TC2: Optimization for Machine Learning

Master of Science in AI and Master of Science in Data Science

@ UPSaclay

2025/2026.

Faïcel Chamroukhi

université PARIS-SACLAY

SystemX INSTITUT DE RECHERCHE TECHNOLOGIQUE

chamroukhi.com

week 4 : November 27, 2025.

- **Mid-term exam**

- **Gradient descent acceleration methods,**
- **Second order methods**

  (Newton methods including Quasi-Newton, secant, IRLS)

Classical Momentum Method

université PARIS-SACLAY

- Momentum is used to accelerate convergence by adding an inertia effect, keeping the model from being too influenced by noisy gradients.
- emphasizes the directions that persistently reduce f across iterations :

$$v_{k+1} = \mu v_k - \alpha \nabla f(x_k) \tag{1}$$

$$x_{k+1} = x_k + v_{k+1} \tag{2}$$

  where :
  - v_k : *velocity* vector at iteration k : acts as a memory that accumulates the directions of reduction that were chosen in the previous k steps
  - μ : *momentum coefficient* $\in [0, 1]$: controlling the influence of memory
- Notice that if $\mu = 0$ we recover GD.
- The velocity helps accumulate gradients from previous iterations, leading to faster convergence especially in regions of smooth descent.
- By memorizing, through the velocity vector, the direction where the gradient has been consistent over the iterations, CM helps. This is also the direction where the gradient changes slowly or, equivalently, where the curvature is low

Classical Momentum (continued)

université PARIS-SACLAY

Convergence rate of CM :

- Typically, for convex and smooth functions (not necessarily strongly convex), Classical Momentum (CM) can achieve a convergence rate of :

$$O\left(\frac{1}{k}\right)$$

- This is the same rate as basic Gradient Descent (GD) for general convex functions.
- The difference is that CM can reach this rate more stably and smoothly by leveraging the momentum term *to avoid oscillations* and accelerate through flat regions.
- CM helps to "accumulate" the velocity vector from previous gradients, making the overall movement smoother. This can help in some practical scenarios, but theoretically, it doesn't necessarily improve the convergence rate beyond $O(1/k)$.

Nesterov's Accelerated Gradient (NAG)

université PARIS-SACLAY

- Nesterov's Accelerated Gradient (NAG), introduced in 1983.
- The update equations of NAG are :

$$v_{k+1} = \mu v_k - \alpha \nabla f(x_k + \mu v_k) \quad (1^*)$$

$$x_{k+1} = x_k + v_{k+1} \quad (2^*)$$

- Equations (1), (2) are identical to equations (1*), (1*) except for a single, seemingly benign, difference :
  - While the classical momentum (CM) method updates the velocity vector by inspecting the gradient at the current iterate x_k,
  - NAG updates it by inspecting the gradient at $x_k + \mu v_k$.
- NAG uses a momentum-like approach to "look ahead" in the direction suggested by the velocity vector.

Nesterov's Accelerated Gradient (continued)

- To make an analogy :
  - While CM faithfully trusts the gradient at the current iterate, NAG puts less faith into it and looks ahead in the direction suggested by the velocity vector.
  - It then moves in the direction of the gradient at the look-ahead point.
- If $\nabla f(x_k + \mu v_k) \approx \nabla f(x_k)$, then the two updates are similar.
- **But**, if not, this is an indication of curvature and the NAG update has, correctly, put less faith in the gradient.
- This small difference, compounded over the iterations, gives the two methods distinct properties, allowing NAG to adapt faster and in a more stable way than CM in many settings, particularly for higher values of μ.
- As a result, NAG also enjoys provably faster convergence in certain settings.
- Concretely, for any convex, smooth function, i.e., not necessarily strongly convex, optimally tuned NAG converges at an $O(1/k^2)$ rate, while GD converges at a rate of $O(1/k)$.

Convergence Rate of NAG vs GD :

- **Gradient Descent (GD)** : For convex and smooth functions, GD has a convergence rate of $O(1/k)$.
  - This means that the error (in terms of how close you are to the optimal value) decreases in proportion to $1/k$ as the number of iterations k increases.
- **Nesterov's Accelerated Gradient (NAG)** : NAG is an accelerated method that, when optimally tuned, has a faster convergence rate of $O(1/k^2)$ for convex and smooth functions.
  - This rate is significantly better than that of GD, implying that NAG reduces the error much faster as the number of iterations increases.
  - The absence of strong convexity implies that we are considering general convex functions, which do not have a strict lower bound on their curvature.

Continuous optimization

(2nd order methods : Newton methods, Quasi-Newton, secant, IRLS)

Newton's method I

université PARIS-SACLAY

Origin of its construction :
The Newton method is widely known as a technique for finding a root of a function of one variable. Let $f(x) : \mathbb{R} \to \mathbb{R}$. Consider the equation :

$$f(x) = 0.$$

The Newton method is based on linear approximation.
Taylor Approximation for $f : \mathbb{R} \to \mathbb{R}$: If f is differentiable, then for any $x, x_0 \in \mathbb{R}$, Provided that $\|x - x_0\|$ is small (i.e., x is close to x_0), by Taylor's theorem we have

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + R_2(x)$$

where $R_2(x) = o(\|x - x_0\|^2)$ is the remainder which vanishes as $x \to x_0$.
Assume that we get some x_0 close enough to x.
Therefore, the equation $f(x) = 0$ can be approximated by the linear equation :

$$f(x_0) + f'(x_0)(x - x_0) = 0.$$

Newton's method II

We then have :

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Converting this idea in an algorithmic form, we get the process :

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

In optimization, we are finding the roots of $f'(x)$, i.e. solving

$$f'(x) = 0.$$

Hence, the Newton method for optimization problems appears to be in the form

$$x_{k+1} = x_k - [f''(x_k)]^{-1} f'(x_k).$$

assuming f is twice-differentiable.

Newton's Method

université PARIS-SACLAY

Construction as a quadratic approximation

We can obtain the previous process using the idea of quadratic approximation.
Consider the quadratic approximation of $f(x)$, centered at the point x_k :

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2 + o(\|x - x_k\|^2)$$

Assume that $f''(x_k) > 0$. Then we choose x_{k+1} as the point that **minimizes the quadratic approximation** $f(x)$.
This means we solve

$$f'(x) = \frac{d}{dx}\{f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2\} = 0$$

which gives

$$f'(x) = f'(x_k) + f''(x_k)(x - x_k) = 0$$

Solving this in x for x_{x+1} gives us the Newton process

$$x_{k+1} = x_k - [f''(x_k)]^{-1} f'(x_k)$$

in $\mathbb{R}^n$ (Construction as a quadratic approximation) I

Taylor : If f is continuously twice differentiable, then for any $x, x_0 \in \mathbb{R}^n$, provided that $\|x - x_0\|$ is small (i.e., x is close to x_0), we can approximate $f(x)$ by :
(second-order approximation).

$$f(x) = f(x_0) + \nabla f(x_0)^T (x - x_0) + \frac{1}{2}(x - x_0)^T \nabla^2 f(x_0)(x - x_0) + o\|x - x_0\|^3$$

To derive the Newton update form from the quadratic approximation, consider again the quadratic approximation of $f(x)$ at the current point x_k :

$$f(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k) + o\|x - x_k\|^3$$

The quadratic approximation, $f(x)$, captures the behavior of $f(x)$ locally around x_k :
The quadratic term incorporates the Hessian, which captures the curvature of f.

Assume that $\nabla^2 f(x_k) \succ 0$. Then we can choose x_{k+1} as a point of minimum of the quadratic function $f(x)$.
Minimizing $f(x)$ involves taking its gradient w.r.t x and setting it to zero :

$$\nabla f(x) = \nabla f(x_k) + \nabla^2 f(x_k)(x - x_k) = 0$$

This leads to :

$$\nabla^2 f(x_k)(x_{k+1} - x_k) = -\nabla f(x_k)$$

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$$

Step Size Adjustment :
- The term $-[\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$ uses both gradient and curvature information to determine the optimal direction and step size.
- This approach avoids issues in gradient descent, such as slow convergence in areas of different curvatures.

- This iterative process is applied until convergence.
- Each iteration involves recalculating the quadratic approximation at the new point x_{k+1}, then minimizing this approximation to determine the next point :

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$$

- The gradient, as usual, tells us the direction of steepest ascent (the anti-gradient determines the steepest descent)
- The Hessian allows the method to **adapt the step size** depending on the curvature.
- In regions of **high curvature**, the inverse Hessian shrinks the step size, preventing overshooting.
- In **flat regions**, it permits larger steps, speeding up convergence.
- The Newton method can converge faster than standard gradient descent, especially near well-behaved minima, due to its use of second-order information.

Quasi-Newton Methods

Efficient second-order approximation without explicitly computing the Hessian.

Secant Method

Approximates derivatives using prior information, avoiding full differentiation.

Issue : Lack of Positive Definiteness

The matrix $\nabla^2 f(x)$ may not be positive definite. As a result :

- The problem might lack solutions.
- The descent direction $-[\nabla^2 f(x)]^{-1}\nabla f(x)$ may not always be effective.

Solution : Add Diagonal Matrix $\mathbf{E}$

To resolve this, we add a diagonal matrix $\mathbf{E}$ such that $\nabla^2 f(x) + \mathbf{E}$ becomes positive definite.

- Example : Let $\mathbf{E} = \gamma I^n$, with $-\gamma$ chosen to be smaller than all non-positive eigenvalues of $\nabla^2 f(x)$.
- This modification shifts the original eigenvalues by $\gamma > 0$.

The required value of γ is found when solving the "Newton equation" :

$$\nabla^2 f(x)\mathbf{p} = -\nabla f(x)$$

This approach is known as the *Levenberg-Marquardt* modification.

Note : As γ becomes larger, $\mathbf{p}$ increasingly resembles the steepest descent direction.

Issue : Lack of enough Differentiability The function f may not belong to C^2, or computing the second derivatives ($\nabla^2 f$) might be too costly.

Solution : Use Quasi-Newton Methods

In quasi-Newton methods, we approximate the Newton equation by replacing the Hessian $\nabla^2 f(x_k)$ with a more computationally efficient matrix $\mathbf{B}_k$.

- $\mathbf{B}_k$ is computed using gradient values at the current and previous points.
- Using a first-order Taylor expansion for $\nabla f(x_k)$:

$$\nabla f(x_k) \approx \nabla f(x_{k-1}) + \nabla^2 f(x_k)(x_k - x_{k-1})$$

then

$$\nabla^2 f(x_k)(x_k - x_{k-1}) \approx \nabla f(x_k) - \nabla f(x_{k-1})$$

Updating the Approximation

The matrix $\mathbf{B}_k$ is chosen to satisfy the following system :

$$\mathbf{B}_k(x_k - x_{k-1}) = \nabla f(x_k) - \nabla f(x_{k-1})$$

Secant Method : Special Case for $n = 1$

For $n = 1$, this process reduces to the *secant method*, which approximates the second derivative as :

$$f''(x_k) \approx \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$$

Matrix $\mathbf{B}_k$ Properties

The matrix $\mathbf{B}_k$ has n^2 elements, which makes it *under-determined* by the n equations available. Additional requirements, such as making sure that $\mathbf{B}_k$ is *symmetric* and *positive definite*, result in specific quasi-Newton methods.

Initialization and Update of $\mathbf{B}_k$. Typically :

- Start with $\mathbf{B}_0 = I^n$.
- Update $\mathbf{B}_k$ to $\mathbf{B}_{k+1}$ using a rank-one or rank-two matrix update .
- allows efficient updating of the factorization of $\mathbf{B}_k$, utilizing linear algebra

BFGS Update Rule : There are many ways to update $\mathbf{B}_k$, but the **Broyden-Fletcher-Goldfarb-Shanno** method is among the most effective :

$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{(\mathbf{B}_k \mathbf{s}_k)(\mathbf{B}_k \mathbf{s}_k)^\top}{\mathbf{s}_k^\top \mathbf{B}_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^\top}{\mathbf{y}_k^\top \mathbf{s}_k}$ where :

- $\mathbf{s}_k = x_{k+1} - x_k$
- $\mathbf{y}_k = \nabla f(x_{k+1}) - \nabla f(x_k)$

Remark If f is quadratic, $\mathbf{B}_k$ will converge to the Hessian after n steps.

IRLS

Iteratively Reweighted Least Squares

Newto descent for Logistic Regression

Optimizing the Logistic Regression function

- **Problem :** Given $(x_i, y_i) \in \mathbb{R}^p \times \{0, 1\}$, $i = 1, \ldots, n$, Let $p_i(\theta) = \frac{\exp(x_i^T \theta)}{1+\exp(x_i^T \theta)}$ be the logistic function.

  In logistic regression we minimize the negative log-likelihood : $\min_\theta f(\theta)$

$$f(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left[-y_i x_i^T \theta + \log\left(1 + \exp(x_i^T \theta)\right) \right].$$

- Newton algorithm : $\theta^{(k+1)} = \theta^{(k)} - \left[\nabla^2 f(\theta^{(k)})\right]^{-1} \nabla f(\theta^{(k)})$

  Gradient vector and Hessian Matrix : (details in the next slide)

- Gradient vector $\nabla f(\theta) = \frac{1}{n} \frac{\partial L(\theta)}{\partial \theta} = -\sum_{i=1}^{n} x_i (y_i - p_i(\theta))$.
- Hessian Matrix $\nabla^2 f(\theta) = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\top p_i(\theta)(1 - p_i(\theta))$
- The Newton iterative update of θ has therefore the following expression :

$$\theta^{(k+1)} = \theta^{(k)} + \left[\sum_{i=1}^{n} x_i x_i^\top p_i(x_i; \theta^{(k)})(1 - p_i(x_i; \theta^{(k)}))\right]^{-1} \sum_{i=1}^{n} x_i (y_i - p_i(x_i; \theta^{(k)}))$$

Optimizing the Logistic Regression function

université PARIS-SACLAY

- **Problem :** Given $(x_i, y_i) \in \mathbb{R}^p \times \{0, 1\}$, $i = 1, \ldots, n$, Let $p_i(\theta) = \frac{\exp(x_i^T \theta)}{1+\exp(x_i^T \theta)}$ be the logistic function.

  In logistic regression we minimize the negative log-likelihood : $\min_\theta f(\theta)$

$$f(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left[-y_i x_i^T \theta + \log\left(1 + \exp(x_i^T \theta)\right) \right].$$

- Newton algorithm : $\theta^{(k+1)} = \theta^{(k)} - \left[\nabla^2 f(\theta^{(k)})\right]^{-1} \nabla f(\theta^{(k)})$

  Gradient vector and Hessian Matrix : (details in the next slide)

- Gradient vector $\nabla f(\theta) = \frac{1}{n} \frac{\partial L(\theta)}{\partial \theta} = -\sum_{i=1}^{n} x_i (y_i - p_i(\theta))$.
- Hessian Matrix $\nabla^2 f(\theta) = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\top p_i(\theta)(1 - p_i(\theta))$
- The Newton iterative update of θ has therefore the following expression :

$$\theta^{(k+1)} = \theta^{(k)} + \left[\sum_{i=1}^{n} x_i x_i^\top p_i(x_i; \theta^{(k)})(1 - p_i(x_i; \theta^{(k)}))\right]^{-1} \sum_{i=1}^{n} x_i (y_i - p_i(x_i; \theta^{(k)}))$$

Optimizing the Logistic Regression function

université PARIS-SACLAY

- **Problem :** Given $(x_i, y_i) \in \mathbb{R}^p \times \{0, 1\}$, $i = 1, \ldots, n$, Let $p_i(\theta) = \frac{\exp(x_i^T \theta)}{1+\exp(x_i^T \theta)}$ be the logistic function.

  In logistic regression we minimize the negative log-likelihood : $\min_\theta f(\theta)$

$$f(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left[-y_i x_i^T \theta + \log\left(1 + \exp(x_i^T \theta)\right) \right].$$

- Newton algorithm : $\theta^{(k+1)} = \theta^{(k)} - \left[\nabla^2 f(\theta^{(k)})\right]^{-1} \nabla f(\theta^{(k)})$

  Gradient vector and Hessian Matrix : (details in the next slide)

- Gradient vector $\nabla f(\theta) = \frac{1}{n} \frac{\partial L(\theta)}{\partial \theta} = -\sum_{i=1}^{n} x_i (y_i - p_i(\theta))\cdot$
- Hessian Matrix $\nabla^2 f(\theta) = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\top p_i(\theta)(1 - p_i(\theta))$
- The Newton iterative update of θ has therefore the following expression :

$$\theta^{(k+1)} = \theta^{(k)} + \left[\sum_{i=1}^{n} x_i x_i^\top p_i(x_i; \theta^{(k)})(1 - p_i(x_i; \theta^{(k)}))\right]^{-1} \sum_{i=1}^{n} x_i (y_i - p_i(x_i; \theta^{(k)}))$$

Optimizing the Logistic Regression function

- **Problem :** Given $(x_i, y_i) \in \mathbb{R}^p \times \{0, 1\}$, $i = 1, \ldots, n$, Let $p_i(\theta) = \frac{\exp(x_i^T \theta)}{1+\exp(x_i^T \theta)}$ be the logistic function.

  In logistic regression we minimize the negative log-likelihood : $\min_\theta f(\theta)$

$$f(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left[-y_i x_i^T \theta + \log\left(1 + \exp(x_i^T \theta)\right) \right].$$

- Newton algorithm : $\theta^{(k+1)} = \theta^{(k)} - \left[\nabla^2 f(\theta^{(k)})\right]^{-1} \nabla f(\theta^{(k)})$

  Gradient vector and Hessian Matrix : (details in the next slide)

- Gradient vector $\nabla f(\theta) = \frac{1}{n} \frac{\partial L(\theta)}{\partial \theta} = -\sum_{i=1}^{n} x_i (y_i - p_i(\theta))\cdot$
- Hessian Matrix $\nabla^2 f(\theta) = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\top p_i(\theta)(1 - p_i(\theta))$
- The Newton iterative update of θ has therefore the following expression :

$$\theta^{(k+1)} = \theta^{(k)} + \Big[\sum_{i=1}^{n} x_i x_i^\top p_i(x_i; \theta^{(k)})(1 - p_i(x_i; \theta^{(k)})) \Big]^{-1} \sum_{i=1}^{n} x_i (y_i - p_i(x_i; \theta^{(k)}))$$

gradient and hessian for the logistic regression's objective

université PARIS-SACLAY

[Seen as Practical Work (TD)]

- Gradient vector

$$\begin{aligned}\nabla f(\theta) = \frac{1}{n}\frac{\partial f(\theta)}{\partial \theta} &= \frac{1}{n}\sum_{i=1}^{n}\left[\frac{\partial}{\partial \theta} - y_i x_i^\top \theta + \frac{\partial}{\partial \theta}\log(1+\exp(x_i^\top \theta))\right] \\ &= \frac{1}{n}\sum_{i=1}^{n} -y_i x_i + x_i p_i(\theta) \\ &= -\frac{1}{n}\sum_{i=1}^{n} x_i(y_i - p_i(\theta))\,.\end{aligned}$$

- Hessian matrix :

$$\begin{aligned}\nabla^2 f(\theta) = \frac{1}{n}\frac{\partial^2 f(\theta)}{\partial \theta \partial \theta^\top} &= \frac{1}{n}\sum_{i=1}^{n} x_i \frac{\partial}{\partial \theta^\top}\{\frac{\exp(x_i^\top \theta)}{1+\exp(x_i^\top \theta)}\} \\ &= \frac{1}{n}\sum_{i=1}^{n} x_i \frac{x_i^\top \exp(x_i^\top \theta)}{\left(1+\exp(x_i^\top \theta)\right)^2} \\ &= \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top p_i(\theta)(1-p_i(\theta))\end{aligned}$$

Iteratively Reweighted Least Squares (IRLS)

Let's write the previous quantities in a matrix form (also very useful for coding) : Let

- $X = (x_1, \ldots, x_n)^\top$ matrix whose rows are the input vectors x_i
- $y = (y_1, \ldots, y_n)^\top$ the vector on binary labels y_i
- $p = (p_1(\theta), \ldots, p_n(\theta))^\top$ the vector of logistic probabilities
- $W = \mathrm{diag}(p \odot (\mathbf{1}_n - p))$ diagonal matrix with $(W)_{ii} = p_i(\theta)\,(1 - p_i(\theta))$
- $z = X\theta^{(k)} + (W^{(k)})^{-1}(y - p^{(k)})$ the current approximate response

Then we can write the vectorized forms of the gradient and the hesssian :

$\hookrightarrow \nabla f(\theta) = -\sum_{i=1}^n x_i(y_i - p_i(x_i; \theta^{(k)})) = -X^\top(y - p^{(k)})$

$\hookrightarrow \nabla^2 f(\theta) = \sum_{i=1}^n x_i x_i^\top p_i(x_i; \theta^{(k)})(1 - p_i(x_i; \theta^{(k)})) = X^\top W X$

Then we get the vectorized form of the Newton iteration :

$$\begin{aligned} \theta^{(k+1)} &= \theta^{(k)} + \Big[\sum_{i=1}^n x_i x_i^\top p_i(x_i; \theta^{(k)})(1 - p_i(x_i; \theta^{(k)}))\Big]^{-1} \sum_{i=1}^n x_i(y_i - p_i(x_i; \theta^{(k)})) \\ &= \theta^{(k)} + \left(X^\top W X\right)^{-1} X^\top (y - p^{(k)}) \\ &= (X^\top W^{(k)} X)^{-1} X^\top W^{(k)} z \end{aligned} \tag{1}$$

The NR update has the form of Least Squares, with weights W, and this is calculated iteratively $\hookrightarrow$ we call it iteratively re-weighted least squares (IRLS).

Iteratively Reweighted Least Squares (IRLS)

Let's write the previous quantities in a matrix form (also very useful for coding) : Let

- $X = (x_1, \ldots, x_n)^\top$ matrix whose rows are the input vectors x_i
- $y = (y_1, \ldots, y_n)^\top$ the vector on binary labels y_i
- $p = (p_1(\theta), \ldots, p_n(\theta))^\top$ the vector of logistic probabilities
- $W = \mathrm{diag}(p \odot (\mathbf{1}_n - p))$ diagonal matrix with $(W)_{ii} = p_i(\theta)\,(1 - p_i(\theta))$
- $z = X\theta^{(k)} + (W^{(k)})^{-1}(y - p^{(k)})$ the current approximate response Then we can write the vectorized forms of the gradient and the hesssian :

$\hookrightarrow \nabla f(\theta) = -\sum_{i=1}^n x_i(y_i - p_i(x_i; \theta^{(k)})) = -X^\top(y - p^{(k)})$

$\hookrightarrow \nabla^2 f(\theta) = \sum_{i=1}^n x_i x_i^\top p_i(x_i; \theta^{(k)})(1 - p_i(x_i; \theta^{(k)})) = X^\top W X$

Then we get the vectorized form of the Newton iteration :

$$\begin{aligned} \theta^{(k+1)} &= \theta^{(k)} + \Big[\sum_{i=1}^n x_i x_i^\top p_i(x_i; \theta^{(k)})(1 - p_i(x_i; \theta^{(k)}))\Big]^{-1} \sum_{i=1}^n x_i(y_i - p_i(x_i; \theta^{(k)})) \\ &= \theta^{(k)} + \left(X^\top W X\right)^{-1} X^\top (y - p^{(k)}) \\ &= (X^\top W^{(k)} X)^{-1} X^\top W^{(k)} z \end{aligned} \tag{1}$$

The NR update has the form of Least Squares, with weights W, and this is calculated iteratively $\hookrightarrow$ we call it iteratively re-weighted least squares (IRLS).

Iteratively Reweighted Least Squares (IRLS)

Let's write the previous quantities in a matrix form (also very useful for coding) : Let

- $X = (x_1, \ldots, x_n)^\top$ matrix whose rows are the input vectors x_i
- $y = (y_1, \ldots, y_n)^\top$ the vector on binary labels y_i
- $p = (p_1(\theta), \ldots, p_n(\theta))^\top$ the vector of logistic probabilities
- $W = \mathrm{diag}(p \odot (\mathbf{1}_n - p))$ diagonal matrix with $(W)_{ii} = p_i(\theta)\,(1 - p_i(\theta))$
- $z = X\theta^{(k)} + (W^{(k)})^{-1}(y - p^{(k)})$ the current approximate response Then we can write the vectorized forms of the gradient and the hesssian :

$\hookrightarrow$ $\nabla f(\theta) = -\sum_{i=1}^n x_i(y_i - p_i(x_i; \theta^{(k)})) = -X^\top(y - p^{(k)})$

$\hookrightarrow$ $\nabla^2 f(\theta) = \sum_{i=1}^n x_i x_i^\top p_i(x_i; \theta^{(k)})(1 - p_i(x_i; \theta^{(k)})) = X^\top W X$

Then we get the vectorized form of the Newton iteration :

$$\begin{aligned}
\theta^{(k+1)} &= \theta^{(k)} + \Big[\sum_{i=1}^n x_i x_i^\top p_i(x_i; \theta^{(k)})(1 - p_i(x_i; \theta^{(k)}))\Big]^{-1} \sum_{i=1}^n x_i(y_i - p_i(x_i; \theta^{(k)})) \\
&= \theta^{(k)} + \left(X^\top W X\right)^{-1} X^\top (y - p^{(k)}) \\
&= (X^\top W^{(k)} X)^{-1} X^\top W^{(k)} z \qquad (1)
\end{aligned}$$

The NR update has the form of Least Squares, with weights W, and this is calculated iteratively $\hookrightarrow$ we call it iteratively re-weighted least squares (IRLS).

Iteratively Reweighted Least Squares (IRLS)

Let's write the previous quantities in a matrix form (also very useful for coding) : Let

- $X = (x_1, \ldots, x_n)^\top$ matrix whose rows are the input vectors x_i
- $y = (y_1, \ldots, y_n)^\top$ the vector on binary labels y_i
- $p = (p_1(\theta), \ldots, p_n(\theta))^\top$ the vector of logistic probabilities
- $W = \mathrm{diag}(p \odot (\mathbf{1}_n - p))$ diagonal matrix with $(W)_{ii} = p_i(\theta)\,(1 - p_i(\theta))$
- $z = X\theta^{(k)} + (W^{(k)})^{-1}(y - p^{(k)})$ the current approximate response Then we can write the vectorized forms of the gradient and the hesssian :

$\hookrightarrow$ $\nabla f(\theta) = -\sum_{i=1}^n x_i(y_i - p_i(x_i; \theta^{(k)})) = -X^\top(y - p^{(k)})$

$\hookrightarrow$ $\nabla^2 f(\theta) = \sum_{i=1}^n x_i x_i^\top p_i(x_i; \theta^{(k)})(1 - p_i(x_i; \theta^{(k)})) = X^\top W X$

Then we get the vectorized form of the Newton iteration :

$$\begin{aligned}
\theta^{(k+1)} &= \theta^{(k)} + \Big[\sum_{i=1}^n x_i x_i^\top p_i(x_i; \theta^{(k)})(1 - p_i(x_i; \theta^{(k)}))\Big]^{-1} \sum_{i=1}^n x_i(y_i - p_i(x_i; \theta^{(k)})) \\
&= \theta^{(k)} + \left(X^\top W X\right)^{-1} X^\top (y - p^{(k)}) \\
&= (X^\top W^{(k)} X)^{-1} X^\top W^{(k)} z \qquad (1)
\end{aligned}$$

The NR update has the form of Least Squares, with weights W, and this is calculated iteratively $\hookrightarrow$ we call it iteratively re-weighted least squares (IRLS).