Chapter 3
Linear models for classification

I) Logistic regression

Remember that the model is the following:

$$p(y|x,w) = \mathcal{B}(y \mid \text{sigma}(w^T x))$$ (and, sometimes, this is replaced by $\text{sigma}(w_0 + w^T x)$)

↳ y is a Bernoulli law of parameter $\text{sigma}(w^T x)$

1) MLE

The negative log-likelihood for logistic regression is given by:

$$NLL(w) = -\sum_{i=1}^{N} \log\left(\mu_i^{\mathbb{1}(y_i=1)} (1-\mu_i)^{\mathbb{1}(y_i=0)}\right)$$

↳ where $\mu_i = \text{sigma}(w^T x_i)$

$$= -\sum_{i=1}^{N} \left[y_i \log \mu_i + (1-y_i)\log(1-\mu_i)\right]$$

(remember: $\text{sigm}(\eta) = \frac{1}{1+e^{-\eta}}$)

Let us set $\tilde{y}_i \begin{cases} = 1 & \text{if } y_i = 1 \\ = -1 & \text{if } y_i = 0 \end{cases}$.

We then have:
$$\begin{cases} \mathbb{P}(\tilde{y}_i = 1) = \dfrac{1}{1+\exp(-w^T x_i)} \\ \mathbb{P}(\tilde{y}_i = -1) = \dfrac{1}{1+\exp(w^T x_i)} \end{cases}$$

and so: $NLL(w) = \sum_{i=1}^{N} \log\left(1 + \exp\left(-\tilde{y}_i w^T x_i\right)\right)$.

There is no closed form expression for the MLE. We need to use an optimization algorithm to compute it.

Let us set $\sigma(\dots) = \text{sign}(\dots)$.

We have : $\sigma'(t) = \frac{e^{-t}}{(1+e^{-t})^2} = \frac{1}{1+e^{-t}}\left(1 - \frac{1}{1+e^{-t}}\right)$

$= \sigma(t)(1-\sigma(t))$

So :

$$\underbrace{\nabla NLL(w)}_{=: g(w)} = -\sum_{i=1}^{N} \left\{ \frac{y_i}{\mu_i} \sigma'(w^T x_i) \times x_i - \frac{(1-y_i)}{1-\mu_i} \sigma'(w^T x_i) \times x_i \right\}$$

$$= -\sum_{i=1}^{N} \left\{ \left(\frac{y_i}{\mu_i} \mu_i(1-\mu_i) x_i - \frac{(1-y_i)}{1-\mu_i} \mu_i(1-\mu_i) x_i \right\} \right.$$

$$= -\sum_{i=1}^{N} \left\{ (y_i - \mu_i) x_i \right\}$$

$$\boxed{g(w) = \sum_{i=1}^{N} (\mu_i - y_i) x_i}$$

We write : $x_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,D} \end{pmatrix}$ $X = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,D} \\ \vdots & \vdots & & \vdots \\ x_{N,1} & x_{N,2} & \dots & x_{N,D} \end{bmatrix}$.

(the space is $\mathbb{R}^D$ and we have N observations)

So $\nabla NLL(w) = \underbrace{\begin{bmatrix} x_{1,1} & \dots & x_{N,1} \\ x_{1,2} & & x_{N,2} \\ \vdots & & \vdots \\ x_{1,D} & & x_{N,D} \end{bmatrix}}_{= X^T} \times \underbrace{\begin{pmatrix} \mu_1 - y_1 \\ \vdots \\ \mu_N - y_N \end{pmatrix}}_{= \mu - y}$

$$\boxed{H(w) = \sum_{i=1}^{N} (\nabla_w \mu_i) x_i^T = \sum_{i=1}^{N} \mu_i(1-\mu_i) x_i x_i^T}$$

$$\boxed{= X^T S X}$$

where $S := \text{diag}\left(\mu_i(1-\mu_i)\right)_{1 \le i \le N}$

Suppose that $\forall i,\ \mu_i \in (0,1)$ and that $N > D$, $\operatorname{rank}(X) = D$. In this case:

We write $\vec{x}_j = \begin{pmatrix} x_{1,j} \\ x_{2,j} \\ \vdots \\ x_{N,j} \end{pmatrix}$ for all $j \in \{1, 2, \dots, D\}$.

For any $\eta = \begin{pmatrix} \eta_1 \\ \vdots \\ \eta_D \end{pmatrix} \in \mathbb{R}^D$, we have: $X\eta = \left[\vec{x}_1 \middle| \dots \middle| \vec{x}_D\right] \begin{pmatrix} \eta_1 \\ \vdots \\ \eta_D \end{pmatrix}$

$$= \eta_1 \vec{x}_1 + \dots + \eta_D \vec{x}_D.$$

So: $$\eta^T X^T S X \eta = (\eta_1 \vec{x}_1 + \dots + \eta_D \vec{x}_D)^T S (\eta_1 \vec{x}_1 + \dots + \eta_D \vec{x}_D)$$

$$= \sum_{i=1}^{N} \mu_i (1-\mu_i) \left(\sum_{j=1}^{D} \eta_j x_{i,j} \right)^2$$

↳ It is always ≥ 0.

And if it is equal to zero then:

$$\forall i \in \{1, 2, \dots, N\} : \sum_{j=1}^{D} \eta_j x_{i,j} = 0$$

That is: $$\sum_{j=1}^{D} \eta_j \vec{x}_j = 0$$

(as $\vec{x}_1, \dots, \vec{x}_D$ are linearly independent)

this proves $\eta_1 = \eta_2 = \dots = \eta_D = 0$.

So H is definite positive.

2) Optimization algorithms (non exhaustive list)

a) Steepest descent (/ gradient descent)

We define a sequence: $$\begin{cases} \theta_0 \in \mathbb{R}^D \\ \theta_{k+1} = \theta_k - \gamma_k g(\theta_k) \end{cases}$$

γ_k: step size / learning rate

How should we set the step size?

- If it is too small, the convergence will be slow.
- If it is too big, it might never converge.

We observe that: $f(\theta + \eta d) \approx f(\theta) + \eta \nabla f(\theta) d$

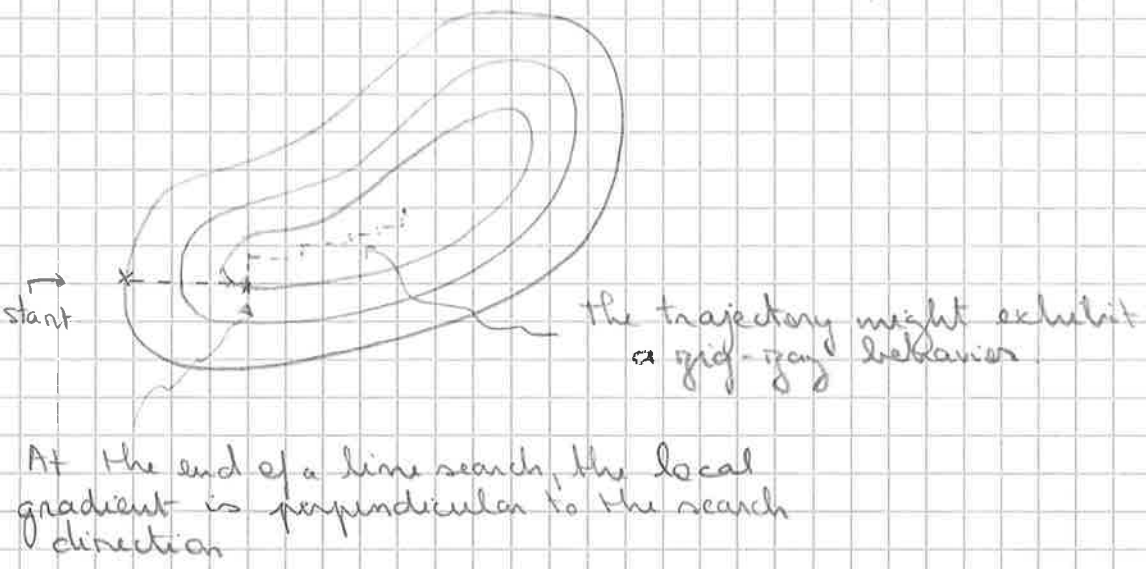
(function we want to minimize; Taylor expansion; where d is our descent direction)

If η is small enough then $f(\theta + \eta d) < f(\theta)$.

Let us pick η to minimize: $\phi(\eta) = f(\theta + \eta d)$.

This is called <u>line minimization</u> / <u>line search</u>.

↳ This is a 1-d minimization problem which can be solved using various methods.



<u>b) Newton's method</u> (for a strictly convex function)

We could obtain a faster optimization method by taking into account the curvature of the space (i.e. the Hessian). These are called <u>second order</u> optimization methods. The primary example is <u>Newton's algorithm</u>.

We set: $\begin{cases} \theta_0 \in \mathbb{R}^d \\ \theta_{k+1} = \theta_k - \gamma_k H_k^{-1} g_k \end{cases}$

where: $\begin{cases} H_k = H(\theta_k) \\ g_k = \nabla f(\theta_k) \end{cases}$

(H is the Hessian of f, f is the function to minimize)

$\begin{cases} d_k \text{ sol. of } H_k d_k = -g_k \\ \gamma_k = \underset{\gamma}{\operatorname{argmin}} f(\theta_k + \gamma_k d_k) \end{cases}$

the idea behind the algorithm is the following.

$$f(\theta) = f(\theta_k) + g_k^T(\theta - \theta_k) + \frac{1}{2}(\theta - \theta_k)^T H_k (\theta - \theta_k) + \dots$$

↑ second order Taylor expansion

$$f(\theta) = \underbrace{c}_{f_k - g_k^T \theta_k} + \underbrace{b^T \theta}_{b = g_k - H_k \theta_k} + \theta^T A \theta \quad \longrightarrow \quad A = \frac{1}{2} H_k$$

↓

the min. of this is at: $\theta = -\frac{1}{2} A^{-1} b = \theta_k - H_k^{-1} g_k$

thus the Newton step $d_k = -H_k^{-1} g_k$ is what should be added to θ_k to minimize the second order approximation of f around θ_k. We choose then γ_k (not necessarily $=1$) using a line search to be even more optimal.

c) Application of Newton's method : IRLS

We want to apply the above method to find the MLE for binary logistic regression. We fix $\gamma_k = 1$ (as the Hessian is constant, the Taylor expansion of order 2 is exact, so the best θ is given for $\gamma_k = 1$).

We have: $\theta_{k+1} = \theta_k - H^{-1} g_k$

$$= \theta_k + (X^T S_k X)^{-1} X^T (y - \mu_k)$$

(μ_k: at iteration k)

$$= (X^T S_k X)^{-1} [(X^T S_k X)\theta_k + X^T (y - \mu_k)]$$

$$= (X^T S_k X)^{-1} X^T [S_k X \theta_k + y - \mu_k]$$

$$= (X^T S_k X)^{-1} X^T S_k z_k$$

(z_k: $= X\theta_k + S_k^{-1}(y - \mu_k)$, called working response)

Observe that θ_{k+1} is a minimizer of:

$$(D_k z_k - D_k X\theta)^T (D_k z_k - D_k X\theta)$$

where $S_k = D_k^2$

and: $(D_k z_k - D_k X\theta)^T (D_k z_k - D_k X\theta)$

$$= \sum_{i=1}^{N} \sqrt{\mu_i(1-\mu_i)} (z_{ki} - \theta^T x_i)^2$$

which is called a weighted least square problem.

because $(X^T S_k X)^{-1} X^T S_k z_k = (X^T D_k^T D_k X)^{-1} X^T D_k^T D_k z_k$ + remember chapter 2

We can write for each component of z_k:

$$z_{ki} = \theta_k^T x_i + \frac{y_i - \mu_{ki}}{\mu_{ki}(1-\mu_{ki})}$$

This algorithm is known as iterated reweighted least square (IRLS) since at each iteration, we solve a weighted least squares problem, where the weight matrix S_k changes at each iteration.

Algorithm

$\theta = 0$; $w_0 = \log(\bar{y}/(1-\bar{y}))$

Repeat:

$$\left.\begin{aligned} \eta_i &= w_0 + \theta^T x_i \\ \mu_i &= \text{sigm}(\eta_i) \\ s_i &= \mu_i(1-\mu_i) \\ z_i &= \eta_i + \frac{y_i - \mu_i}{s_i} \end{aligned}\right\} \text{ for all } i$$

$$S = \text{diag}(s_{1:N})$$

$$\theta = (X^T S X)^{-1} X^T S z$$

until converged

3) l_2 - regularization

If the data is linearly separable, the MLE is obtained when $\|w\| \to +\infty$, corresponding to an infinitely steep sigmoid function $\mathbb{1}_{w^T x > w_0}$ also known as <u>linear threshold limit</u>. This assigns the maximal amount of probability mass to the training data (which will not be good for generalisation).

To prevent this, we can use a l_2 - regularisation (just like in the ridge regression). The new objective function, gradient and Hessian have the form:

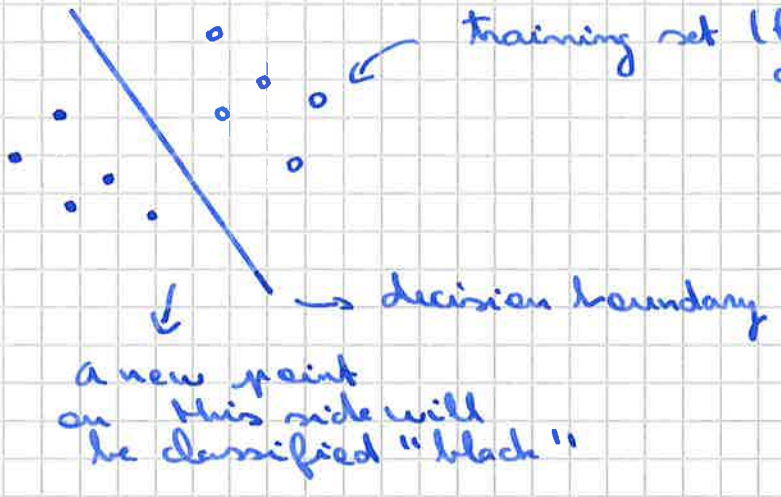
$$\begin{cases} \tilde{f}(w) = NLL(w) + \lambda w^T w \\ \tilde{g}(w) = g(w) + \lambda w \\ \tilde{H}(w) = H(w) + \lambda \, Id \end{cases}$$

(again, we can apply our optimization alg. to $\tilde{f}$).

II) Linear support vector machine (SVM)

The linear SVM uses an hyperplane as a decision boundary

Ex:



The decision boundary is $\{x : w^T x + b = 0\}$ for a vector w and a scalar b.

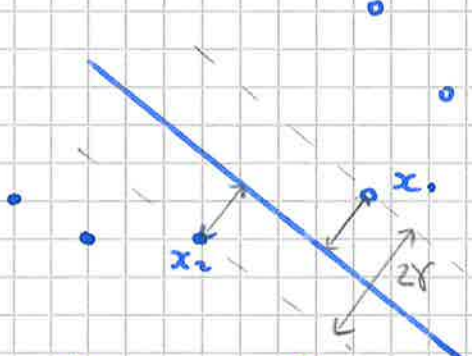
The SVM decision function for a test point x_{new} is therefore given by: $t_{new} = \text{sign}(w^T x_{new} + b)$

$(t_{new} \in \{-1, +1\})$

1) The margin

The margin is defined as the perpendicular distance from the decision boundary to the closest point on either side.

Ex: margin $= \gamma$



x_1, x_2 and the boundary are always such that dist(x_1, boundary) = dist(x_2, boundary)

2) Maximizing the margin

w is a vector perpendicular to the boundary so

$$2\gamma = \frac{1}{\|w\|} w^T (x_1 - x_2)$$

the decision function $h_{new} = \text{sign}(w^T x_{new} + b)$ is invariant to the scaling of its argument by a positive constant. this means that we can multiply $w^T x_{new} + b$ by a positive λ and the output of the function will be unchanged. Therefore we can fix the scaling of w and b such that $w^T x + b = \pm 1$ for the closest points on either side. thus:

$$2\gamma = \frac{w^T}{\|w\|}(x_1 - x_2) = \frac{1}{\|w\|}(w^T x_1 + b - w^T x_2 - b)$$

$$= \frac{(1+1)}{\|w\|}$$

$$\gamma = \frac{1}{\|w\|}$$

To maximize the margin, we must therefore maximize $\frac{1}{\|w\|}$.

There are some constraints : $\begin{cases} w^T x_n + b \geq 1 & \text{for all points in class 1} \\ w^T x_n + b \leq -1 & \text{for all points in class 2} \end{cases}$

We formalize the problem in the following form:

(1) $\begin{cases} \text{find } \hat{w} = \underset{w}{\text{argmin}} \frac{1}{2}\|w\|^2 \\ \text{subject to } t_n(w^T x_n + b) \geq 1 \text{ for all } n \end{cases}$

To solve this optimization problem with constraints, we need to incorporate the constraints into the objective function through a set of Lagrange multipliers.

Remark: See the optimization course (M1 in) for the Lagrange multipliers or
[BD2007] Binmore, Ken and Davies, Joan ; Calculus, concepts and methods

Our new objective function is: $= f_L(w, \alpha)$

$$\textcircled{2} \quad \begin{cases} (\hat{w}, \hat{\alpha}) = \underset{w,\alpha}{\operatorname{argmin}} \frac{1}{2} w^T w - \sum_{n=1}^{N} \alpha_n \left(t_n (w^T x_n + b) - 1 \right) \\ \text{subject to } \alpha_n \geq 0 \text{ for all } n \end{cases}$$

↓

here, we have N points in the learning data set

(The idea of the Lagrange multiplier technique is that a sol. to problem ① is a solution to problem ② - and that we apply the reverse implication without proof.)

We look for a critical point of our new objective function:

$$\begin{cases} \nabla_w f_L = w - \sum_{n=1}^{N} \alpha_n t_n x_n \\ \frac{\partial f_L}{\partial b} = - \sum_{n=1}^{N} \alpha_n t_n \end{cases}$$

At a critical point: $\begin{cases} w = \sum_{n=1}^{N} \alpha_n t_n x_n \\ \sum_{n=1}^{N} \alpha_n t_n = 0 \end{cases}$

If we compute the value of f_L at a critical point, we get:

$$\frac{1}{2} w^T w - \sum_{n=1}^{N} \alpha_n \left(t_n (w^T x_n + b) - 1 \right) =$$

$$\frac{1}{2} \left(\sum_{n=1}^{N} \alpha_n t_n x_n \right)^T \left(\sum_{n=1}^{N} \alpha_n t_n x_n \right) - \sum_{n=1}^{N} \alpha_n \left(t_n \left(\sum_{m=1}^{N} \alpha_m t_m x_m^T x_n + b \right) - 1 \right) =$$

$$\dots = \frac{1}{2}\sum_{n,m=1}^{N} \alpha_n \alpha_m t_n t_m x_n^T x_m - \sum_{n,m=1}^{N} \alpha_n \alpha_m t_n t_m x_n^T x_m - \sum_{n=1}^{N} \alpha_n t_n b + \sum_{n=1}^{N} \alpha_n$$

(as $\sum_{n=1}^{N} \alpha_n t_n = 0$)

$$= \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n,m=1}^{N} \alpha_n \alpha_m t_n t_m x_n^T x_m$$

There is a continuum of critical points. We want to find, among the critical points, the one minimizing the above expression:

$$\begin{cases} \lambda = \arg\min_{\alpha} \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n,m=1}^{N} \alpha_n \alpha_m t_n t_m x_n^T x_m \\ \text{under : } \alpha_n \geq 0, \ \sum_{n=1}^{N} \alpha_n t_n = 0 \end{cases}$$

(this is known as the dual optimization problem.)

There is no explicit solution for λ but, as the objective function is quadratic, this new problem is relatively easy to solve numerically.

3) Making predictions

$$t_{new} = \text{sign}\left(\sum_{n=1}^{N} \alpha_n t_n x_n^T x_{new} + b\right)$$

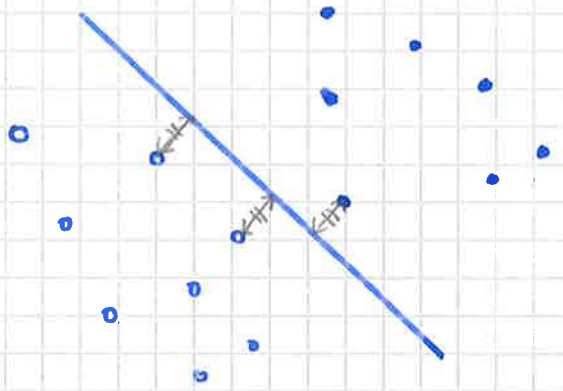
To find b, we use the fact that for the closest point x_n:

$$t_n(w^T x_n + b) = 1$$

and using that $t_n = 1/t_n$, $w = \sum_{n=1}^{N} \alpha_n t_n x_n$

$$b = t_n - \sum_{m=1}^{N} \alpha_m t_m x_m^T x_n \quad (b)$$

4) Support vectors

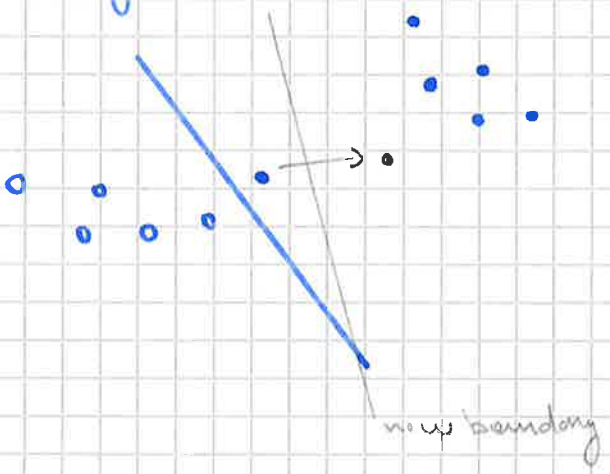


The set of points closest to the maximum margin decision boundary are known collectively as the support vectors.

* the decision is function of only a small subset of the training example (we say we have a sparse solution). Consider classifying a test point using KNN when the training data consists of several thousand objects → it could take a lot of time

(so the classifying task is quick)

* Drawback: moving a single training point might have a huge effect on the boundary



this method is implementable only if there is a boundary separating the two populations (this is called a hard margins SVM).

5) Soft margins

We want to allow points to potentially lie on the wrong side of the boundary. To achieve this, the constraint becomes $t_n(w^T x_n + b) \geq 1 - \xi_n$

$(0 \leq \xi_n \leq 1)$

New optimization task:

$$\begin{cases} \underset{w}{\text{argmin}} \ \frac{1}{2} w^T w + C \sum_{n=1}^{N} \xi_n \\ \text{subject to } \xi_n \geq 0 \text{ and } t_n(w^T x_n + b) \geq 1 - \xi_n \text{ for all } n \end{cases}$$

the new parameter C controls to what extent we are willing to allow points to sit within the margin band or on the wrong side of the decision boundary.

We now need to find the maximum of the following quadratic programming problem:

$$\begin{cases} \underset{\alpha}{\text{argmax}} \ \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} \alpha_n \alpha_m t_n t_m x_n^T x_m \\ \text{subject to } \sum_{n=1}^{N} \alpha_n t_n = 0 \text{ and } 0 \leq \alpha_n \leq C \text{ for all } n \end{cases}$$

The constant C needs to be fixed. We can set this using cross-validation.

To compute b, we need to find the support vector with the highest value of $w^T x_n$ (or $\sum_m \alpha_m t_m x_m^T x_n$) and compute b from equation (b)

TP: p. 56-60 of the O'Reilly book