Linear methods of classification

Victor Kitov

v.v.kitov@yandex.ru

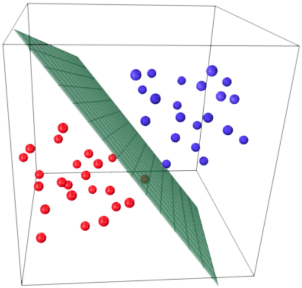


Table of Contents

Multiclass general classifier

- Define discriminant functions $g_c(x)$ for classes $c = \overline{1, C}$.
- General classifier:

$$\widehat{y}(x) = \arg\max_c g_c(x)$$

- Boundary between classes i and j:

$$\{x : g_i(x) = g_j(x)\}$$

- Margin (quality of classification):

$$M(x, y) = g_y(x) - \max_{c \neq y} g_c(x)$$

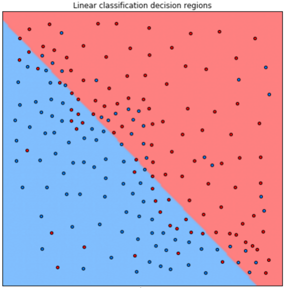
Linear classifier

- Include constant feature: $x = [1, x^2, x^3, ...x^D]$.
- Linear classifier:

$$g_c(x) = w_c^T x, \quad c = \overline{1, C}.$$

- Boundary between classes i and j is linear:

$$\left\{x : w_i^T x = w_j^T x\right\}$$



Binary general classifier

- $y \in \{+1, -1\}$. Discriminant functions: $g_{+1}(x), g_{-1}(x)$.
- Define $g(x) = g_{+1}(x) - g_{-1}(x)$ - preference for class $+1$.
- Binary classifier:

$$\widehat{y}(x) = \arg\max_{c \in \{+1,-1\}} g_c(x) = \text{sign}\left(g_{+1}(x) - g_{-1}(x)\right) = \text{sign}\left(g(x)\right)$$

- Margin:

$$M(x, y) = g_y(x) - g_{-y}(x) = y\left(g_{+1}(x) - g_{-1}(x)\right) = yg(x)$$

Binary linear classifier

- Discriminant functions: $w_{+1}^T x, w_{-1}^T x$.
- Define $w = w_{+1} - w_{-1}$.
- Binary linear classifier:

$$\widehat{y}(x) = \arg\max_{c \in \{+1,-1\}} w_c^T x = \text{sign}\left(w_{+1}^T x - w_{-1}^T x\right) = \text{sign}\left(w^T x\right)$$

- Margin:

$$M(x,y) = w_y^T x - w_{-y}^T x = y\left(w_{+1}^T x - w_{-1}^T x\right) = y w^T x = w^T x y$$

Table of Contents

Estimation of w

- Select w to minimize # of misclassifications:

$$\sum_{n=1}^{N} \mathbb{I}\left[w^T x_n y_n < 0\right] \to \min_w$$

- Piecewise constant criterion, gradient=0 almost everywhere.
  - can't use optimization methods to find minimum!
- Use decreasing $\mathcal{L}$ with non-trivial gradients:

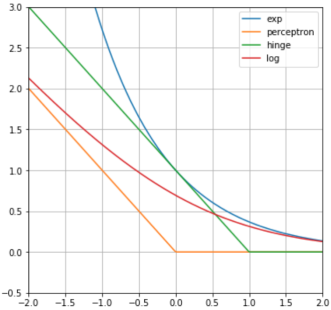
$$\sum_{n=1}^{N} \mathcal{L}\left(w^T x_n y_n\right) \to \min_w$$

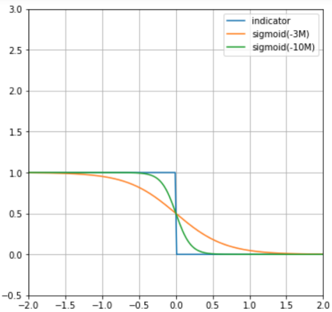
- For general binary classifier can use:

$$\sum_{n=1}^{N} \mathcal{L}\left(y_n g_w(x_n)\right) \to \min_w, \quad w\text{-parameters of } g(x)$$

Common loss functions

$$\mathcal{L}_{exp}(M) = e^{-M} \quad \mathcal{L}_{perceptron}(M) = [-M]_+$$
$$\mathcal{L}_{hinge}(M) = [1 - M]_+ \quad \mathcal{L}_{log}(M) = \ln\left(1 + e^{-M}\right)$$





Check properties: robustness to outliers, convexity, penalization for small positive margin.

Table of Contents

Regularization

- Insert additional requirement for regularizer $R(\beta)$ to be small:

$$\sum_{n=1}^{N}\mathcal{L}\left(M(x_n,y_n|w\right)+\lambda R(w)\to\min_{\beta}$$

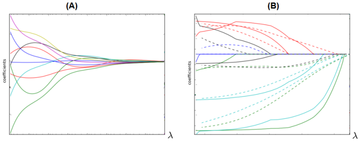
- $\lambda>0$ - hyperparameter.
- $R(w)$ penalizes complexity of models.

$R(\beta)=\|\|w\|\|_1$	L_1 regularization
$R(\beta)=\|\|w\|\|_2^2$	L_2 regularization
$R(\beta)=\alpha\left\|w\right\|_1+(1-\alpha)\left\|w\right\|_2^2$	combination with $\alpha\in(0,1)$

- Not only *accuracy* matters for the solution but also *model simplicity*!
- λ controls complexity of the model: $\uparrow\lambda\Leftrightarrow$ complexity$\downarrow$.

Comments

- Dependency of β from λ for L_2 (A) and L_1 (B) regularization:



- L_2 can be used for automatic feature selection.
- λ is usually found using cross-validation on exponential grid, e.g. $[10^{-6}, 10^{-5}, ...10^{5}, 10^{6}]$.
- It's always recommended to use regularization because
  - it gives smooth control over model complexity.
  - reduces ambiguity for multiple solutions case.

Different account for different features

- Traditional approach regularizes all features uniformly:

$$\sum_{n=1}^{N} \mathcal{L}\left(M(x_n, y_n|w)\right) + \lambda R(w) \to \min_{w}$$

- Suppose we have K groups of features with indices:

$$I_1, I_2, ... I_K$$

- We may control the impact of each feature group by minimizing:

$$\sum_{n=1}^{N} \mathcal{L}\left(M(x_n, y_n|w)\right) + \lambda_1 R(\{w_i | i \in I_1\}) + ... + \lambda_K R(\{w_i | i \in I_K\})$$

- $\lambda_1, \lambda_2, ... \lambda_K$ can be set using cross-validation
- In practice use common regularizer but with different feature scaling.

Table of Contents

Orthogonal vector to hyperplane

Theorem 1

Vector w is orthogonal to hyperplane $w^T x + w_0 = 0$

Proof. Consider arbitrary $x_A, x_B \in \{x: \; w^T x + w_0 = 0\}$:

$$w^T x_A + w_0 = 0 \quad (1)$$

$$w^T x_B + w_0 = 0 \quad (2)$$

By substracting (2) from (1), obtain $w^T(x_A - x_B) = 0$, so w is orthogonal to hyperplane. $\square$

Distance from point to hyperplane

Theorem 2

Distance from point x to hyperplane $w^T x + w_0 = 0$ is equal to $\frac{w^T x + w_0}{\|w_0\|}$.

Proof. Project x on the hyperplane, let the projection be p and complement $h = x - p$, orthogonal to hyperplane. Then

$$x = p + h$$

Since p lies on the hyperplane,

$$w^T p + w_0 = 0$$

Since h is orthogonal to hyperplane and according to theorem 1

$$h = r\frac{w}{\|w\|},\ r \in \mathbb{R} \text{ - distance to hyperplane.}$$

Distance from point to hyperplane

$$x = p + r\frac{w}{\|w\|}$$

After multiplication by w and addition of w_0:

$$w^T x + w_0 = w^T p + w_0 + r\frac{w^T w}{\|w\|} = r\|w\|$$

because $w^T p + w_0 = 0$ and $\|w\| = \sqrt{w^T w}$. So we get, that

$$r = \frac{w^T x + w_0}{\|w\|}$$

Comments:

- From one side of hyperplane $r > 0 \Leftrightarrow w^T x + w_0 > 0$
- From the other side $r < 0 \Leftrightarrow w^T x + w_0 < 0$.
- Distance from hyperplane to origin 0 is $\frac{w_0}{\|w\|}$. So w_0 accounts for hyperplane offset.

Binary linear classifier geometric interpretation

Binary linear classifier:

$$\widehat{y}(x) = \text{sign}\left(w^T x + w_0\right)$$

divides feature space by hyperplane $w^T x + w_0 = 0$.

- Confidence of decision is proportional to distance to hyperplane $\frac{|w^T x + w_0|}{\|w\|}$.
- $w^T x + w_0$ is the confidence that class is positive.

Table of Contents

Binary classification

- Linear classifier:

$$score(y=1|x)=w^{T}x$$

- +relationship between score and class probability is assumed:

$$p(y=1|x)=\sigma(w^{T}x)$$

where $\sigma(z)=\frac{1}{1+e^{-z}}$ - sigmoid function

Binary classification: estimation

Using the property $1-\sigma(z)=\sigma(-z)$ obtain that

$$p(y=+1|x)=\sigma(w^T x)\Longrightarrow p(y=-1|x)=\sigma(-w^T x)$$

So for $y\in\{+1,-1\}$

$$p(y|x)=\sigma(y\langle w,x\rangle)$$

Therefore ML estimation can be written as:

$$\prod_{i=1}^{N}\sigma(\langle w,x_i\rangle y_i)\to\max_w$$

Loss function for 2-class logistic regression

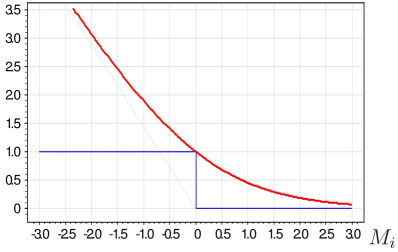
For binary classification $p(y|x) = \sigma(\langle w, x\rangle y)$ $w = [\beta_0', \beta]$, $x = [1, x_1, x_2, ...x_D]$.

Estimation with ML:

$$\prod_{i=1}^{N} \sigma(\langle w, x_i\rangle y_i) \to \max_w$$

which is equivalent to

$$\sum_{i=1}^{N} \ln(1 + e^{-\langle w, x_i\rangle y_i}) \to \min_w$$



It follows that logistic regression is linear discriminant estimated with loss function $\mathcal{L}(M) = \ln(1 + e^{-M})$.

Multiple classes

Multiple class classification:

$$\begin{cases} score(\omega_1|x) = w_1^T x \\ score(\omega_2|x) = w_2^T x \\ \cdots \\ score(\omega_C|x) = w_C^T x \end{cases}$$

+relationship between score and class probability is assumed:

$$p(\omega_c|x) = softmax(w_c^T x|x_1^T x, ...x_C^T x) = \frac{exp(w_c^T x)}{\sum_i exp(w_i^T x)}$$

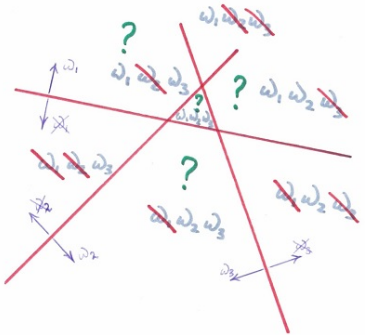
Table of Contents

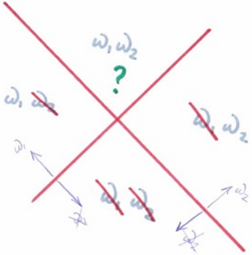
Multiclass classification with binary classifiers

- Task - make C-class classification using many binary classifiers.
- Approaches:
  - **one-versus-all**
    - for each $c = 1, 2, \dots C$ train binary classifier on all objects and output $\mathbb{I}[y_n = c]$,
    - assign class, getting the highest score in resulting C classifiers.
  - **one-versus-one**
    - for each $i, j \in [1, 2, \dots C]$, $i \neq j$ learn on objects with $y_n \in \{i, j\}$ with output y_n
    - assign class, getting the highest score in resulting $C(C-1)/2$ classifiers.
  - **error correcting codes**

One versus all - ambiguity

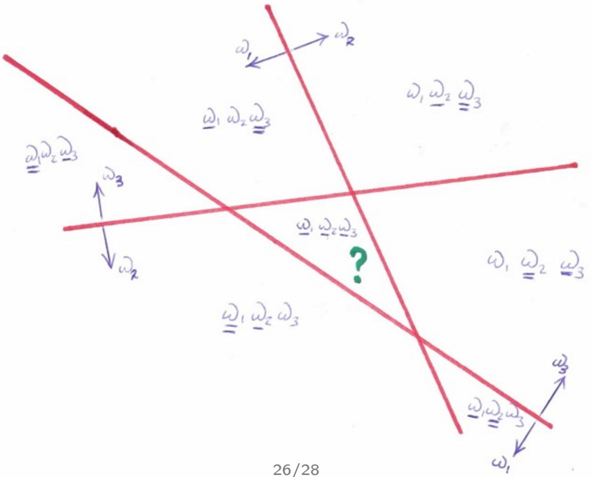
Classification among three classes: $\omega_1, \omega_2, \omega_3$





One versus one - ambiguity

Classification among three classes $\omega_1, \omega_2, \omega_3$ depending only on halfspace may be ambiguous:



Error correcting codes

- Used in classification
- Each class ω_i is coded as a binary codeword W_i consisting of B bits:

$$\text{class } i \to W_i$$

- Minimum sufficient amount of bits to code C classes is $\lceil \log_2 C \rceil$
- Given x, B binary classifiers predict each bit of the class codeword.
- Class is predicted as

$$\hat{c}(x) = \arg\min_c \sum_{b=1}^{B} |W_{cb} - \widehat{p}_b(x)|$$

- where W_{cb} is the b-th bit of codeword, corresponding to class c.
- More bits are used to make classification more robust to errors of individual binary classifiers.
- Codewords are selected to have maximum mutual Hamming distance or randomly.

Summary

- Linear classifier - classifier with linear discriminant functions.
- Binary linear classifier: $\widehat{y}(x) = \text{sign}(w^T x + w_0)$.
- Perceptron, logistic, SVM - linear classifiers estimated with different loss functions.
- Weights are selected to minimize total loss on margins.
- Regularization gives smooth control over model complexity.
- L_1 regularization automatically selects features.