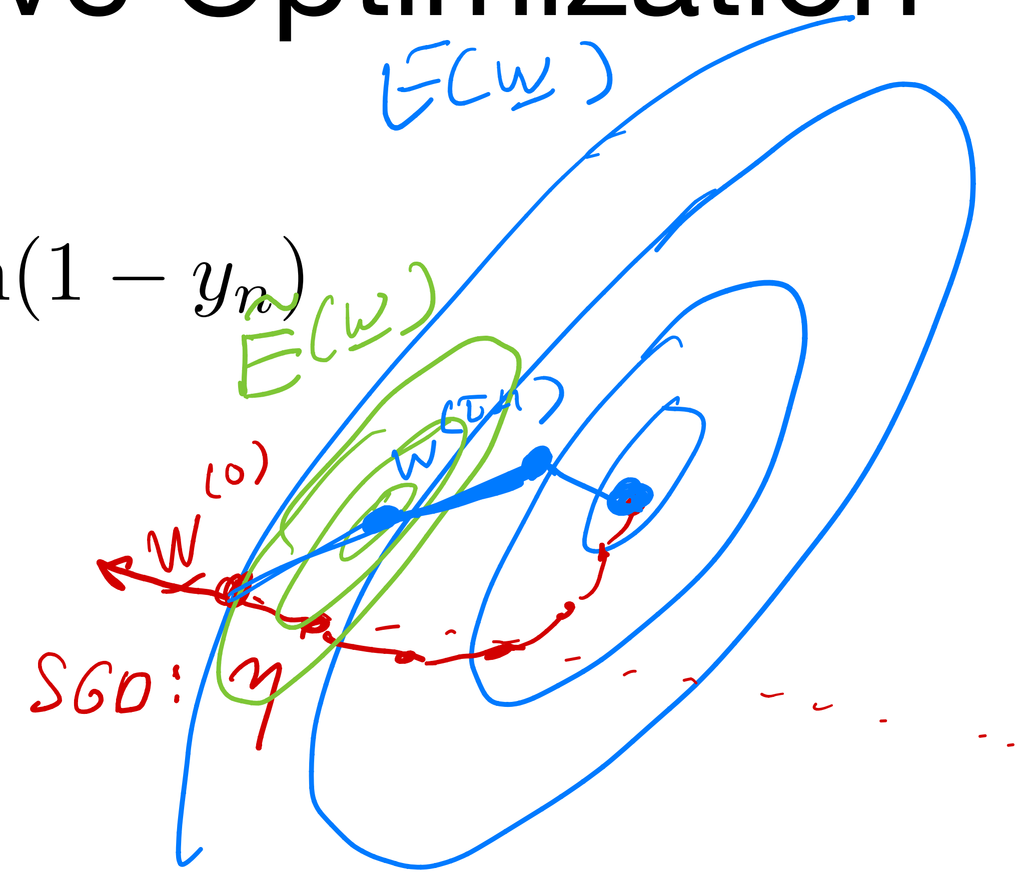
- E(**w**): convex, but no closed form solution!

  $y_n = \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)$ is nonlinear in **w**

# Newton-Raphson Iterative Optimization

- Goal: minimize

$$E(\mathbf{w}) = -\sum_{n=1}^{N} t_n \ln y_n + (1 - t_n)\ln(1 - y_n)$$




- Newton-Raphson iterative optimization scheme:

1. Initial guess $\mathbf{w}^{(0)}$

2. For $\tau$ =1,… :

   I. Approximate E($\mathbf{w}$) with a quadratic function Ẽ($\mathbf{w}$) around $\mathbf{w}^{(\tau-1)}$

   II. Construct $\mathbf{w}^{(\tau)}$ such that it minimizes Ẽ($\mathbf{w}$)

   III. Stop when || $\mathbf{w}^{(\tau-1)}$-$\mathbf{w}^{(\tau)}$ || = 0

3. You have found $\mathbf{w}^{*}$ such that $\frac{\partial}{\partial \mathbf{w}} E(\mathbf{w}^*) = 0$

# Newton-Raphson Iterative Optimization

- Given your old estimate $\mathbf{w}^{(\tau\text{-}1)}$, approximate E(**w**) with a second order Taylor expansion around $\mathbf{w}^{(\tau\text{-}1)}$

$$E(\mathbf{w}) \approx \tilde{E}(\mathbf{w}^{(n-1)} + \Delta\mathbf{w}) = E(\mathbf{w}^{(n-1)}) + (\Delta\mathbf{w})^T \nabla^T E(\mathbf{w}^{(n-1)}) + \frac{1}{2}(\Delta\mathbf{w})^T \mathbf{H} \Delta\mathbf{w}$$

- Gradient $\nabla E(\mathbf{w}) = \left( \frac{\partial E(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial E(\mathbf{w})}{\partial w_{M-1}} \right)$

- Hessian Matrix: $H_{ij} = \frac{\partial^2 E(w)}{\partial w_i \partial w_j}$ is symmetric

- Choose $\Delta\mathbf{w}$ such that next estimate $\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} + \Delta\mathbf{w}$ minimizes Ẽ(**w**)

$$\frac{\partial}{\partial \Delta\mathbf{w}} \tilde{E}(\mathbf{w}^{(\tau-1)} + \Delta\mathbf{w}) = \nabla E + (\Delta w)^T H = 0 \rightarrow H \Delta w = -\nabla^T E$$

$$\rightarrow \Delta w = -H^{-1} \nabla^T E$$

- Update rule: $\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \mathbf{H}^{-1} \nabla^T E(\mathbf{w}^{(\tau-1)})$

# Newton-Raphson Iterative Optimization

- Update rule: $$\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \mathbf{H}^{-1}\nabla E(\mathbf{w}^{(\tau-1)})$$ $\left( E(\underline{w}) = \sum_{n=1}^{N} E_n(\underline{w}) \right)$

- Gradient $$\nabla E_n(\mathbf{w})^T = \left(\frac{\partial E_n(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial E_n(\mathbf{w})}{\partial w_{M-1}}\right)^T = (y_n - t_n)\boldsymbol{\phi}_n$$

$\Phi \in \mathbb{R}^{N \times M}$

$$\nabla E(\underline{w})^T = \sum_{n=1}^{N} (y_n - t_n)\,\phi_n = \Phi^T (\underline{y} - \underline{t}) \in \mathbb{R}^M$$

- Hessian

$y_n = \sigma(\underline{w}^T \phi_n)$

$$\mathbf{H}_{ij} = \frac{\partial E(\mathbf{w}^{(\tau-1)})}{\partial w_i \partial w_j} = \frac{\partial}{\partial w_i} \overbrace{\sum_{n=1}^{N}(y_n - t_n)\phi_j(\mathbf{x}_n)}^{\frac{\partial}{\partial w_j} E(\underline{w})} = \sum_{n=1}^{N} \phi_j(x_n) \underbrace{\frac{\partial y_n}{\partial w_i}}_{y_n(1-y_n)\phi_i(x_n)}$$

$$= \sum_{n=1}^{N} y_n(1-y_n)\,\phi_i\,\phi_j$$

$\in \mathbb{R}^{M \times M}$

$$\mathbf{H} = \sum_{n=1}^{N} y_n(1-y_n)\,\phi_n \phi_n^T = \Phi^T R \Phi$$

$R_{nn} = y_n(1-y_n)$

$R_{nm} = 0$ if $n \neq m$

# The cross entropy loss is convex

- The error function $E(\mathbf{w})$ is convex when its Hessian is positive definite, meaning $\forall_{\mathbf{w}\neq\mathbf{0}\in\mathbb{R}^M} : \mathbf{w}^T\mathbf{H}\mathbf{w} > 0$

$\mathbb{R}^{N\times N}$ $\mathbb{R}^{N\times M}$

- The Hessian of $E(\mathbf{w})$ is given by $\mathbf{H} = \mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}$ , with $\mathbf{R} = \mathrm{diag}_{N\times N}\{y_n(1-y_n)\}$

$0 < y_n < 1$

$R = R^{\frac{1}{2}} R^{\frac{1}{2}}$

$R^{\frac{1}{2}} = \mathrm{diag}\left(\sqrt{y_n(1-y_n)}\right)$

$$\underline{w}^T H \underline{w} = \underline{w}^T \Phi^T R \Phi \underline{w}$$

$$= \underbrace{\underline{w}^T \Phi^T R^{\frac{1}{2}}} \underbrace{R^{\frac{1}{2}} \Phi \underline{w}}$$

$$= \left(R^{\frac{1}{2}} \Phi \underline{w}\right)^T \left(R^{\frac{1}{2}} \Phi \underline{w}\right)$$

$$= \| R^{\frac{1}{2}} \Phi \underline{w} \|^2 > 0$$

# Iterative Reweighted Least Squares

- Update rule: $\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \mathbf{H}^{-1}\nabla E(\mathbf{w}^{(\tau-1)})$

- Gradient: $\nabla E(\mathbf{w}) = \mathbf{\Phi}^T(\mathbf{y} - \mathbf{t})$

- Hessian: $\mathbf{H} = \mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}$ with $R_{nn} = y_n(1 - y_n)$

- Newton-Raphson update rule for two-class logistic regression:

$$\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T(\mathbf{y} - \mathbf{t})$$

$$= (\Phi^T R \Phi)^{-1}\{\Phi^T R \Phi w^{(\tau-1)} - \Phi^T(y - t)\}$$

$$= (\Phi^T R \Phi)^{-1}\{\Phi^T R(z + R^{-1}(y-t)) - \Phi^T(y-t)\}$$

$$= (\Phi^T R \Phi)^{-1}\Phi^T R z$$

$$\mathbf{z} = \mathbf{\Phi}\mathbf{w}^{(\tau-1)} - \mathbf{R}^{-1}(\mathbf{y} - \mathbf{t})$$

- Note similarity with ML solution to linear regression:

$$\mathbf{w}_{\mathrm{ML}} = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{t}$$

weights target

solution to

$$\underset{w}{\operatorname{argmin}} \sum r_i\left(w^T\phi_n - z_n\right)^2$$

# SGD vs Newton-Raphson

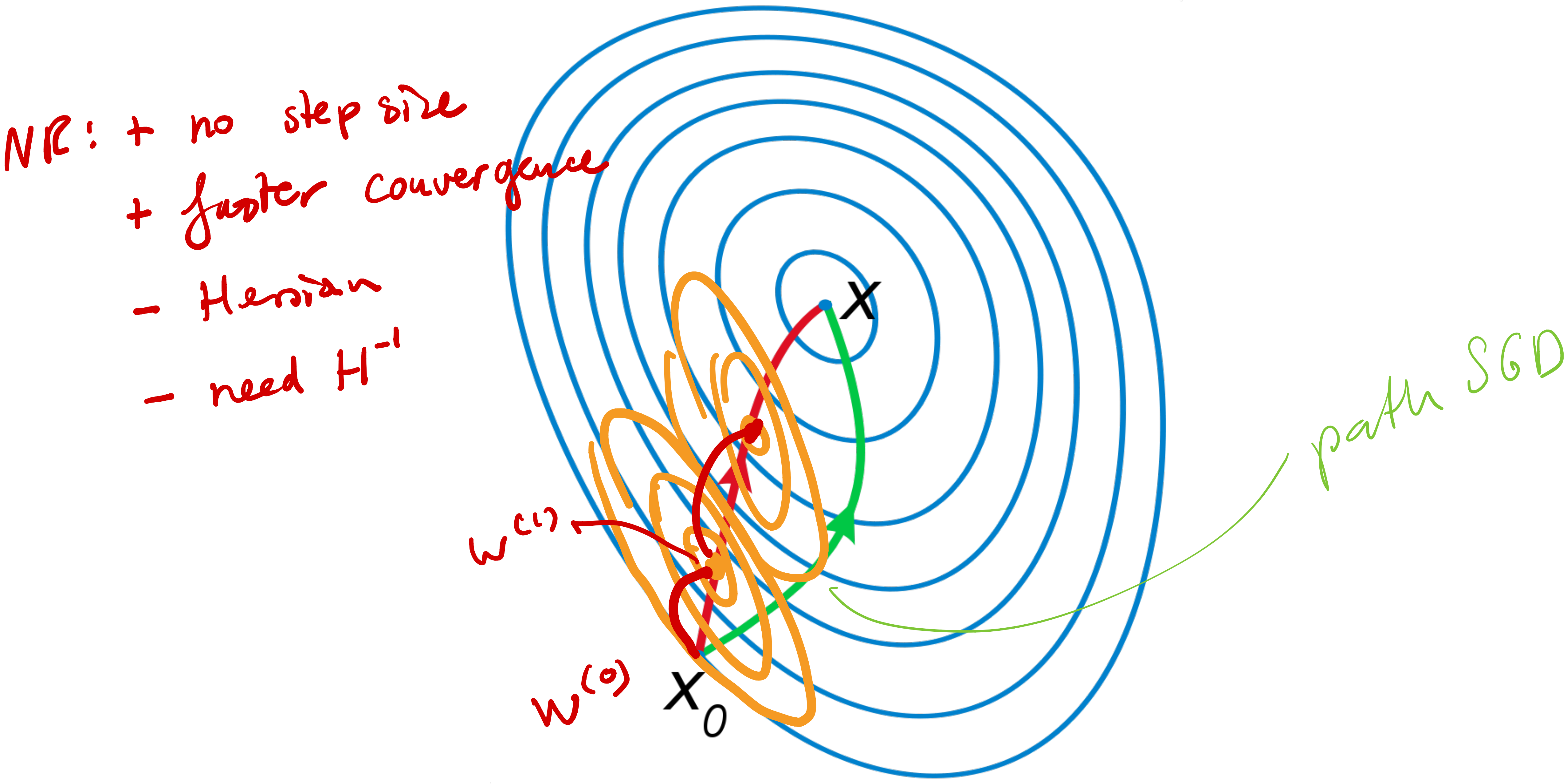



**Figure:** Green: gradient descent, which always goes in the direction of steepest descent. Red: Newton-Raphson's procedure, which takes into account curvature to take a more direct path. (Wikipedia)