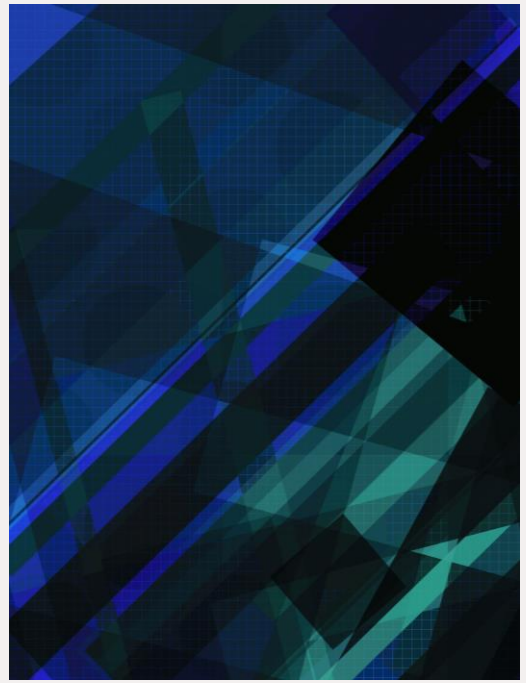
Linear methods for Classification 4.4-4.5

ALEKSANDRA SIEPIELA

KWIECIEŃ 2022



Spis treści prezentacji

Regresja logistyczna

Postać modelu regresji logistycznej

$$
\begin{aligned}
\log \frac{\Pr(G=1|X=x)}{\Pr(G=K|X=x)} &= \beta_{10} + \beta_1^T x \\
\log \frac{\Pr(G=2|X=x)}{\Pr(G=K|X=x)} &= \beta_{20} + \beta_2^T x \\
&\vdots \\
\log \frac{\Pr(G=K-1|X=x)}{\Pr(G=K|X=x)} &= \beta_{(K-1)0} + \beta_{K-1}^T x.
\end{aligned}
\tag{4.17}
$$

$$\begin{aligned}
\Pr(G=k|X=x) &= \frac{\exp(\beta_{k0}+\beta_k^T x)}{1+\sum_{\ell=1}^{K-1}\exp(\beta_{\ell 0}+\beta_\ell^T x)}, \; k=1,\ldots,K-1, \\
\Pr(G=K|X=x) &= \frac{1}{1+\sum_{\ell=1}^{K-1}\exp(\beta_{\ell 0}+\beta_\ell^T x)}, \qquad (4.18)
\end{aligned}$$

By podkreślić zależność od całego zbioru parametrów $\theta = \{\beta_{10}, \beta_1^T, ..., \beta_{(K-1)0}, \beta_{K-1}^T\}$, powyższe prawdopodobieństwa będziemy oznaczać jako:

$$Pr(G=k|X=x) = p_k(x;\theta)$$

Dopasowywanie modeli regresji logistycznej

Model regresji logistycznej jest zazwyczaj dopasowywany przy pomocy metody największej wiarygodności, wykorzystując prawdopodobieństwo warunkowe G przy danym X. Logarytm funkcji wiarygodności dla N obserwacji możemy zapisać następująco:

$$\ell(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta), \qquad (4.19)$$

$$p_k(x_i; \theta) = \Pr(G = k | X = x_i; \theta).$$

Przypadek dwóch klas

By ułatwić sobie rozumowanie rozważmy przypadek dwóch klas. Wówczas możemy przyjąć, że $y_i \in \{0, 1\}$, gdzie jeśli $y_i = 1$, to $g_i = 1$, a gdy $y_i = 0$, to $g_i = 2$. Niech $p_1(x, \theta) = p(x, \theta)$ i $p_2(x, \theta) = 1 - p(x, \theta)$. Wówczas funkcja wiarygodności (po nałożeniu logarytmu) ma następującą postać:

$$\begin{aligned} \ell(\beta) &= \sum_{i=1}^{N} \Big\{ y_i \log p(x_i; \beta) + (1 - y_i) \log(1 - p(x_i; \beta)) \Big\} \\ &= \sum_{i=1}^{N} \Big\{ y_i \beta^T x_i - \log(1 + e^{\beta^T x_i}) \Big\} . \end{aligned} \tag{4.20}$$

W tym wypadku $\beta = \{\beta_{10}, \beta_1\}$, a wektor wejść x_i zawiera stałą równą 1 odpowiadającą wyrazowi wolnemu (intercept).

Algorytm Newtona-Raphsona

Szukamy wartości pochodnej logarytmu funkcji wiarygodności w zerze:

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i(y_i - p(x_i;\beta)) = 0, \qquad (4.21)$$

By rozwiązać równanie 4.21 wykorzystujemy algorytm Newtona-Raphsona, który wymaga od nas policzenia drugiej pochodnej logarytmu funkcji wiarygodności lub hejsanu macierzy:

$$\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = -\sum_{i=1}^{N} x_i x_i{}^T p(x_i;\beta)(1 - p(x_i;\beta)). \qquad (4.22)$$

Algorytm Newtona-Raphsona

Zaczynając od β^{old}, pojedyncza aktualizacja β^{new} wygląda następująco:

$$\beta^{\text{new}} = \beta^{\text{old}} - \left(\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} \right)^{-1} \frac{\partial \ell(\beta)}{\partial \beta}, \qquad (4.23)$$

Uwaga: pochodne we wzorze dotyczą β^{old}

Notacja macierzowa

Możemy również przedstawić wzory z poprzednich slajdów w notacji macierzowej. Niech y będzie wektorem wartości y_i, X macierzą $N \times (p+1)$ o wartościach x_i, p wektorem dopasowanych prawdopodobieństw z $i-tym$ wyrazem $p(x_i;\beta^{old})$ i W macierzą diagonalną $N \times N$ wag z $i-tym$ elementem na diagonali równym $p(x_i;\beta^{old})(1-p(x_i;\beta^{old}))$. Wówczas możemy zapisać:

$$\frac{\partial \ell(\beta)}{\partial \beta} = \mathbf{X}^T(\mathbf{y}-\mathbf{p}) \qquad (4.24)$$

$$\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = -\mathbf{X}^T\mathbf{W}\mathbf{X} \qquad (4.25)$$

Krok Newtona wygląda następująco:

$$\begin{aligned}\beta^{\text{new}} &= \beta^{\text{old}} + (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y}-\mathbf{p}) \\ &= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\left(\mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y}-\mathbf{p})\right) \\ &= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{z}. \qquad (4.26)\end{aligned}$$

Algorytm IRLS

Algorytm ten jest określany jako iteratively reweighted least squares lub IRLS, ponieważ każda iteracja rozwiązuje problem ważonych najmniejszych kwadratów:

$$\beta^{\text{new}} \leftarrow \arg\min_{\beta}(\mathbf{z} - \mathbf{X}\beta)^T \mathbf{W}(\mathbf{z} - \mathbf{X}\beta). \qquad (4.28)$$

Wydaje się, że $\beta = 0$ jest dobrą wartością początkową procedury iteracyjnej, chociaż zbieżność nigdy nie jest zagwarantowana. Zazwyczaj algorytm jest zbieżny, ponieważ logarytm funkcji wiarygodności jest wklęsły.

W przypadku, gdy rozważamy wiele klas ($K \geqslant 3$) algorytm Newtona również można wyrazić jako IRLS, ale z wektorem $K - 1$ odpowiedzi i niediagonalną macierzą wag per obserwacja. Alternatywnie możemy również skorzystać z metody coordinate-decent, która również efektywnie maksymalizuje logarytm funkcji wiarygodności.

Aproksymacje kwadratowe i wnioskowanie

Estymatory największej wiarygodności $\hat{\beta}$ są współczynnikami w modelu dopasowanym metodą ważonych najmniejszych kwadratów, gdzie odpowiedzi wyrażane są następującym wzorem:

$$z_i = x_i^T \hat{\beta} + \frac{(y_i - \hat{p}_i)}{\hat{p}_i(1 - \hat{p}_i)}, \qquad (4.29)$$

Ponadto, wagi wyrażane są wzorem $w_i = \hat{p}_i(1 - \hat{p}_i)$. Zarówno z_i, jak i w_i zależą od estymatora $\hat{\beta}$.

Aproksymacje kwadratowe i wnioskowanie

Zauważmy również, że:

1. Ważona rezydualna suma kwadratów jest równa statystyce chi-kwadrat Pearsona i ma następującą postać:

$$\sum_{i=1}^{N} \frac{(y_i - \hat{p}_i)^2}{\hat{p}_i(1 - \hat{p}_i)}, \tag{4.30}$$

2. Jeśli model jest poprawny, to $\hat{\beta}$ zbiega do prawdziwego β;

3. Na mocy centralnego twierdzenia granicznego rozkład $\hat{\beta}$ zbiega do $N(\beta, (X^T W X)^{-1})$.

4.Budowa modelu może być kosztowna w przypadku modeli regresji logistycznej, ponieważ każde dopasowanie modelu wymaga iteracji. W przypadku regresji logistycznej możemy użyć testu Rao score i testu Walda. Żaden z nich nie wymaga iteracyjnego dopasowania modelu.

Powyższe własności znajdują swoje zastosowanie na przykład w R. Objekty GLM (generalized linear model) są traktowne jak modele liniowe i wówczas wszystkie narzędzia dostępne dla modeli liniowych mogą zostać zastosowane automatycznie.

Regresja logistyczna regularyzowana normą L1

Kara L_1 wykorzystywana przy metodzie Lasso, może być również wykorzystywana do selekcji i ściągania do zera zmiennych w dowolnym modelu liniowym. W przypadku regresji logistycznej będziemy maksymalizować wyrażenie 4.20, na które nałożona została kara L_1. Wyrażenie to ma następującą postać:

$$\max_{\beta_0,\beta} \left\{ \sum_{i=1}^{N} \left[y_i(\beta_0 + \beta^T x_i) - \log(1 + e^{\beta_0+\beta^T x_i}) \right] - \lambda \sum_{j=1}^{p} |\beta_j| \right\}. \qquad (4.31)$$

Podobnie jak w przypadku Lasso, zazwyczaj nie karzemy interceptu i standaryzujemy predyktory.

Regresja logistyczna regularyzowana normą L1

Wyrażenie 4.31 jest wklęsłe, zatem rozwiązanie możemy znaleźć przy pomocy różnych nieliniowych metod programowania. Alternatywnie możemy użyć tych samych kwadratowych aproksymacji, których użyliśmy przy algorytmie Newtona. Wówczas możemy rozwiązać 4.31 poprzez wielokrotna aplikację ważonego algorytmu lasso.

Co ciekawe, równania 4.24 dla zmiennych z niezerowymi współczynnikami mają następującą postać:

$$\mathbf{x}_j^T(\mathbf{y} - \mathbf{p}) = \lambda \cdot \text{sign}(\beta_j), \tag{4.32}$$

Regresja logistyczna regularyzowana normą L1

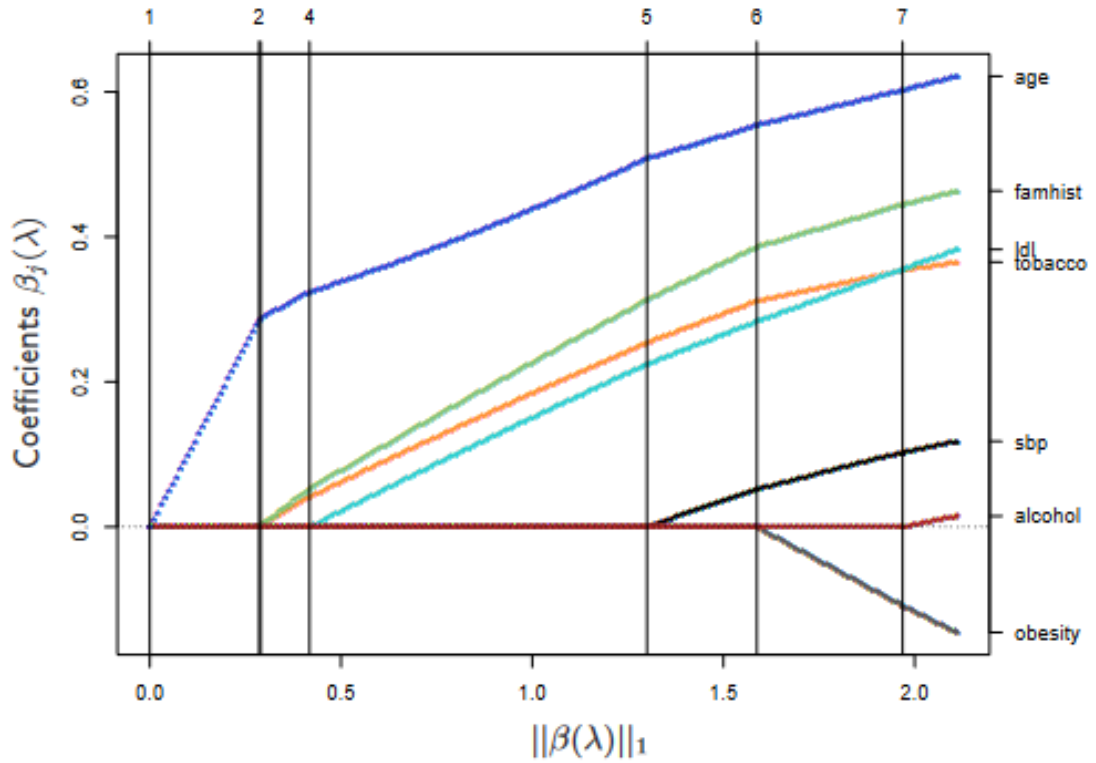


FIGURE 4.13. *L_1 regularized logistic regression coefficients for the South African heart disease data, plotted as a function of the L_1 norm. The variables were all standardized to have unit variance. The profiles are computed exactly at each of the plotted points.*

Hiperpłaszczyzny separujące

Hiperpłaszczyzny separujące

Na kolejnym etapie zajmiemy się klasyfikatorem wykorzystującym płaszczyzny separujące. Klasyfikator ten wyznacza liniowe granice decyzyjne, które w najlepszy możliwy sposób dzielą dane na klasy.

Obraz 4.14 pokazuje 20 punktów należących do dwóch klas w R^2. Dane te można rozdzielić przy pomocy linii. Na obrazku widzimy dwie niebieskie linie, które przedstawiają dwie spośród nieskończonej liczby hiperpłaszyzn separujących. Linia pomarańczowa została wyznaczona przy pomocy metody najmniejszych kwadratów. Widzimy, że rozwiązanie uzyskane przy pomocy metody najmniejszych kwadratów popełnia jeden błąd. Metoda LDA wyznacza taką samą granicę jak ta zaznaczona na rysunku kolorem pomarańczowym. Wynika to z faktu, że w przypadku, gdy rozważamy dwie klasy rozwiązanie uzyskane dla regresji liniowej i LDA jest takie samo.

Hiperpłaszczyzny separujące

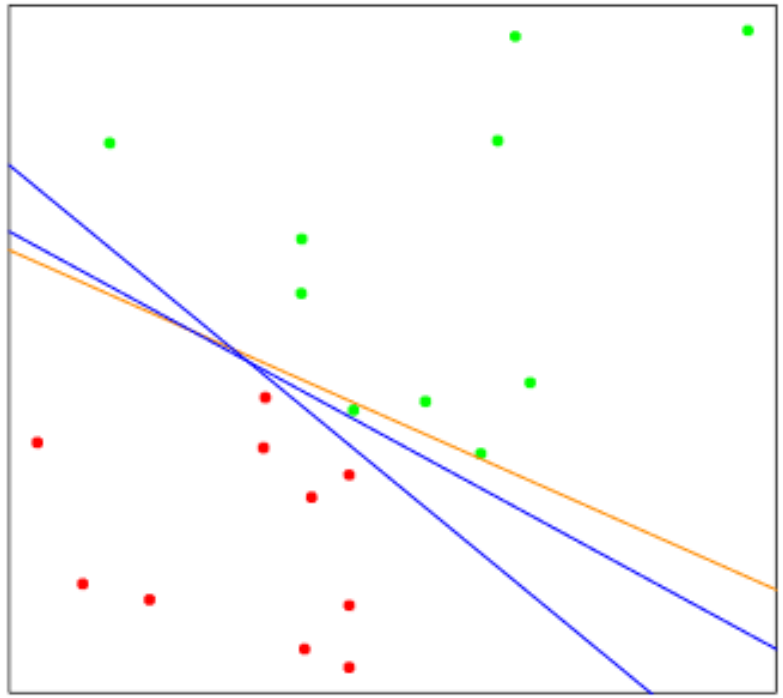


FIGURE 4.14. *A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the* perceptron learning algorithm *with different random starts.*

Perceptron Rosenblatta

Algorytm uczenia perceptron Rosenblatta stara się znaleźć hiperpłaszczyznę separującą poprzez minimalizację odległości źle sklasyfikowanych punktów od granicy decyzyjnej. Jeśli $y_i = 1$ jest źle sklasyfikowane, wówczas $x_i^T \beta + \beta_0 < 0$, a gdy $y_i = 1$, jest źle sklasyfikowane mamy: $x_i^T \beta + \beta_0 > 0$.

Naszym celem jest minimalizacja następującego wyrażenia:

$$D(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i (x_i^T \beta + \beta_0), \qquad (4.41)$$

gdzie $\mathcal{M}$ jest zbiorem indeksów źle sklasyfikowanych punktów.

Perceptron Rosenblatta

Gradienty względem β i β_0 (zakładając, że $\mathcal{M}$ jest ustalony) wyrażane są następującymi wzorami:

$$\partial \frac{D(\beta, \beta_0)}{\partial \beta} = -\sum_{i \in \mathcal{M}} y_i x_i, \tag{4.42}$$

$$\partial \frac{D(\beta, \beta_0)}{\partial \beta_0} = -\sum_{i \in \mathcal{M}} y_i. \tag{4.43}$$

Algorytm w rzeczywistości wykorzystuje procedurę stochastic gradient decent by zminimalizować to odcinkowo liniowe kryterium 4.41.

Perceptron Rosenblatta

Parametry modelu są natomiast aktualizowane w następujący sposób:

$$\begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} \leftarrow \begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} + \rho \begin{pmatrix} y_i x_i \\ y_i \end{pmatrix}. \tag{4.44}$$

gdzie ρ to współczynnik uczenia się, za który w tym przypadku można przyjąć 1 bez straty w ogólności. Ponadto warto zaznaczyć, że jeżeli klasy są liniowo separowalne, można wykazać, że algorytm zbiega do rozdzielającej hiperpłaszczyzny w skończonej liczbie kroków.

Perceptron Rosenblatta

Algorytm ten generuje jednak następujące problemy:

- gdy dane są separowalne, istnieje wiele rozwiązań i to, które zostanie znalezione zależy od wartości startowej;
- skończona liczba kroków (prowadzących do zbieżnosci algorytmu) może być bardzo duża;
- gdy dane nie są separowalne, algorytm nie jest zbieżny.

Dziękuję za uwagę