Project 3: List Prediction and Online SVM

Dawei Wang (daweiwan@andrew.cmu.edu)

Theoritical: List Prediction

- 1.2.1: Monotone Submodularity:

  MONOTONICITY: $\forall L_1, L_2, T \subseteq S$, $L_1 \cap T \subseteq (L_1 \cup L_2) \cap T$, [1] $\Rightarrow |L_1 \cap T| \leq |(L_1 \cup L_2) \cap T|$, therefore $f(L_1; T) = \min(1, |L_1 \cap T|) \leq \min(1, |(L_1 \cup L_2) \cap T|) = f(L_1 \cup L_2; T)$;

  SUBMODULARITY: $\forall L_1, L_2, T \subseteq S$, and $\forall s \in S$, we have

$$b(s|L_1) = f(L_1 \cup \{s\}; T) - f(L_1; T) \tag{1}$$
$$b(s|L_1 \cup L_2) = f(L_1 \cup L_2 \cup \{s\}; T) - f(L_1 \cup L_2; T) \tag{2}$$

  consider the following cases:

  - $s \in L_1 \Rightarrow L_1 \cup \{s\} = L_1, L_1 \cup L_2 \cup \{s\} = L_1 \cup L_2 \Rightarrow b(s|L_1) = 0 \geq 0 = b(s|L_1 \cup L_2)$;
  - $s \notin L_1$, $s \in L_2 \Rightarrow L_1 \cup L_2 \cup \{s\} = L_1 \cup L_2 \Rightarrow b(s|L_1) \geq 0$ (monotonicity) $= b(s|L_1 \cup L_2)$;
  - $s \notin L_1$, $s \notin L_2$, $s \in T$, from Equations (1) and (2):

$$b(s|L_1) = \min(1, |(L_1 \cup \{s\}) \cap T|) - f(L_1; T) \tag{3}$$
$$= \min(1, |(L_1 \cap T) \cup (\{s\} \cap T)|) - f(L_1; T) \tag{4}$$
$$= \min(1, |L_1 \cap T| + 1) - f(L_1; T) = 1 - f(L_1; T) \tag{5}$$
$$b(s|L_1 \cup L_2) = \min(1, |(L_1 \cap L_2 \cap T) \cup (\{s\} \cap T)|) - f(L_1 \cup L_2; T) \tag{6}$$
$$= \min(1, |L_1 \cap L_2 \cap T| + 1) - f(L_1 \cup L_2; T) = 1 - f(L_1 \cup L_2; T) \tag{7}$$

    since $f(L_1; T) \leq f(L_2 \cup L_2; T)$ (monotonicity), we have $b(s|L_1) \geq b(s|L_1 \cup L_2)$;
  - $s \notin L_1$, $s \notin L_2$, $s \notin T \Rightarrow \{s\} \cap T = \varnothing$, from Equations (4) and (6):

$$b(s|L_1) = \min(1, |L_1 \cap T|) - f(L_1; T) = 0 \tag{8}$$
$$b(s|L_1, L_2) = \min(1, |L_1 \cap L_2 \cap T|) - f(L_1 \cup L_2; T) = 0 \tag{9}$$

    we still have $b(s|L_1) = 0 \geq 0 = b(s|L_1 \cup L_2)$.

  therefore, the *multiple guess* function is monotone and submodular. ■

- 1.2.2: Greedy Guarantee:

$$\textbf{STEP1}: \Delta_i = f(L^*) - f(L^G_{i-1}) \leq f(L^G_{i-1} \oplus L^*) - f(L^G_{i-1}) \quad \text{(monotonicity)} \tag{10}$$
$$= \textstyle\sum_{j=1}^{k} \left[f(L^G_{i-1} \oplus L^*_j) - f(L^G_{i-1} \oplus L^*_{j-1}) \right] \tag{11}$$
$$= \textstyle\sum_{j=1}^{k} \left[f(L^G_{i-1} \oplus L^*_{j-1} \oplus l^*_j) - f(L^G_{i-1} \oplus L^*_{j-1}) \right] \tag{12}$$
$$\leq \textstyle\sum_{j=1}^{k} \left[f(L^G_{i-1} \oplus l^*_j) - f(L^G_{i-1}) \right] \quad \text{(submodularity)} \tag{13}$$
$$\textbf{STEP2}: \Delta_i \leq \textstyle\sum_{j=1}^{k} \left[f(L^G_{i-1} \oplus l^*_j) - f(L^G_{i-1}) \right] \tag{14}$$
$$\leq \textstyle\sum_{j=1}^{k} \left[f(L^G_{i-1} \oplus l^G_i) - f(L^G_{i-1}) \right] \quad \text{(greediness)} \tag{15}$$
$$= \textstyle\sum_{j=1}^{k} \left[f(L^G_i) - f(L^*) \right] + \left[f(L^*) - f(L^G_{i-1}) \right] \tag{16}$$
$$= \textstyle\sum_{j=1}^{k} \left[-\Delta_{i+1} + \Delta_i \right] = k(\Delta_i - \Delta_{i+1}) \tag{17}$$
$$\Delta_{i+1} \leq (1 - 1/k)\, \Delta_i \tag{18}$$
$$\textbf{STEP3}: \Delta_{k+1} \leq (1 - 1/k)\Delta_k \leq (1 - 1/k)^k \Delta_1 \leq (1/e)\Delta_1 \tag{19}$$
$$f(L^*) - f(L_G) \leq (1/e) f(L^*) \tag{20}$$
$$f(L_G) \geq (1 - 1/e) f(L^*) \tag{21}$$

  therefore, the greedy optimization policy ensures near-optimal performance. ■

[1] $\forall x \in L_1 \cap T$, $x \in L_1$ and $x \in T$; therefore $x \in L_1 \cap L_2$, and $x \in (L_1 \cup L_2) \cap T$.

Theoritical: Online SVM

- 2.2.1: Formulation Equivalency: the constraints can be re-written as

$$\begin{aligned} &\xi_i \geq 0 \\ &\xi_i \geq 1 - y_i w^T f_i \end{aligned} \quad \Leftrightarrow \quad \xi_i \geq \max\left(0, 1 - y_i w^T f_i\right) \tag{22}$$

and the optimization can be performed sequentially with respect to ξ and w:

$$\min_{\xi, w} \left[\frac{\lambda}{2}\|w\|^2 + \sum_{i=1}^{T} \xi_i\right] = \min_{w} \left[\frac{\lambda}{2}\|w\|^2 + \min_{\xi} \sum_{i=1}^{T} \xi_i\right] = \min_{w} \left[\frac{\lambda}{2}\|w\|^2 + \sum_{i=1}^{T} \max(0, 1 - y_i w^T f_i)\right] \tag{23}$$

therefore the two problem formulations are equivalent. ■

- 2.2.2: Convexity: considering the objective function

$$J(w; \mathcal{D}) = \sum_{i=1}^{T} \left[\frac{\lambda}{2T}\|w\|^2 + \max(0, 1 - y_i w^T f_i)\right] \tag{24}$$

with $\forall w_1, w_2$ and $\gamma \in [0, 1]$, we have [2]

$$\begin{aligned} \max(0, 1 - y_i[(1-\gamma)w_1 + \gamma w_2]^T f_i) &= \max(0, (1-\gamma)(1 - y_i w_1^T f_i) + \gamma(1 - y_i w_2^T f_i)) && (25)\\ &\leq \max(0, (1-\gamma)(1 - y_i w_1^T f_i)) + \max(0, \gamma(1 - y_i w_2^T f_i)) && (26)\\ &= (1-\gamma)\max(0, 1 - y_i w_1^T f_i) + \gamma \max(0, 1 - y_i w_2^T f_i) && (27)\\ \|(1-\gamma)w_1 + \gamma w_2\|^2 &\leq (1-\gamma)^2\|w_1\|^2 + \gamma^2\|w_2\|^2 \quad \text{(triangle inequality)} && (28)\\ &\leq (1-\gamma)\|w_1\|^2 + \gamma\|w_2\|^2 && (29) \end{aligned}$$

and combining both terms yields

$$\begin{aligned} J\left[(1-\gamma)w_1 + \gamma w_2; \mathcal{D}\right] &\leq \textstyle\sum_{i=1}^{T} \left[\frac{\lambda}{2T}\|(1-\gamma)w_1 + \gamma w_2\|^2 + \max(0, 1 - y_i[(1-\gamma)w_1 + \gamma w_2]^T f_i)\right] && (30)\\ &\leq \textstyle\sum_{i=1}^{T} \frac{\lambda}{2T}((1-\gamma)\|w_1\|^2 + \gamma\|w_2\|^2) \\ &\quad + (1-\gamma)\max(0, 1 - y_i w_1^T f_i) + \gamma \max(0, 1 - y_i w_2^T f_i) && (31)\\ &= (1-\gamma)J(w_1; \mathcal{D}) + \gamma J(w_2; \mathcal{D}) && (32) \end{aligned}$$

therefore this objective function is a convex function. ■

- 2.2.3: Sub-gradient Descent: for $\forall w, u$, if $1 - y_t w^T f_t > 0$

$$\begin{aligned} l(w) + \nabla l_t(w)^T(u - w) &= l(w) + (\tfrac{\lambda}{T} w^T - y_t f_t^T)(u - w) && (33)\\ &= \tfrac{\lambda}{2T}\|w\|^2 + (1 - y_t w^T f_i) + \tfrac{\lambda}{T} w^T u - y_t f_t^T u - \tfrac{\lambda}{T}\|w\|^2 + y_t f_t^T w && (34)\\ &= -\tfrac{\lambda}{2T}\|w\|^2 + \tfrac{2\lambda}{2T} w^T u + (1 - y_t f_t^T u) && (35)\\ &= -\tfrac{\lambda}{2T}\|w - u\|^2 + \tfrac{\lambda}{2T}\|u\|^2 + (1 - y_t f_t^T u) && (36)\\ &\leq \tfrac{\lambda}{2T}\|u\|^2 + \max(0, 1 - y_t f_t^T u) = l(u) && (37) \end{aligned}$$

otherwise, i.e., if $1 - y_t w^T f_t \leq 0$, $\max(0, 1 - y_t w^T f_t) = 0$, we have

$$\begin{aligned} l(w) + \nabla l_t(w)^T(u - w) &= l(w) + (\tfrac{\lambda}{T} w^T)(u - w) && (38)\\ &= \tfrac{\lambda}{2T}\|w\|^2 - \tfrac{\lambda}{T}\|w\|^2 + \tfrac{2\lambda}{2T} w^T u && (39)\\ &= -\tfrac{\lambda}{2T}\|w - u\|^2 + \tfrac{\lambda}{2T}\|u\|^2 + 0 && (40)\\ &\leq \tfrac{\lambda}{2T}\|u\|^2 + \max(0, 1 - y_t f_t^T u) = l(u) && (41) \end{aligned}$$

therefore the proposed sub-gradient $\nabla l_t(w)$ is valid. ■

[2] $\max(0, x+y) \leq \max(0, x) + \max(0, y)$, since if $x + y \leq 0$, $\max(0, x+y) = 0 \leq \max(0, x) + \max(0, y)$; otherwise, if $x + y > 0$, $\max(0, x) + \max(0, y) \geq x + y = \max(0, x + y)$. Hence this inequality always holds regardless of x and y.

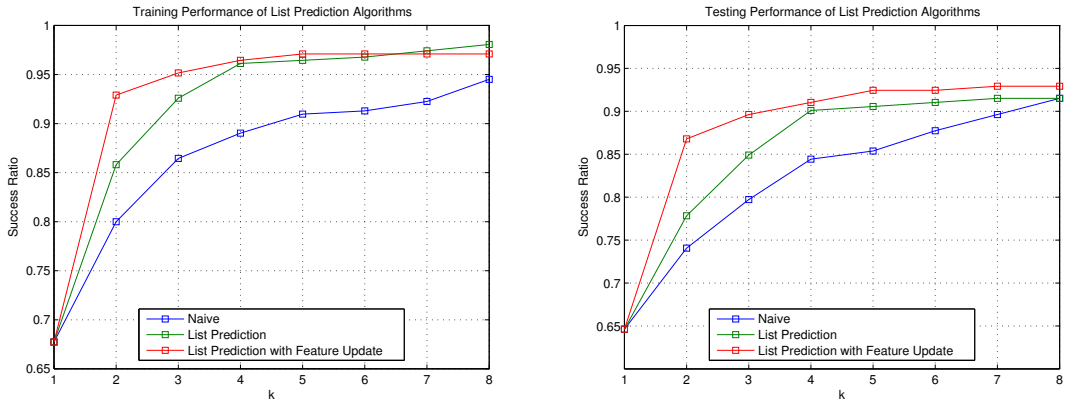


Figure 1: Performance of List Prediction Algorithms (Training: Left; Testing: Right)

Table 1: Binary Mis-classification Errors (Chance Errors) (%)

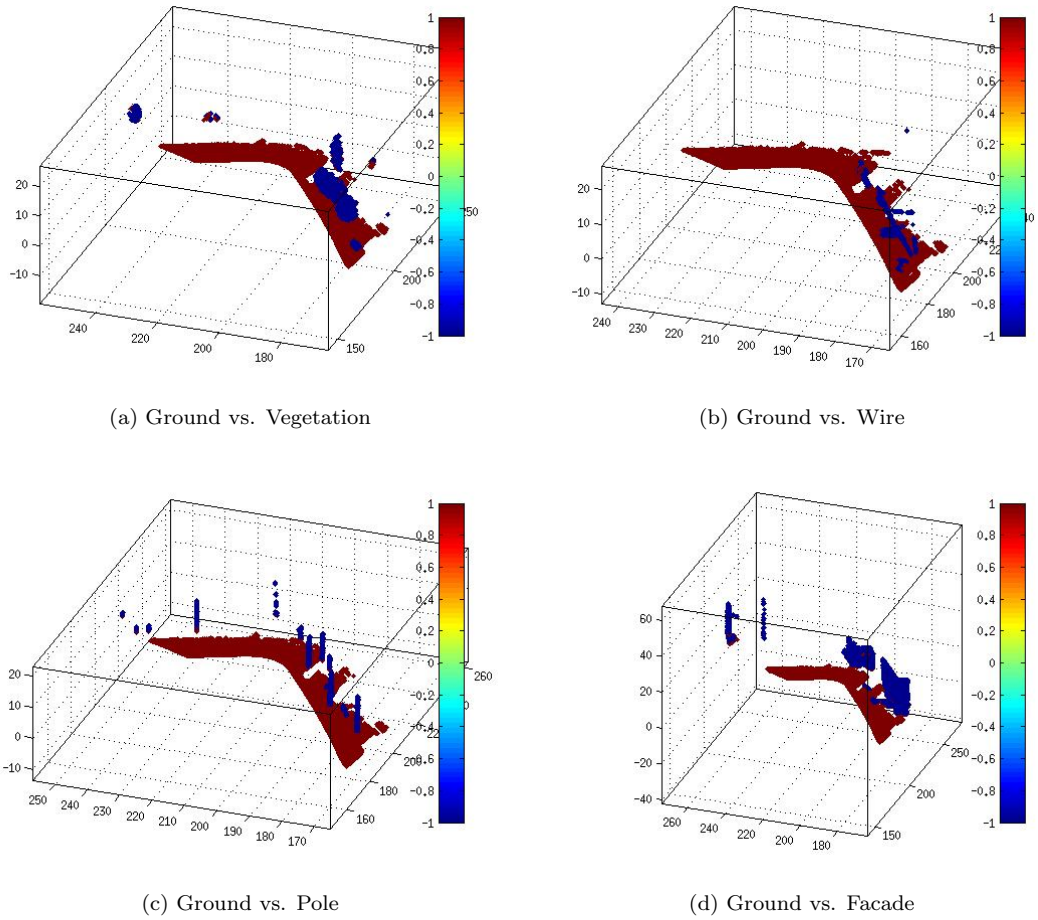
	Vegetation	Wire	Pole	Ground	Facade
Count	8322	818	1429	67161	12092
Vegetation	-	8.96 (8.95)	11.08 (14.65)	0.56 (11.02)	16.56 (40.77)
Wire	8.96 (8.95)	-	5.61 (36.40)	0.20 (1.20)	6.34 (6.34)
Pole	13.68 (14.65)	6.19 (36.40)	-	0.18 (2.08)	10.13 (10.57)
Ground	0.55 (11.02)	0.21 (1.20)	0.19 (2.08)	-	0.55 (15.26)
Facade	16.59 (40.77)	6.34 (6.34)	10.26 (10.57)	0.53 (15.26)	-

Programming: List Prediction

Run `gen_plots.m` to obtain Figure 1. A linear regressor is used over all three algorithms for fair comparison; an extra intercept term is explicitly appended to each feature vector (trajectory). It is expected that the success ratio will be monotonically increasing as k continues to grow; for both training and testing, list prediction strategies, with or without feature update, outperform the naive strategy consistently. This can be explained by the fact that list prediction attempts to explore diverse options by greedily selecting the features that yield the greatest marginal benefits, as opposed to the naive strategy which does not. This property is particularly important when the benefit function is *multiple guess*. Updating the features essentially incorporates the diversity information into the features, allowing the learner to exploit them more directly, therefore resulting in better accuracies. Testing accuracies are generally lower than training accuracies due to generalization errors. ■

Programming: Online SVM

Run `gen_results.m` to obtain Table 1. (1) Each element dictates the binary mis-classification error (chance error) when the online support vector machine is run on the dataset constructed by combining the data points from the two classes, as displayed at its corresponding row and column titles. (2) Apparently there are a few combinations that were not classified well (errors were marginally smaller than the chance errors) such as (Pole, Vegetation), (Wire, Vegetation), (Pole, Facade), and (Wire, Facade). This may be primarily attributed to the quality (distinctiveness) of the features. (3) It took roughly 10 seconds to compute that table. The time complexity is linear in the number of data points and the number of features. (4) The next page includes some images from select combinations of classes. ■



(a) Ground vs. Vegetation

(b) Ground vs. Wire

(c) Ground vs. Pole

(d) Ground vs. Facade

Figure 2: Images of the Classified Data

Appendix: List of Functions and Scripts

The implementation is entirely in MATLAB.

- `ftupdate.m`: update the features by appending similarity metrics;
- `lblupdate.m`: update the labels by replacing them with marginal benefits;
- `train.m`: train a linear regressor given features and labels;
- `predict.m`: pick the best data point for each environment, given the model and features;
- `naive.m`: the function that implements the naive prediction strategy;
- `listpred.m`: the function that implements the list prediction strategy;
- `lpwftupdate.m`: the function that implements the list prediction strategy with feature update.
- `onlinesvm.m`: the function that implements online SVM;
- `gen_plots.m`: a script that executes all the list prediction strategies;
- `gen_results.m`: a script that runs the online SVM on all possible pairs of classes;
- `fscatter3.m`: a third-party script that plots 3D point cloud in MATLAB;
- `data/`: a directory that contains all the data files.

The submission also includes this writeup.