Notes on MVA

Thomas Gadfort

July 16, 2017

1 Classification

A classification algorithm (classifier) is represented by a decision function $f(\vec{x}) : V \to \{-1, 1\}$ such that $f(\vec{x}) = 1$ if the classifier assigns $\vec{x}$ to the first class, and $f(\vec{x}) = -1$ if the classifier assigns $\vec{x}$ to the second class.

2 Linear Model

A model with a formula of the form

$$y_j = \sigma(\vec{\alpha} \cdot \vec{x}_j + \alpha_0) \tag{1}$$

where y_j is the j^{th} entry of the dependent variable, $\vec{x}_j$ is the j^{th} set of features (variables), and $\vec{\alpha}$ are the fit parameters (α_0 is a constant offset). The fit parameters are determined by maximizing the following objective function or loss function, L. That is, determine the set of $\vec{\alpha}$ that satisfy $\partial L / \partial \vec{\alpha} = 0$.

$$L = [y - (\vec{\alpha} \cdot \vec{x} + \alpha_0)]^2 \tag{2}$$

3 Generalized Linear Model

3.1 Solution to Logistic Regression

The maximum likelihood (L) solution for $\vec{\alpha}$ maximizes the following product over all training data,

$$L(\vec{y} \mid \mathbf{X}; \vec{\alpha}) \equiv \left[\prod_j^N \mathcal{P}(\vec{x}_j; \vec{\beta}) \right] \times \left[\prod_j^N (1 - \mathcal{P}(\vec{x}_j; \vec{\beta})) \right] \tag{3}$$

That is, the product of all observations assuming a true event, i.e., $\mathcal{P}(\vec{x}_j; \vec{\beta})$ for event j and the product of all observations assuming a false event, i.e., $1 - \mathcal{P}(\vec{x}_j; \vec{\beta})$. In this equation, $\vec{y}$ is the vector of outcomes, $\mathbf{X}$ is the matrix of features (i.e., $\vec{x}_j$ for outcome each outcome y_j).

Mathematically it is simpler to maximize the log-likelihood function (LL or $\mathcal{L} \equiv \mathcal{L}(\vec{y} \mid \mathbf{X}; \vec{\alpha})$) given as,

$$\mathcal{L}(\vec{y} \mid \mathbf{X}; \vec{\alpha}) = \left[\sum_j^N \log(\mathcal{P}(\vec{x}_j; \vec{\beta}))\right] \times \left[\sum_j^N \log(1 - \mathcal{P}(\vec{x}_j; \vec{\beta}))\right] \tag{4}$$

The solution for the model parameters is found by solving $d\mathcal{L}/d\vec{\alpha} = 0$. The partial derivative or gradient is defined as,

$$\frac{\partial \mathcal{L}}{\partial \vec{\alpha}} = \nabla_{\vec{\alpha}} \mathcal{L} = \sum_j^N \frac{1}{\mathcal{P}(\vec{x}_j; \vec{\beta})} \mathcal{P}'(\vec{x}_j; \vec{\beta}) - \sum_j^N \frac{1}{1 - \mathcal{P}(\vec{x}; \vec{\beta})} \mathcal{P}'(\vec{x}_j; \vec{\beta}) \tag{5}$$

Using $\mathcal{P}'(\vec{x}; \vec{\beta}) = \mathcal{P}(\vec{x}; \vec{\beta})(1 - \mathcal{P}(\vec{x}; \vec{\beta}))$, we find,

$$\begin{aligned} \nabla_{\vec{\alpha}} \mathcal{L} &= \sum_j^N \frac{1}{\mathcal{P}(\vec{x}_j; \vec{\beta})} \mathcal{P}(\vec{x}_j; \vec{\beta})(1 - \mathcal{P}(\vec{x}_j; \vec{\beta})) - \sum_j^N \frac{1}{1 - \mathcal{P}(\vec{x}_j; \vec{\beta})} \mathcal{P}(\vec{x}_j; \vec{\beta})(1 - \mathcal{P}(\vec{x}_j; \vec{\beta})) \\ &= \sum_j^N (1 - \mathcal{P}(\vec{x}; \vec{\beta})) - \sum_j^N \mathcal{P}(\vec{x}; \vec{\beta}) \end{aligned}$$

Combining the two sums representing $y_j = 1$ and $y_j = 0$, and inserting $(y_j - y_j)$ into the numerator yields,

$$\nabla_{\vec{\alpha}} \mathcal{L} = \sum_j^N y_j (1 - \mathcal{P}(\vec{x}; \vec{\beta})) - (1 - y_j) \mathcal{P}(\vec{x}; \vec{\beta})$$

4 Derivation of Logistic Regression

We assume that the case of interest (or "true") is coded to **1**, and the alternative case (or "false") is coded to **0**.

The logistic regression model assumes that the log-odds of an observation y can be expressed as a linear function of the input variables $\vec{x}$. That is,

$$\log\left[\frac{\mathcal{P}(\vec{x}; \vec{\beta})}{1 - \mathcal{P}(\vec{x}; \vec{\beta})}\right] = \sum_i^N \alpha_i x_i = \vec{\alpha} \cdot \vec{x} \tag{6}$$

where $\mathcal{P}(\vec{x};\vec{\beta})$ is the probability of observing y_j given $\vec{x}_j$ and $\vec{\alpha}$. The left-hand-side of the equation is known as the *logit* of $\mathcal{P}(\vec{x};\vec{\beta})$. Exponentiating each side yields,

$$\begin{aligned} \frac{\mathcal{P}(\vec{x};\vec{\beta})}{1-\mathcal{P}(\vec{x};\vec{\beta})} &= \exp\left[\sum_{\mathrm{i}}^{\mathrm{N}} \alpha_{\mathrm{i}}\mathrm{x}_{\mathrm{i}}\right] && (7)\\ &= \prod_{i}^{N} \exp(\alpha_{\mathrm{i}}\mathrm{x}_{\mathrm{i}}) && (8) \end{aligned}$$

Inverting the *logit* equation gives,

$$\mathcal{P}(\vec{x};\vec{\beta}) = \left[\frac{e^z}{1+e^z}\right] \tag{9}$$

where,

$$z = \sum_{i}^{N} \alpha_i x_i = \vec{\alpha}\cdot\vec{x} \tag{10}$$

4.1 Derivation of $\mathcal{P}'(z)$

$\mathcal{P}'(z)$ is defined as,

$$\mathcal{P}(\vec{x}) = \left[\frac{e^z}{1+e^z}\right] \tag{11}$$

Taking the derivative yields,

$$\begin{aligned} \mathcal{P}'(\vec{x}) &= \left[\frac{e^z}{1+e^z}\right] + \left[\frac{-(e^z)^2}{(1+e^z)^2}\right] && (12)\\ &= \frac{e^z}{(1+e^z)^2} && (13)\\ &= \frac{e^z}{(1+e^z)}\cdot\frac{1}{(1+e^z)} && (14)\\ &= [\mathcal{P}(z)][1-\mathcal{P}(z)] && (15) \end{aligned}$$

5 `glmnet`: Lasso and Elastic-Net Regularized Generalized Linear Models

`glmnet` is a package that fits a generalized linear model via penalized maximum likelihood. In this model, the following is minimized loss function,

$$L = \frac{1}{N}\sum_{i}^{N} \omega_i \times \ell(y_i;\vec{\beta};\vec{x}_i) + \lambda\left[\frac{1}{2}(1-\alpha)||\vec{\beta}||_2^2 + \alpha||\vec{\beta}||_1\right] \tag{16}$$

where ω_i is the weight given to event i, $\vec{\beta}$ are the fit parameters, $\ell(y, \eta)$ is the negative log-likelihood contribution for event i and α is the elastic-net penalty that bridges the gap between the `lasso` technique where $\alpha = 1$ and the `ridge` technique where $\alpha = 0$. The tuning parameter λ controls the overall strength of the penalty.

5.1 Gaussian Family

The gaussian family is identical the linear regression with the additional penalty term. The objective function for this family is then written as

$$L = -\left[\frac{1}{N}\sum_i^N (y_i - (\vec{\beta}\cdot\vec{x}_i))^2\right] + \lambda\left[\frac{1}{2}(1-\alpha)||\vec{\beta}||_2^2 + \alpha||\vec{\beta}||_1\right] \tag{17}$$

5.2 Binomial Family

The response variable for the binomial model takes the form

$$\mathcal{P}(\vec{x};\vec{\beta}) = \frac{e^{\vec{\beta}\cdot\vec{x}}}{1+e^{\vec{\beta}\cdot\vec{x}}} \tag{18}$$

The objective function for this family is then written as

$$L = -\left[\frac{1}{N}\sum_i^N y_i(\vec{\beta}\cdot\vec{x}_i) - \log(1+e^{\vec{\beta}\cdot\vec{x}_i})\right] + \lambda\left[\frac{1}{2}(1-\alpha)||\vec{\beta}||_2^2 + \alpha||\vec{\beta}||_1\right] \tag{19}$$

5.3 Poisson Family

The objective function for this family is then written as

$$L = -\left[\frac{1}{N}\sum_i^N y_i(\vec{\beta}\cdot\vec{x}_i) - e^{\vec{\beta}\cdot\vec{x}_i}\right] + \lambda\left[\frac{1}{2}(1-\alpha)||\vec{\beta}||_2^2 + \alpha||\vec{\beta}||_1\right] \tag{20}$$

5.4 Cox Model

The Cox proportional hazards model is commonly used for the study of the relationship between predictor variables and survival time. In the usual survival analysis framework, we have data of the form (y_1, x_1, δ_1), ..., (y_1, x_1, δ_1) where y_i, the observed time, is a time of failure if δ_i is 1 or right-censoring if δ_i is 0. We also let $t_1 < t_2 < ... < t_m$ be the increasing list of unique failure times, and $j(i)$ denote the index of the observation failing at time t_i.

The Cox model assumes a semi-parametric form for the hazard

$$h_i(t) = h_0(t)e^{\vec{\beta}\cdot\vec{x}_i} \tag{21}$$

where $h_i(t)$ is the hazard for patient i at time t, $h_0(t)$ is a shared baseline hazard, and $\vec{\beta}$ is a fixed vector of length p. In the classic setting $n \geq p$, inference is made via the partial likelihood,

$$L(\vec{\beta}) = \prod_{i}^{m} \frac{e^{\vec{\beta}\cdot\vec{x}_i}}{\sum_j^{R_i} e^{\vec{\beta}\cdot\vec{x}_j}} \tag{22}$$

where R_i is the set of indices j with $y_j \geq t_i$ (i.e., those at risk at time t_i).

6 `ranger`: A Fast Implementation of Random Forests

Use this...

7 `gbm`: Generalized Boosted Regression Models

8 `earth`: Multivariate Adaptive Regression Splines (MARS)

MARS builds models with an estimator of the form

$$f(\vec{x}) = \sum_{i}^{K} c_i B_i(\vec{x}) \tag{23}$$

where $B_i(\vec{x})$ is the i^{th} basis function of the input features $\vec{x}$ and c_i is the weight for that basis function. Each basis function takes one of the following three forms:

- Constant 1
- A hinge function. A hinge function has the form $\max(0, x - c)$ or $\max(0, c - x)$.
- Product of hinge functions.

9 `RSNNS`: Neural Networks in R using the Stuttgart Neural Network Simulator (SNNS)

Doesn't work with MLR yet.

10 Support Vector Machines (SVM)

Start with a set of N points for the feature set $\vec{x}$ for each outcomes y_j. The goal is to find a hyperplane that separates (maximally divides) the points belonging to $y_j = 1$ (TRUE) and

$y_j = -1$ (FALSE). Generally, a hyperplane is defined as

$$\vec{w} \cdot \vec{x} - b = 0 \tag{24}$$

where w is the normal vector and the parameter $b/||\vec{w}||$ is the offset from the origin along the w vector.

Since we can arbitrary scale parameters $\vec{w}, b$ defining fixed hyperplane π, we can choose $\vec{w}, b$ such that $||\vec{w}|| = 1$. Note that this pair of parameters is unique for any hyperplane. The distance $\rho(\vec{x}, \pi)$ between a $\vec{x}$ and a hyperplane $\pi(\vec{w}, b)$ can be calculated according to the following equation:

$$\rho(\vec{x}, \pi) = \frac{\langle \vec{w}, \vec{x} \rangle}{||\vec{w}||} \tag{25}$$

Two classes are called linearly separable if there exists at least one hyperplane that separates them. If hyperplane $\rho(\vec{x}, \pi)$ separates classes C_1 and C_2, then decision function

$$\begin{aligned} f(\vec{x}) = \text{sign}\{\langle \vec{w}, \vec{x} \rangle + b\} \quad &= 1 \quad \text{if } \langle \vec{w}, \vec{x} \rangle + b > 0 & (26) \\ &= -1 \quad \text{if } \langle \vec{w}, \vec{x} \rangle + b < 0 & (27) \end{aligned}$$

gives us a classifier that correctly classifies all vectors from C_1 and C_2.

For a hyperplane π separating classes C_1 and C_2, its margin $m(\pi, C_1, C_2)$ is defined as the distance between π and class C_1, plus the distance between π and class C_2.

$$m(\pi, C_1, C_2) = \rho(\pi, C_1) + \rho(\pi, C_2) \tag{28}$$

The objective function for an SVM is given by

$$\begin{aligned} L = \frac{1}{2} \sum_k^n w_k^2 + C \sum_j^l \chi_j, \quad \text{such that} & \quad (29) \\ y_j \left(\sum_k^n w_k x_{jk} + b \right) \geq 1 - \chi_j & \quad (30) \\ \chi_j \geq 0 & \quad (31) \end{aligned}$$

This is known as the soft max SVM solution with tunable parameter C.

The goal of the SVM is to minimize $||\vec{w}||$ subject to the criterion,

$$y_j(\vec{w} \cdot \vec{x}_j - b) \geq 1 \tag{32}$$

10.1 Kernals

Kernels are the idea of summing functions that imitate similarity (induce a positive-definite encoding of nearness) and support vector machines are the idea of solving a clever dual problem to maximize a quantity called margin. We can now restate what a support vector machine is: it is a method for picking data weights so that the modeling function for a given kernel has a maximum margin.

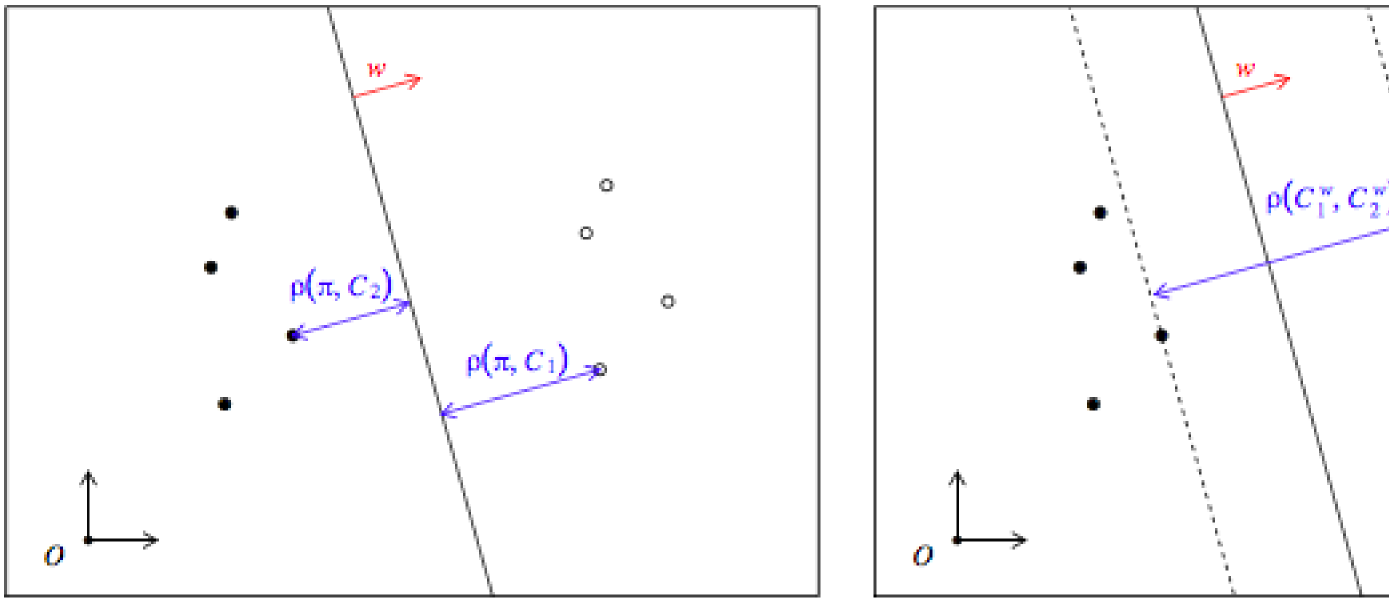

Figure 1: default

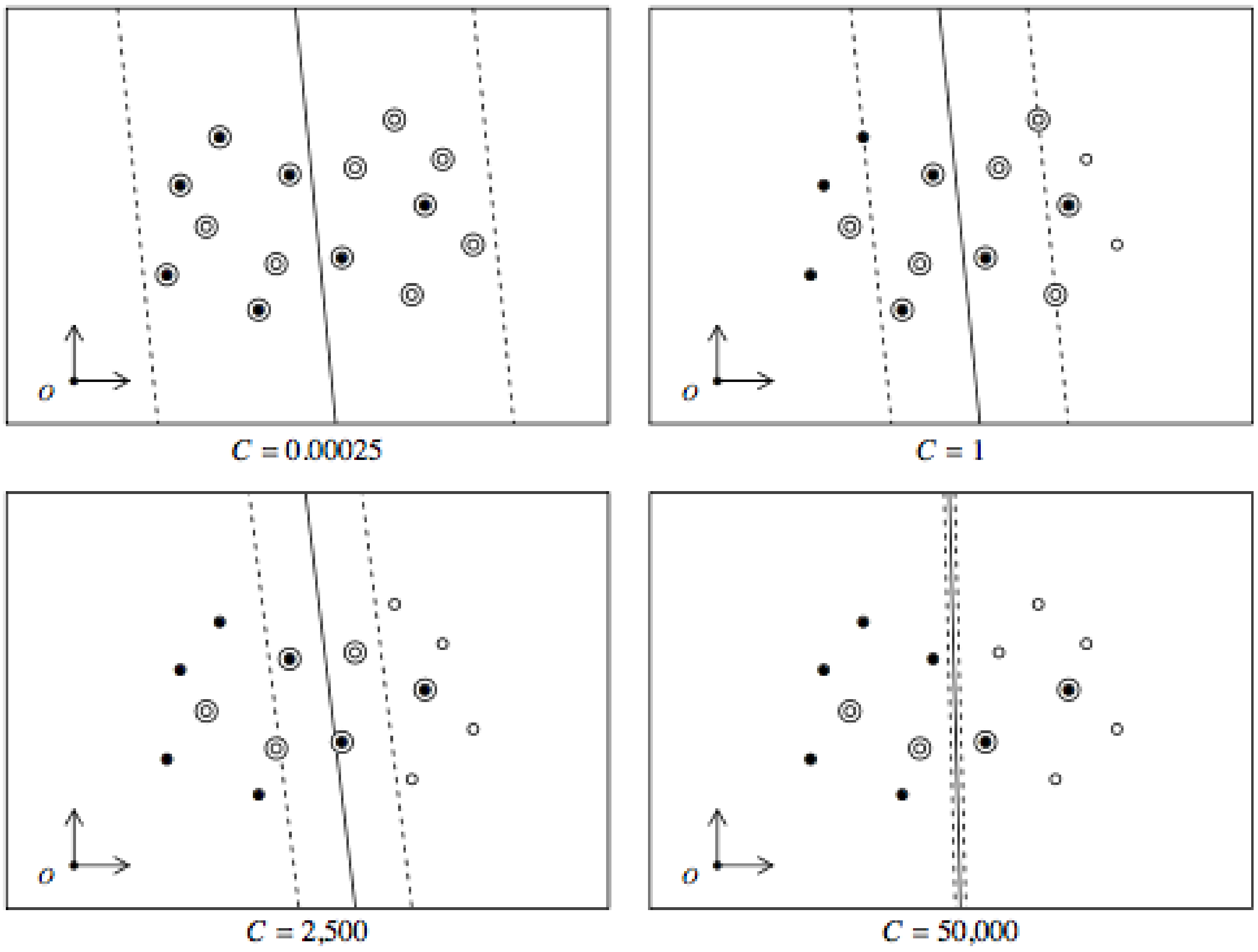

Figure 2: default