Machine Learning and Neural Networks (MATH3431) Epiphany term, 2024

Handout 10: Multi-class classification

Lecturer & author: Georgios P. Karagiannis georgios.karagiannis@durham.ac.uk

Aim. To introduce the Support Vector Machines as a procedure. Motivation, set-up, description, computation, and implementation. We focus on the classical treatment.

Reading list & references:

(1) Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.
 - Ch. 17 (pp. 190-198)

(2) Bishop, C. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: Springer.
 - Ch. 7.1 Sparse Kernel Machines/Maximum marginal classifiers

(3) Vapnik, V. (2013). The nature of statistical learning theory. Springer science & business media.

(4) Boyd, S. P., & Vandenberghe, L. (2004). Convex optimization. Cambridge university press.

(5) Strang, G. (2019). Linear algebra and learning from data. Wellesley-Cambridge Press.

1. Intro and motivation

Note 1. Multi-class categorization is the machine learning problem of classifying instances into one of several possible target classes.

Example 2. Applications include: (i.) categorizing documents according to topic (input features is the set of documents and target is the set of possible topics); (ii.) determining which object appears in a given image (input is the set of images and target is the set of possible objects).

Note 3. We wish to classify objects with features $x \in \mathcal{X}$ into one of the $k \in \mathbb{N} - \{0\}$ categories $\{\mathcal{C}_1, ..., \mathcal{C}_k\}$.

Note 4. To achieve that we aim to learn a predictive rule $h : \mathcal{X} \to \mathcal{Y}$ where $h \in \mathcal{H}$. Here, to easy the notation we let $\mathcal{Y} = \{1, ..., k\}$ implying that the multi-class categorization involves $k \in \mathbb{N} - \{0\}$ possible target classes . If the object $z = (x, y)$ belongs to class $\mathcal{C}_j$ i.e. $z \in \mathcal{C}_j$ then $y = j$.

Note 5. There is available a training data-set $\mathcal{S} = \{z_i = (x_i, y_i)\,; i = 1, ..., n\}$ of n examples where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ independently generated from a data-generating process g.

2. Reduction to the binary classification approaches

Note 6. The multi-class categorization problem can be addressed by a reduction to the binary classification. –We discuss two simple ways.

Note 7. Assume that there is available a binary classification learning algorithm $\mathfrak{A}_{\text{binary}}$, such as the soft-SVM or the Bernoulli regression.

2.1. **One-versus-all approach.**

Note 8. The one-versus-all approach involves training $k > 2$ binary classifiers, each of which discriminates between one class and the rest of the classes.

Note 9. It works as follows

(1) For each class $j = 1, ..., k$, construct a binary training data-sets

$$\mathcal{S}^{[j]} = \left\{ \left(x_1, t_1^{[j]} \right), ..., \left(x_n, t_n^{[j]} \right) \right\}$$

where $t_i^{[j]} = \begin{cases} -1 & y_i \neq j \\ 1 & y_i = j \end{cases}$ from the original available training data-set $\mathcal{S} = \{ z_i = (x_i, y_i) ; i = 1, ..., n \}$. $\mathcal{S}^{[j]}$ is the set of examples labeled 1 if their label in $\mathcal{S}$ was j, and -1 otherwise.

(2) For each binary training data-set $\left\{ \mathcal{S}^{[j]} \right\}$, consider a binary predictor rule $h^{[j]} : \mathcal{X} \to \{-1, +1\}$ with $h^{[j]} \in \mathcal{H}^{[j]}$ and train it against $\mathcal{S}^{[j]}$ such that

$$h^{[j]} = \mathfrak{A}_{\text{binary}} \left(\mathcal{S}^{[j]} \right)$$

(3) Given $\left\{ h^{[j]} \right\}$, the one-versus-all multiclass hypothesis is defined as

$$h_{\text{ova}} (x) \in \arg\max_{j \in \mathcal{Y}} \left\{ h^{[j]} (x) \right\}$$

Note 10. In cases where there are ties, it is reasonable to decide the classification based on some reasonable weights. E.g. if algorithm $\mathfrak{A}_{\text{binary}}$ is a SVM with hypothesis of the form $h^{[j]} (x) = \text{sign} \left(\langle w^{[j]}, x \rangle \right)$, and I get $h^{[j]} (x) = 1 = h^{[j']} (x)$ with $\langle w^{[j]}, x \rangle > \left\langle w^{[j']}, x \right\rangle$ then $y = j$.

Note 11. In Note 9(**??**), we make the strong assumption that hoping that $h^{[j]}$ should equal 1 if and only if x belongs to class j. However, this is not always true. Figure 2.1 shows a case involving three classes where this approach leads to regions of input space that are ambiguously classified.

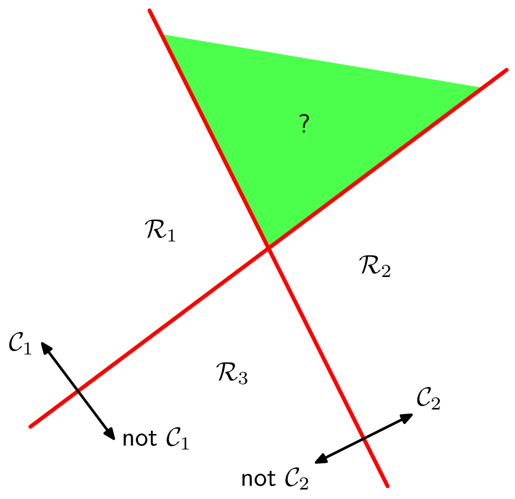

FIGURE 2.1. Two binary classification rules $h^{[1]}$ and $h^{[2]}$ designed to distinguish points in class $\mathcal{C}_j$ from points not in class $\mathcal{C}_j$ for $j \in \{1, 2\}$.

2.2. **One-versus-one approach.**

Note 12. The one-versus-one approach involves the introduction of $k(k-1)/2$ binary classification rules, one for every possible pair of classes, and their comparison to each other, in a manner that each point is classified according to a majority vote amongst these binary classification rules.

Note 13. It works as follows

(1) For each pair of classes (i, j) where $1 \le i < j \le k$, construct binary training data-sets $\mathcal{S}^{[i,j]}$ containing all examples from $\mathcal{S}$ whose label is either i or j.

(2) For each binary training data-set $\mathcal{S}^{[i,j]}$ set the binary label as 1 if the class is i and as -1 if the class is j; i.e. Starting with $\mathcal{S}^{[i,j]} = \emptyset$ for $t = 1, ..., m$
- if $y_t = i$ add $(x_t, 1)$ to $\mathcal{S}^{[i,j]}$
- if $y_t = j$ add $(x_t, -1)$ to $\mathcal{S}^{[i,j]}$

(3) For each pair (i, j) where $1 \le i < j \le k$, train a binary classification algorithm against $\mathcal{S}^{[i,j]}$ to get

$$h^{[i,j]} = \mathfrak{A}_{\text{binary}}\left(\mathcal{S}^{[i,j]}\right)$$

(4) Given $\left\{h^{[i,j]}\right\}$, the one-versus-one multiclass hypothesis is defined as the one with the highest number of "wins"; i.e.

$$h_{\text{ovo}}(x) \in \arg\max_{i \in \mathcal{Y}} \left\{ \sum_{j \in \mathcal{Y}} \text{sign}(j-1)\, h^{[i,j]}(x) \right\}$$

Note 14. The one-versus-one approach suffers from the problem of ambiguous regions as seen in Figure 2.2.

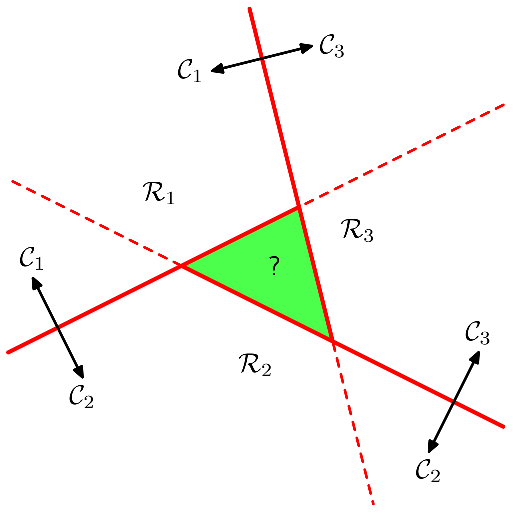

FIGURE 2.2. three binary classification rules each of which is used to separate a pair of classes $\mathcal{C}_k$ and $\mathcal{C}_j$.

3. LINEAR MULTI-CLASS PREDICTORS

3.1. **Setting up the predictive rule and class of hypotheses.**

Note 15. We try to generalize the linear predictor for binary classifications taking the form

$$\begin{aligned} h_{\text{bin}}(x) &= \text{sign}(\langle w, x\rangle) \\ &= \underset{y\in\{-1,+1\}}{\arg\min}\ \langle w, yx\rangle \end{aligned}$$

into linear predictor for multi-class classifications.

Note 16. In the multi-class setting, we extend the binary SVM idea as follows.

- Define a multi-class predictor rule $h : \mathcal{X} \to \mathcal{Y}$ with formula

$$h_w(x) = \underset{y\in\mathcal{Y}}{\arg\min}\,\langle w, \Psi(x, y)\rangle \tag{3.1}$$

  for $w \in \mathcal{W} \subset \mathbb{R}^d$.
- In (3.1), we need to specify a class-sensitive feature mapping $\Psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ whose function's $\Psi(x, y)$ role is to act as a score function that assesses how well the label y fits the instance x.
- The hypothesis class of multi-class predictors is

$$\mathcal{H}_{\Psi,\mathcal{W}} = \left\{ x \mapsto \underset{y\in\mathcal{Y}}{\arg\min}\,\langle w, \Psi(x, y)\rangle : w \in \mathcal{W} \right\}$$

Note 17. The equation of class-sensitive feature mapping Ψ depends on the application.

Example 18. An example of class-sensitive feature mapping Ψ such that

$$\Psi(x, y) = \left(\underbrace{0, \dots 0,}_{(y-1)n\text{ elements}} \quad \overbrace{x_1, \dots x_n,}^{n\text{ elements}} \quad \underbrace{0, \dots 0,}_{(k-y)n\text{ elements}} \right)^\top$$

where $\Psi(x, y) \in \mathbb{R}^{nk}$ and $w \in \mathcal{W} \subset \mathbb{R}^{nk}$. In that way, the predictive rule simplifies to

$$\begin{aligned} h_w(x) &= \underset{y\in\mathcal{Y}}{\arg\min}\,\langle w, \Psi(x, y)\rangle \\ &= \underset{y\in\mathcal{Y}}{\arg\min}\,\langle w_y, x\rangle \end{aligned}$$

where $w = (w_1, ..., w_k)^\top$ and each w_j has n elements.

3.2. About the learning problem.

Note 19. ($0-1$ loss function) The 0-1 loss function ℓ_{0-1} in the multi-class framework can be specified as

$$\ell_{0-1}(h_w, z = (x, y)) = 1_{\{y\}}(h_w(x)) = \begin{cases} 1, & h_w(x) = y \\ 0, & h_w(x) \neq y \end{cases} \tag{3.2}$$

as a straightforward extension of the one in binary classification.

Note 20. $0-1$ loss (3.2) is non-convex as the one in the binary classification problem.

Note 21. (Hinge loss function) The hinge loss function ℓ_{hinge} in the multi-class framework can be specified as

$$\ell_{\text{hinge}}\left(h_w, z=(x,y)\right) = \max_{\xi}\left(\ell_{0-1}\left(\xi, z=(x,y)\right) + \left\langle w, \Psi\left(x,\xi\right) - \Psi\left(x,y\right)\right\rangle\right) \tag{3.3}$$

extending on in binary classification. Example 23 provides a rational for its construction.

Solution. In the binary scenario, I have $\mathcal{Y}=\{-1,+1\}$. If I set $\Psi\left(x,y\right)=\frac{1}{2}yx$ then (3.3) reduces to $\max\left(0, 1-y\left\langle w,x\right\rangle\right)$.

Note 22. The hinge loss function (3.3) is convex w.r.t. w as a maximum over linear functions of w.

Example 23. The hinge loss function (3.3) upper bounds the $0-1$ loss function (3.2).

Solution. It is

$$\begin{aligned} \left\langle w, \Psi\left(x,y\right)\right\rangle &\le \left\langle w, \Psi\left(x, h_w\left(x\right)\right)\right\rangle \qquad \text{by construction} \\ \implies \ell_{0-1}\left(h_w, z=(x,y)\right) &\le \ell_{0-1}\left(h_w, z=(x,y)\right) + \left\langle w, \Psi\left(x, h_w\left(x\right)\right) - \Psi\left(x,y\right)\right\rangle \\ &\le \max_{\xi\in\mathcal{Y}}\left(\ell_{0-1}\left(\xi, z=(x,y)\right) + \left\langle w, \Psi\left(x,\xi\right) - \Psi\left(x,y\right)\right\rangle\right) \end{aligned}$$

Note 24. The hinge loss function ℓ_{hinge} can be used as a surrogate loss instead of the $0-1$ loss ℓ_{0-1} to address the non-convex learning problem with convex learning problem tools. This results from Note 22 and Example 23. Hence the computation of error bounds and the derivation of assumptions guarantee PAC learning algorithm is straightforward.

Example 25. The hinge loss function (3.3) reduces to the hinge loss $\ell_{\text{hinge}}\left(h, z=(x,y)\right) = \max\left(0, 1-y\left\langle w,x\right\rangle\right)$ in binary classification.

Problem 26. The Multi-class SVM learning problem $\left(\mathcal{H}_{\Psi,W}, \mathcal{Z}, \ell_{\text{hinge}}\right)$ using Ridge regularization with shrinkage parameter $\lambda>0$ is

$$w^{*} = \arg\min_{w}\left(\underbrace{\frac{1}{m}\sum_{i=1}^{m}\max_{\xi\in\mathcal{Y}}\left(\ell_{0-1}\left(h_w, z_i\right) + \left\langle w, \Psi\left(x_i,\xi\right) - \Psi\left(x_i,y_i\right)\right\rangle\right)}_{\hat{R}_{\mathcal{S}}(w)\text{ as empirical risk function}} + \lambda\left\|w\right\|_2^2\right) \tag{3.4}$$

with predictive rule

$$h_{w^{*}}\left(x\right) = \arg\max_{y\in\mathcal{Y}}\left\langle w^{*}, \Psi\left(x,y\right)\right\rangle$$

3.3. **About the learning algorithm.**

Note 27. Problem 26 can be solved with linear programming algorithms, as well as any variation of the SGD algorithm. –We discuss the latter.

Example 28. The recursion of SGD applied in (3.4) is

$$w^{(t+1)} = w^{(t)} + \eta_t\left(\Psi\left(x^{(t)}, \hat{y}^{(t)}\right) - \Psi\left(x^{(t)}, y^{(t)}\right)\right)$$

with

$$\hat{y}^{(t)} \in \max_{\xi\in\mathcal{Y}} \left(\ell_{0-1}\left(h_w, z^{(t)}\right) + \left\langle w^{(t)}, \Psi\left(x^{(t)}, \xi\right) - \Psi\left(x^{(t)}, y^{(t)}\right)\right\rangle\right),$$

learning rate η_t and for a randomly chosen example $\left(x^{(t)}, y^{(t)}\right) \sim g$, at iteration t.

Solution. The recursion is

$$w^{(t+1)} = w^{(t)} + \eta_t v_t$$

where

$$v_t \in \partial_w \ell_{\text{hinge}}\left(h_w, z = (x, y)\right)$$

I will use Proposition 12 from Handout 2: Elements of convex learning problems. Because I can write

$$\ell_{\text{hinge}}\left(h_w, z\right) = \max_{\xi\in\mathcal{Y}} \left\{c\left(w; \xi\right)\right\}$$

with

$$c\left(w; \xi\right) = \ell_{0-1}\left(h_w, z_i\right) + \left\langle w, \Psi\left(x_i, \xi\right) - \Psi\left(x_i, y_i\right)\right\rangle$$

if $\hat{y}$ is such as $c\left(w; \hat{y}\right) = \max_{\xi\in\mathcal{Y}}\left(c\left(w; \xi\right)\right)$ then

$$\begin{aligned}
v_t &\in \partial_w \ell_{\text{hinge}}\left(h_w, z = (x, y)\right)\\
&= \partial_w c\left(w; \hat{y}\right)\\
&= \partial_w \left(\ell_{0-1}\left(h_w, \hat{y}\right) + \left\langle w, \Psi\left(x^{(t)}, \hat{y}\right) - \Psi\left(x^{(t)}, y^{(t)}\right)\right\rangle\right)\\
&= \partial_w \ell_{0-1}\left(h_w, \hat{y}\right) + \nabla_w \left\langle w, \Psi\left(x^{(t)}, \hat{y}\right) - \Psi\left(x^{(t)}, y^{(t)}\right)\right\rangle\\
&= \Psi\left(x^{(t)}, \hat{y}\right) - \Psi\left(x^{(t)}, y^{(t)}\right)
\end{aligned}$$

Note 29. For the SGD implementation, the computation of error bounds and the derivation of assumptions that guarantee a PAC learning algorithm is straightforward.