Lecture 6: Probabilistic Classification - 24/02/20

Generalised Linear Model

Review: Discriminant Function

With linear regression predict real number with

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

A 2-class linear discriminant:

- ***Evaluates*** : $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$
- ***Assigns*** $\mathbf{x}$ to class C_1 if $y(\mathbf{x}) \geq 0$ and to C_0 otherwise
- Decision boundary defined by $y(\mathbf{x}) = 0$
- Decision boundaries are hyperplanes

To generalise this, we want a function $y(\mathbf{x})$ which...

- predicts class labels, $1, \ldots, K$
- or posterior probabilities of class labels: $p(C_k|\mathbf{x})$
- e.g. for 2 -classes either $y(\mathbf{x}) \in \{0, 1\}$ or $y(\mathbf{x}) \in [0, 1]$

Can achieve this with ***non-linear activation function*** f and

$$y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x} + w_0)$$

- Decision boundaries are surfaces where $y(\mathbf{x}) = constant$, namely $(D-1)$ -dimensional hyperplanes
- So decision surfaces are linear even though f is not

> ***Generalised Linear Model***
>
> $$y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x} + w_0)$$
>
> for non-linear activation function f

- Not lienar in the parameters (unlike linear regression models)
- Implies more complex analytical and computational procedures
- Nevertheless, they are still relatively simple
- Can replace input, $\mathbf{x}$, with a fixed non-linear transformation to a vector of basis functions, $\Phi(\mathbf{x})$

Generative Model

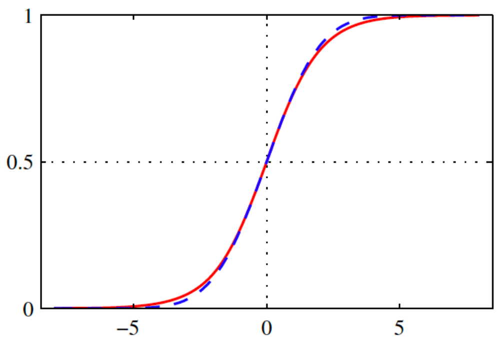
Consider the generative approach:

- class-conditional densities $p(\mathbf{x}|C_k)$
- class priors $p(C_k)$
- use Bayes Theorem to compute $p(C_k|\mathbf{x})$
- consider 2-classes:

$$
\begin{aligned}
p(C_1|\mathbf{x}) &= \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1)+p(\mathbf{x}|C_0)p(C_0)} \\
&= \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1)+p(\mathbf{x}|C_0)p(C_0)} \times \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1)} \\
&= \frac{1}{1+\frac{p(\mathbf{x}|C_0)p(C_0)}{p(\mathbf{x}|C_1)p(C_1)} \times \frac{p(\mathbf{x})}{p(\mathbf{x})}} \\
&= \frac{1}{1+\frac{p(C_1|\mathbf{x})}{p(C_0|\mathbf{x})}} \\
&= \frac{1}{1+\exp(-a(\mathbf{x}))}
\end{aligned}
$$

where $a(\mathbf{x}) = \ln(\frac{p(C_1|\mathbf{x})}{p(C_0|\mathbf{x})})$

The Logistic Sigmoid



$$\boxed{\sigma(a) = \frac{1}{1+e^{-a}} \quad (6.2)}$$

- Sigmoid shown in red
- Gaussian CDF shown in blue

- Sigmoid means S-shaped
- Maps from real line to interval $[0, 1]$
- Symmetric, $\sigma(-a) = 1 - \sigma(a)$
- Inverse called logit or log-odds-ratio:

$$a = \ln\left(\frac{\sigma}{1-\sigma}\right) = \ln\left(\frac{p(C_1|\mathbf{x})}{p(C_0|\mathbf{x})}\right)$$

2-Gaussian Classes with Identical Covariance

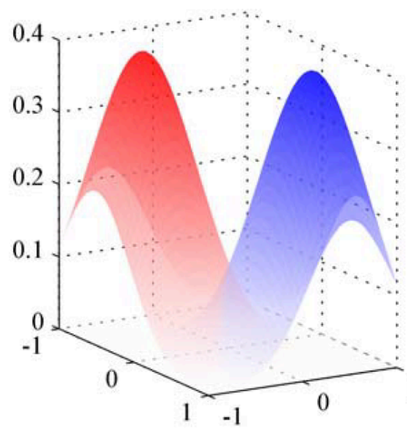
Assume each class distribution $p(\mathbf{x}|C_k)$ is Gaussian with the same covariacne matrix, $\mathbf{S}$, then we can show

$$p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0)$$

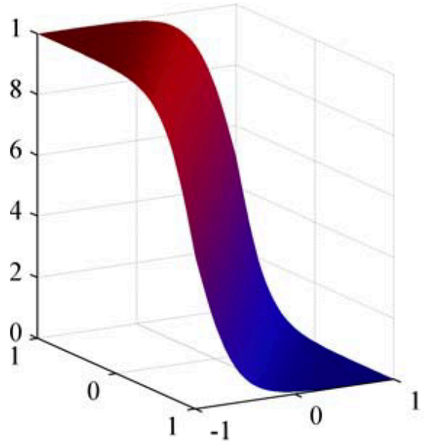
where

$$
\begin{aligned}
\mathbf{w} &= \mathbf{S}^{-1}(\mu_1 - \mu_0) \\
w_0 &= -\frac{1}{2}\mu_1^T\mathbf{S}^{-1}\mu_1 + \frac{1}{2}\mu_0^T\mathbf{S}^{-1}\mu_0 + \ln\frac{p(C_1)}{p(C_0)}
\end{aligned}
$$

- Posterior $p(C_1|\mathbf{x})$ is a generalised linear function
- Prior probabilities, $p(C_k)$ enter only through the bias term
- Minimum misclassification decision boundary is lienar



Class-conditional densities for two classes:
$p(\boldsymbol{x}|\mathcal{C}_1) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{S})$ in red
$p(\boldsymbol{x}|\mathcal{C}_0) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_0, \boldsymbol{S})$ in blue



Corresponding posterior probability $p(\mathcal{C}_1|\boldsymbol{x})$, given by a logistic sigmoid of a linear function of $\boldsymbol{x}$.

Assume a joint distribution for input $\mathbf{x}_n$, class C_k

$$\begin{aligned} p(\mathbf{x}_n, C_1) = p(C_1)p(\mathbf{x}_n|c_1) = \pi N(\mathbf{x}_n|\mu_1, \mathbf{S}) \\ p(\mathbf{x}_n, C_0) = p(C_0)p(\mathbf{x}_n|c_0) = (1-\pi)N(\mathbf{x}_n|\mu_0, \mathbf{S}) \end{aligned}$$

with prior $p(C_1) = \pi$, class means μ_1 & μ_0, and shared covariacne $\mathbf{S}$

> $$p(\mathbf{t}|\mathbf{X}, \pi, \mu_1, \mu_0, \mathbf{S}) = \prod_{n=1}^{N} q_1(\mathbf{x}_n)^{t_n} q_0(\mathbf{x}_n)^{(1-t_n)}$$
>
> where
>
> $$\begin{aligned} q_1(\mathbf{x}_n) = \pi N(\mathbf{x}_n|\mu_1, \mathbf{S}) \\ q_0(\mathbf{x}_n) = (1-\pi)N(\mathbf{x}_n|\mu_0, \mathbf{S}) \end{aligned}$$

Maximum Likelihood (ML) Solution for this model

> Class-bias and means:
>
> $$\begin{aligned} \pi^* &= \frac{1}{N}\sum_n t_n = \frac{N_1}{N_1 + N_0} \\ \mu_1{}^* &= \frac{1}{N_1}\sum_n t_n \mathbf{x}_n \\ \mu_0{}^* &= \frac{1}{N_0}\sum_n (1-t_n)\mathbf{x}_n \end{aligned}$$

- N_k is number of points of class k
- Class bias, π, is the fraction of positive data-points
- Class-means are simply the means of each classes data-points

> Covariane:
>
> $$\mathbf{S}^* = \frac{N_1}{N}\mathbf{S}_1 + \frac{N_0}{N}\mathbf{S}_0$$

> where $\mathbf{S}_1 = \frac{1}{N_1}\sum_n t_n(\mathbf{x}_n - \mu_1)(\mathbf{x} - \mu_1)^T$ and
> $\mathbf{S}_0 = \frac{1}{N_0}\sum_n (1 - t_n)(\mathbf{x}_n - \mu_0)(\mathbf{x} - \mu_0)^T$

- Covariance is the weighted sum of the class covariance
- Shared covariance assumed
- But it takes $\frac{D(D+1)}{2}$ computations for 2-class and $k\frac{D(D+2)}{2}$ for k-class, the time complexity is $O(D^2)$

The Soft-Max

> For K-classes:
>
> $$\begin{aligned} p(C_k|\mathbf{x}) &= \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_{j=1}^K p(\mathbf{x}|C_j)p(C_j)} \\ &= \frac{\exp(a_k)}{\sum_{j=1}^K \exp(a_j)} \end{aligned}$$

A normalised exponential: a natural extention to the logistic sigmoid, and where $a_k = \ln p(\mathbf{x}|C_k)p(C_k)$

Normalised exponential is sometimes called ***soft-max***, because:

- If $a_k \gg a_j$ for all $j \neq k$
- then $p(C_k|\mathbf{x}) \simeq 1$ and $p(C_j|\mathbf{x}) \simeq 0$

For K-classes all with the same covariance matrix:

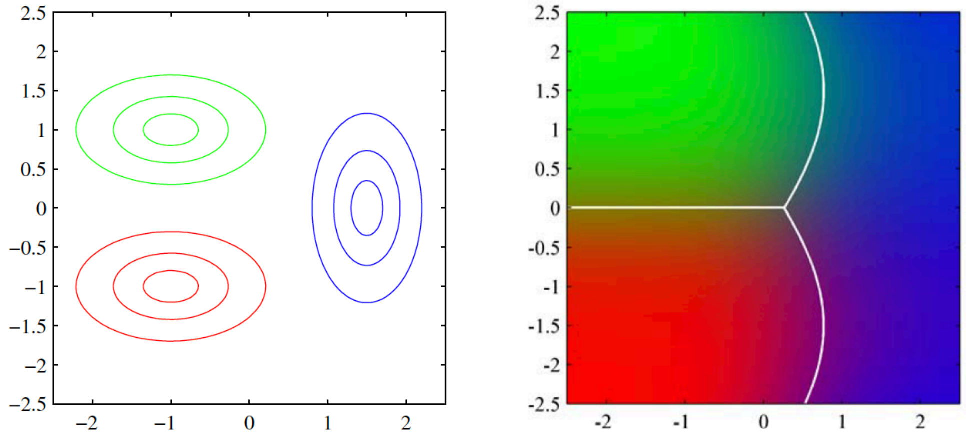
$$a_k = \mathbf{w}_k^T\mathbf{x} + w_{k0}$$

where $\mathbf{w}_k = \mathbf{S}^{-1}\mu_k$ and $w_{k0} = -\frac{1}{2}\mu_k^T\mathbf{S}^{-1}\mu_k + \ln p(C_k)$

- terms a_k again linear in x
- minimum misclassification boundaris again linear

If each class has an ***independent covariance matrix***

- *a_k is quadratic in $\mathbf{x}$*
- *Gives rise to quadratic discriminant with quadratic decision boundaries*



- Three classes: green, red & blue
- Left – class contours.
  Green & red classes have the same covariance matrix
- Right – posterior class probabilities (given by colour density).
  Minimum misclassification boundaries shown in white.

A Discriminative Approach

Recall that we can directly learn posterior class probabilities $p(C_k|\mathbf{x})$, for classification:

- useful when complexities in $p(\mathbf{x}|C_k)$ do not (or only weakly) influence classification task
- or if no good distributional form for each $p(\mathbf{x}|C_k)$

We take inspiration from:

- 2-Classes with Gaussian densities and $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2$ leads to:

$$p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0)$$

- Logistic sigmoid of quadratic functions when $\Sigma_1 \neq \Sigma_0$

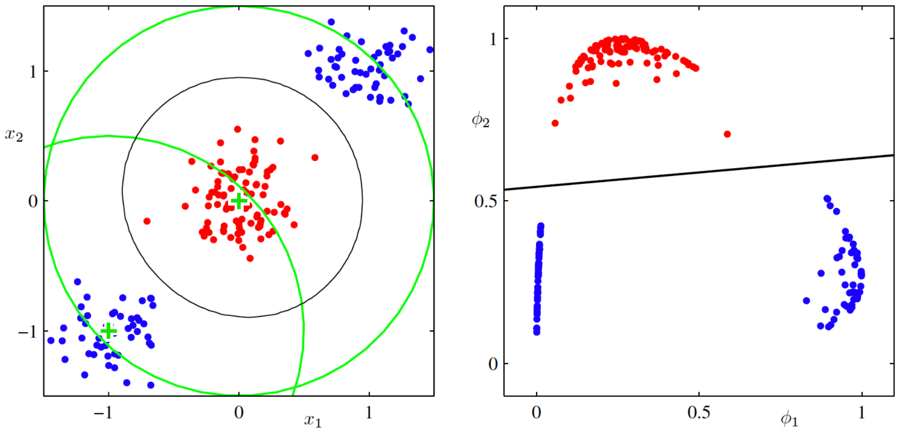
So why not instead, directly define a probabilistic discriminant using the logistic sigmoid?

- Typically this has fewer parameters to fit
- Can improve performance, if generative assumptions lead to poor approximations (e.g. classes not Gaussian)
- As with linear regression, we can use fixed basis functions:

$$\phi(\mathbf{x}) = (\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \ldots, \phi_{M-1}(\mathbf{x}))$$

with $\phi_0(\mathbf{x}) = 1$

- No closed form solution, but iterative approach exist



- Left – classes not linearly separable & not Gaussian
- Left – 2 potential RBF centres and contours shown (green)
- Right – In feature space, classes are linearly separable

Logistic Regression

Consider just 2-classes and fixed basis function ϕ:

- Posterior probability of C_1 written as:

$$p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x}))$$

(No separate bias term, as $\phi(\mathbf{x}) = 1$)

- Using logistic sigmoid:

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

- For M basis functions, we have M -parameters (elements of $\mathbf{w}$)
- Comparisiom: to fit our generative model our paramters would comprise: $2M$ for the means, $\frac{M(M+1)}{2}$ for shared covariance and 1 for class bias

Discriminative Likelihood and Related Error

For data-points (ϕ_n, t_n) where $t_n \in \{0, 1\}$ and $\phi_n = \phi(\mathbf{x}_n)$ for $n = 1, \ldots, N$ our discriminative model has likelihood:

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

for $\mathbf{t} = (t_1, \ldots, t_N)^T$ and $y_n = p(C_1|\phi_n) = \sigma(\mathbf{w}^T \phi_n)$

As with regression, take negative log likelihood as error function:

> *The Cross-Entropy Error Function*

$$
\begin{aligned}
E(\mathbf{w}) &= -\ln p(\mathbf{t}|\mathbf{w}) \\
&= -\sum_{n=1}^{N}(t_n \ln y_n + (1-t_n)\ln(1-y_n))
\end{aligned}
$$

Take gradient to minimise the error:

$$
\begin{aligned}
\nabla_{\mathbf{w}} E(\mathbf{w}) &= \sum_{n=1}^{N}(\sigma(\mathbf{w}^T \mathbf{\Phi}_n) - t_n)\mathbf{\Phi}_n \\
&= \mathbf{\Phi}(\mathbf{y}-\mathbf{t})
\end{aligned}
$$

with design-matrix, $\mathbf{\Phi}$, targets $\mathbf{t} = (t_1, \ldots, t_n)^T$, predictions $y_n = \sigma(\mathbf{w}^T \phi_n)$ and prediction vector $\mathbf{y} = (y_1, \ldots, y_N)^T$

Unfortunately, the non-linear form means we cannot simply set to zero and rearrange

Gradient Ascent Methods

We want to find the maximum of function $f : \mathbb{R}^D \to \mathbb{R}$, and we can calculate the gradient $\nabla f = \nabla_{\mathbf{z}} f$ for any point $\mathbf{z}$

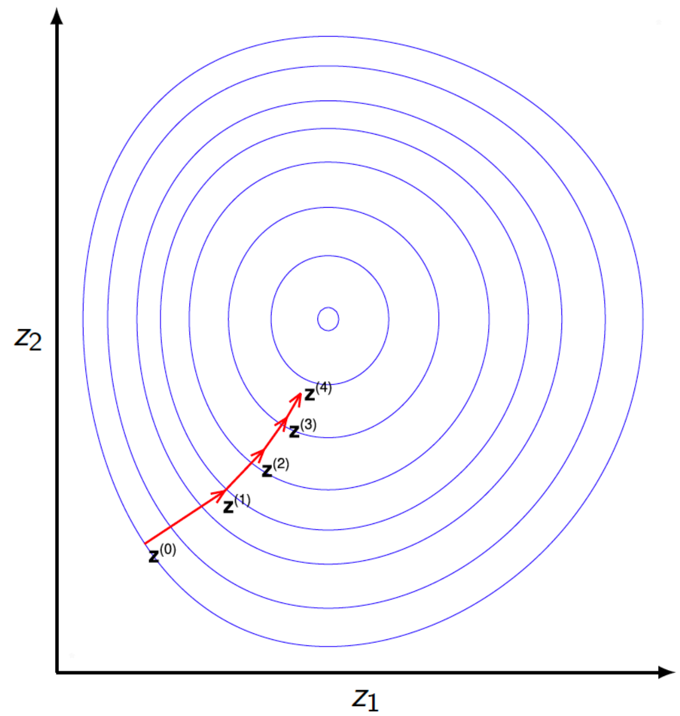
Recipe:

- Choose an initial exstimate $\mathbf{z}^{(0)}$ (possibly randomly)
- Repeatedly ipdate estimate with:

$$
\mathbf{z}^{(\tau+1)} = \mathbf{z}^{\tau} + \eta \nabla f
$$

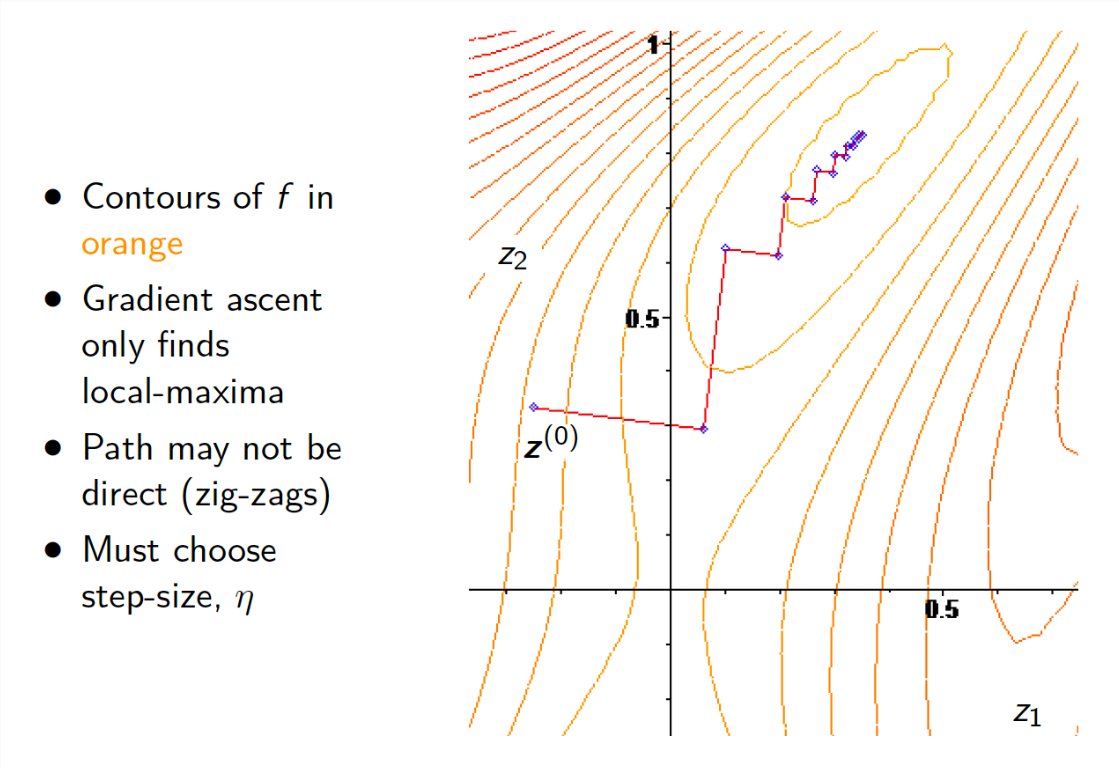
where $\eta > 0$ is a small step size and ∇f is evaluated at $\mathbf{z}^{(\tau)}$

- Stop when magnitude of ∇f falls below a thredhold
- Eventualy reachs a local maximum, if η small enough



- Contours of f in blue
- $\nabla f(z)$ is the direction of steepest ascent
- Each iteration steps a small distance η in direction $\nabla f(z)$.
- As we approach the maximum ∇f approaches $\mathbf{0}$

Problem



Newton-Raphson Method

Consider a univariate function $f : \mathbb{R} \to \mathbb{R}$ which is twice differentiable. Define f_τ, the second order Taylor expansion of f around point $x^{(\tau)}$:

$$\begin{aligned} f(x) \approx f_\tau(x) &= f_\tau(x^{(\tau)} + u_\tau) \\ &= f(x^{(\tau)}) + f'(x^{(\tau)})u_x + \frac{1}{2}f''(x^{(\tau)})u_\tau^2 \end{aligned}$$

where $u_\tau = x - x^{(\tau)}$

u_τ Maximises/minimises this expression when:

$$\begin{aligned} \frac{d}{du_\tau}(f_\tau(x^{(\tau)} + u_\tau)) &= f'(x^{(\tau)}) + f''(x^{(\tau)})u_\tau = 0 \\ \implies u_\tau &= -\frac{f'(x^{(\tau)})}{f''(x^{(\tau)})} \end{aligned}$$

> ***Message***: can use second order derivatives to climb functions more quickly

We seek the maximum of function $f : \mathbb{R}^D \to \mathbb{R}$, and can calculate gradient $\nabla_z f$ and Hessian $\mathbf{H}(\mathbf{z}) = \nabla\nabla f = \nabla^2 f$ for any $\mathbf{z}$.

The Hessian is the matrix of second derivatives evaluatied at $\mathbf{z}$:

$$[\mathbf{H}(\mathbf{z})]_{ij} = \frac{\partial^2 f}{\partial z_i \partial z_j}|_\mathbf{z}$$

$$
\mathbf{H}(\mathbf{z}) = \begin{pmatrix} \frac{\partial^2 f}{\partial z_1^2} & \frac{\partial^2 f}{\partial z_1 \partial z_2} & \cdots & \frac{\partial^2 f}{\partial z_1 \partial z_n} \\ \frac{\partial^2 f}{\partial z_2 \partial z_1} & \frac{\partial^2 f}{\partial z_2 \partial z_2} & \cdots & \frac{\partial^2 f}{\partial z_2 \partial z_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial z_n \partial z_1} & \frac{\partial^2 f}{\partial z_n \partial z_2} & \cdots & \frac{\partial^2 f}{\partial z_n^2} \end{pmatrix}
$$

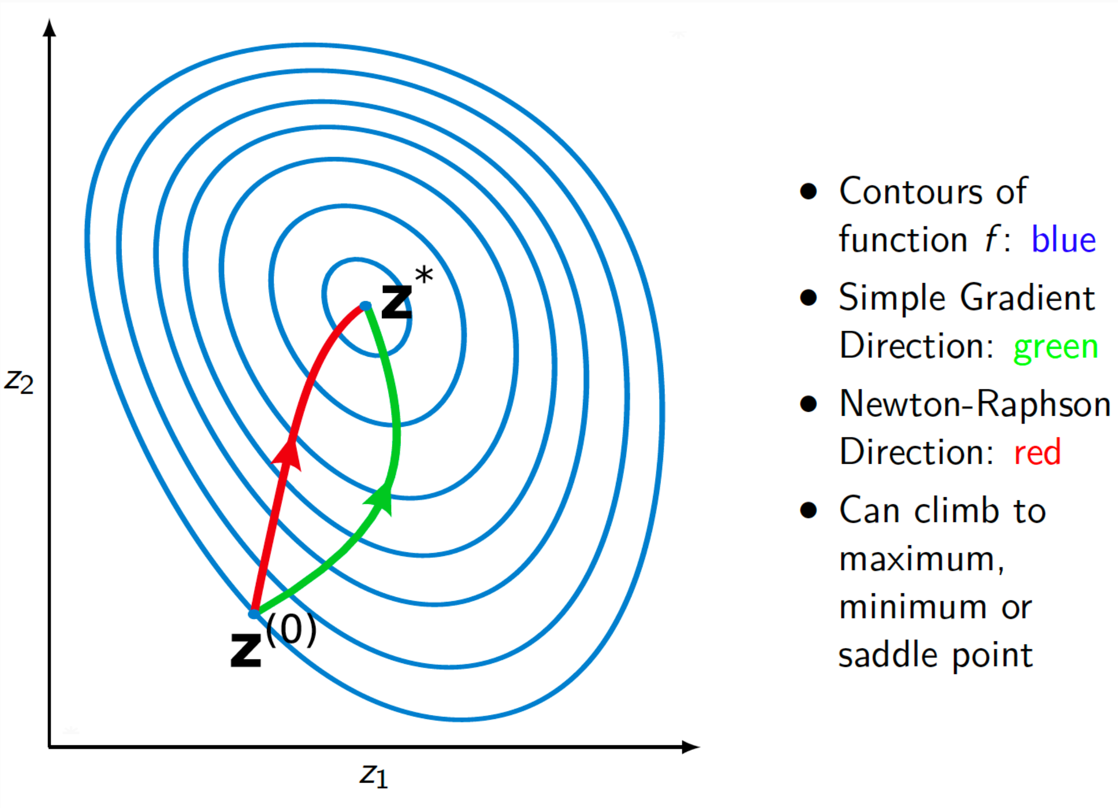
Recipe:

- Choose an initial estimate $\mathbf{z}^{(0)}$ (possibly randomly)
- Repeatedly update estimate with:

$$
\mathbf{z}^{(\tau+1)} = \mathbf{z}^{(\tau)} - \mathbf{H}^{-1} \nabla f
$$

where ∇f and $\mathbf{H}$ are evaluated at $\mathbf{z}^{(\tau)}$

- Stop when magnitude of $\mathbf{H}^{-1} \nabla f$ falls below a threshold



Linear Regression with Newton-Raphson Method

Apply the ***Newton-Raphson Method*** to Linear Regression model:

$$
\begin{aligned}
E(\mathbf{w}) &= \frac{1}{2}(\mathbf{t} - \mathbf{\Phi w})^T(\mathbf{t} - \mathbf{\Phi w}) \\
\nabla E(\mathbf{w}) &= \mathbf{\Phi}^T \mathbf{\Phi w} - \mathbf{\Phi}^T \mathbf{w} \\
\mathbf{H}(\mathbf{w}) &= \nabla^2 E(\mathbf{w}) = \mathbf{\Phi}^T \mathbf{\Phi}
\end{aligned}
$$

Iterative function:

$$
\begin{aligned}
\mathbf{w}^{(new)} &= \mathbf{w}^{(old)} - (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} (\mathbf{\Phi}^T \mathbf{\Phi} \mathbf{w}^{(old)} - \mathbf{\Phi}^T \mathbf{t}) \\
&= (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{t} \\
&= \mathbf{w}_{ML}
\end{aligned}
$$

*We get the maximum in a **single step** for any* $\mathbf{w}^{(old)}$

Logistic Regression with Newton-Raphson Method

For Logistic Regression, we seek to minimise ***cross-entropy error***:

$$
\begin{aligned}
E(\mathbf{w}) &= -\sum_{n=1}^{N}(t_n \ln y_n + (1-t_n)\ln(1-y_n)) \\
\nabla E(\mathbf{w}) &= \mathbf{\Phi}(\mathbf{y}-\mathbf{t})
\end{aligned}
$$

with design-matrix, $\mathbf{\Phi}$, targets $\mathbf{t} = (t_1, \dots, t_N)^T$, predictions $y_n = \sigma(\mathbf{w}^T \phi_n)$ and prediction vector $\mathbf{y} = (y_1, \dots, y_N)^T$

Hessian Matrix:

$$
\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1-y_n)\mathbf{\Phi}_n^T\mathbf{\Phi}_n = \mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}
$$

Where $\mathbf{R} \in \mathbb{R}^{N\times N}$ is a diagonal matrix with non-zero elements:

$$
[\mathbf{R}]_{nn} = y_n(1-y_n)
$$

Algorithm

Iteratively reweighted least squares (IRLS) algorithm [Rub83]:

procedure $\mathrm{IRLS}(\boldsymbol{\Phi}, \boldsymbol{t}, \boldsymbol{w}^{(0)}, \theta)$
    # $\boldsymbol{\Phi}$ is design matrix, $\boldsymbol{t}$ is target vector
    # $\boldsymbol{w}^{(0)}$ is initial weight vector, $\theta > 0$ is threshold
    for $\tau = 1, 2, \dots$ **do**
        for $n = 1, \dots, N$ **do**
            $y_n = \sigma(\boldsymbol{w}^T\phi_n)$
            $r_{nn} = y_n(1-y_n)$
        $\boldsymbol{y} = (y_1, \dots, y_N)^T$
        $\boldsymbol{R} = \mathrm{diag}((r_{11}, \dots, r_{nn})^T)$
        $\boldsymbol{w}^{(\tau)} = \boldsymbol{w}^{(\tau-1)} - (\boldsymbol{\Phi}^T\boldsymbol{R}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T(\boldsymbol{y}-\boldsymbol{t})$
        if $\|\boldsymbol{w}^{(\tau)} - \boldsymbol{w}^{(\tau-1)}\| < \theta$ **then**
            return $\boldsymbol{w}^{(\tau)}$

where ∇E in red, $\boldsymbol{H}$ in blue, $\mathrm{diag}(\cdot)$ constructs a diagonal matrix from a vector, and $\|\boldsymbol{w}^{(\tau)} - \boldsymbol{w}^{(\tau-1)}\|$ is Euclidean distance from $\boldsymbol{w}^{(\tau)}$ to $\boldsymbol{w}^{(\tau-1)}$.

Important: $\boldsymbol{y}$ and $\boldsymbol{R}$ are re-evaluated each iteration