Introduction to Machine Learning
Lecture 13

Mehryar Mohri
Courant Institute and Google Research
mohri@cims.nyu.edu

Multi-Class Classification

Motivation

- Real-world problems often have multiple classes: text, speech, image, biological sequences.
- Algorithms studied so far: designed for binary classification problems.
- How do we design multi-class classification algorithms?
  - can the algorithms used for binary classification be generalized to multi-class classification?
  - can we reduce multi-class classification to binary classification?

Multi-Class Classification Problem

- Training data: sample drawn i.i.d. from set X according to some distribution D,

$$S=((x_1, y_1), \ldots, (x_m, y_m)) \in X \times Y,$$

  - mono-label case: $\mathrm{Card}(Y)=k$.
  - multi-label case: $Y=\{-1,+1\}^k$.

- Problem: find classifier $h: X \to Y$ in H with small generalization error,
  - mono-label case: $R_D(h)=\mathrm{E}_{x\sim D}[1_{h(x)\neq f(x)}]$.
  - multi-label case: $R_D(h)=\mathrm{E}_{x\sim D}\left[\frac{1}{k}\sum_{l=1}^{k} 1_{[h(x)]_k\neq[f(x)]_k}\right]$.

Notes

- In most tasks, number of classes $k \leq 100$.
- For k large or infinite, problem often not treated as a multi-class classification problem, e.g., automatic speech recognition.
- Computational efficiency issues arise for larger ks.
- In general, classes not balanced.

Approaches

- Single classifier:
  - Decision trees.
  - AdaBoost-type algorithm.
  - SVM-type algorithm.
- Combination of binary classifiers:
  - One-vs-all.
  - One-vs-one.
  - Error-correcting codes.

AdaBoost-Type Algorithm

(Schapire and Singer, 2000)

- Training data (multi-label case):

$$(x_1, y_1), \ldots, (x_m, y_m) \in X \times \{-1, 1\}^k.$$

- Reduction to binary classification:
  - each example leads to k binary examples:

$$(x_i, y_i) \rightarrow ((x_i, 1), y_i[1]), \ldots, ((x_i, k), y_i[k]), i \in [1, m].$$

  - apply AdaBoost to the resulting problem.
  - choice of α_t .
- Computational cost: mk distribution updates at each round.

AdaBoost.MH

$H \subseteq (\{-1,+1\}^k)^{(X\times Y)}$.

$\text{AdaBoost.MH}(S=((x_1,y_1),\ldots,(x_m,y_m)))$

for $i \leftarrow 1$ **to** m **do**
  for $l \leftarrow 1$ **to** k **do**
    $D_1(i,l) \leftarrow \frac{1}{mk}$
for $t \leftarrow 1$ **to** T **do**
  $h_t \leftarrow$ base classifier in H with small error $\epsilon_t = \Pr_{D_t}[h_t(x_i,l) \neq y_i[l]]$
  $\alpha_t \leftarrow$ choose ▷ to minimize Z_t
  $Z_t \leftarrow \sum_{i,l} D_t(i,l)\exp(-\alpha_t y_i[l] h_t(x_i,l))$
  for $i \leftarrow 1$ **to** m **do**
    for $l \leftarrow 1$ **to** k **do**
      $D_{t+1}(i,l) \leftarrow \frac{D_t(i,l)\exp(-\alpha_t y_i[l] h_t(x_i,l))}{Z_t}$
$f_T \leftarrow \sum_{t=1}^{T} \alpha_t h_t$
return $h_T = \text{sgn}(f_T)$

Bound on Empirical Error

- **Theorem**: The empirical error of the classifier output by AdaBoost.MH verifies:

$$\widehat{R}(h) \leq \prod_{t=1}^{T} Z_t.$$

- **Proof**: similar to the proof for AdaBoost.
- **Choice of α_t**:
  - for $H \subseteq (\{-1,+1\}^k)^{X \times Y}$, as for AdaBoost, $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$.
  - for $H \subseteq ([-1,1]^k)^{X \times Y}$, same choice: minimize upper bound.
  - other cases: numerical/approximation method.

Multi-Class SVMs

(Weston and Watkins, 1999)

- **Optimization problem:**

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2}\sum_{l=1}^{k} \|\mathbf{w}_l\|^2 + C\sum_{i=1}^{m}\sum_{l\neq y_i} \xi_{il}$$

$$\text{subject to: } \mathbf{w}_{y_i}\cdot\mathbf{x}_i + b_{y_i} \geq \mathbf{w}_l\cdot\mathbf{x}_i + b_l + 2 - \xi_{il}$$

$$\xi_{il} \geq 0, \quad (i,l)\in[1,m]\times(Y-\{y_i\}).$$

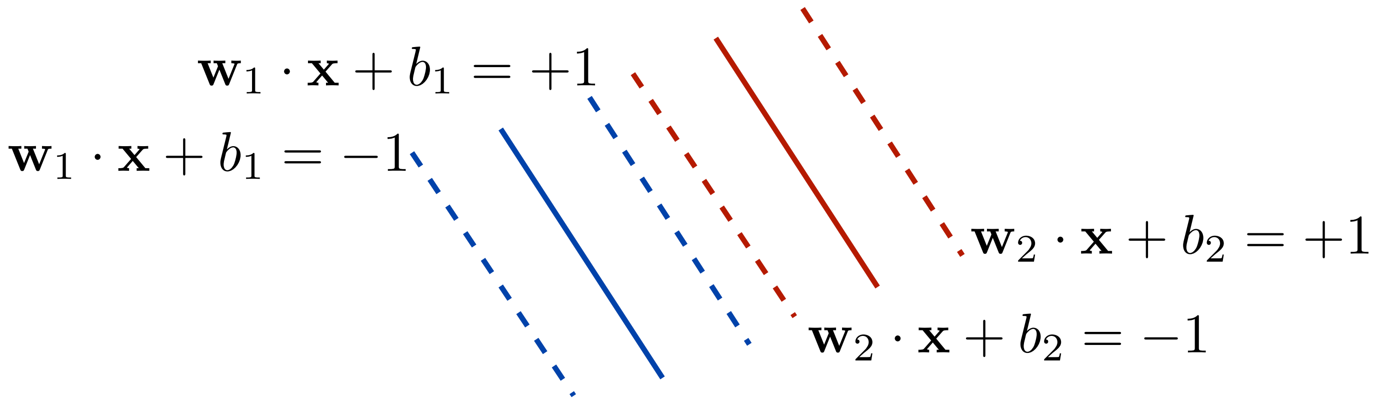
- **Decision function:**

$$h\colon x\mapsto \operatorname*{argmax}_{l\in Y}\,(\mathbf{w}_l\cdot\mathbf{x} + b_l).$$

Notes

- Idea: slack variable ξ_{il} penalizes case where

$$(\mathbf{w}_{y_i} \cdot \mathbf{x}_i + b_{y_i}) - (\mathbf{w}_l \cdot \mathbf{x}_i + b_l) < 2.$$



- Binary SVM obtained as special case:

$$\mathbf{w}_1 = -\mathbf{w}_2, b_1 = -b_2, \xi_{i1} = \xi_{i2} = 2\xi_i.$$

Dual Formulation

- **Notation:**

$$\alpha_i = \sum_{l=1}^{k} \alpha_{il} \quad c_{il} = 1_{y_i=l}.$$

- **Optimization problem:**

$$\max_{\alpha} \; 2\sum_{i=1}^{m} \alpha_i + \sum_{i,j,l} [-\frac{1}{2} c_{jy_i} \alpha_i \alpha_j + \alpha_{il}\alpha_{jy_i} - \frac{1}{2}\alpha_{il}\alpha_{jl}](\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subject to: } \forall l \in [1, k], \sum_{i=1}^{m} \alpha_{il} = \sum_{i=1}^{m} c_{il}\alpha_i$$

$$\forall (i, l) \in [1, m] \times (Y - \{y_i\}), 0 \leq \alpha_{il} \leq C, \alpha_{iy_i} = 0.$$

- **Decision function:**

$$h \colon x \mapsto \operatorname*{argmax}_{l=1,\ldots,k} \Big[\sum_{i=1}^{m} (c_{il}\alpha_i - \alpha_{il})(\mathbf{x}_i \cdot \mathbf{x}) + b_l \Big].$$

Notes

- PDS kernel instead of inner product
- Optimization: complex constraints, mk-size problem.
- Generalization error: leave-one-out analysis and bounds of binary SVM apply similarly.
- One-vs-all solution (non-optimal) feasible solution of multi-class SVM problem.
- Simplification: single slack variable per point (Crammer and Singer, 2002), $\xi_{il} \to \xi_i$.

Simplified Multi-Class SVMs

(Crammer and Singer, 2001)

- Optimization problem:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2}\sum_{l=1}^{k} \|\mathbf{w}_l\|^2 + C\sum_{i=1}^{m} \xi_i$$

$$\text{subject to: } \mathbf{w}_{y_i} \cdot \mathbf{x}_i + \delta_{y_i,l} \geq \mathbf{w}_l \cdot \mathbf{x}_i + 1 - \xi_i$$

$$(i,l) \in [1,m] \times Y.$$

- Decision function:

$$h\colon x \mapsto \underset{l \in Y}{\operatorname{argmax}}\,(\mathbf{w}_l \cdot \mathbf{x}).$$

Notes

- Single slack variable per point: maximum of previous slack variables (penalty for worst class):

$$\sum_{l=1}^{k} \xi_{il} \rightarrow \max_{l=1}^{k} \xi_{il}.$$

- PDS kernel instead of inner product
- Optimization: complex constraints, mk-size problem.
  - specific solution based on decomposition into m disjoint sets of constraints (Crammer and Singer, 2001).

Dual Formulation

- Optimization problem: ($\boldsymbol{\alpha}_i$ ith row of matrix $\boldsymbol{\alpha}$)

$$\max_{\boldsymbol{\alpha}=[\alpha_{ij}]} \sum_{i=1}^{m} \boldsymbol{\alpha}_i \cdot \mathbf{e}_{y_i} - \frac{1}{2} \sum_{i=1}^{m} (\boldsymbol{\alpha}_i \cdot \boldsymbol{\alpha}_j)(\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\text{subject to: } 0 \leq \boldsymbol{\alpha}_i \leq \mathbf{C} \wedge \boldsymbol{\alpha}_i \cdot \mathbf{1} = 0, i \in [1, m].$$

- Decision function:

$$h(x) = \operatorname*{argmax}_{l=1}^{k} \Big(\sum_{i=1}^{m} \alpha_{il} (\mathbf{x}_i \cdot \mathbf{x}) \Big).$$

Approaches

- Single classifier:
  - Decision trees.
  - AdaBoost-type algorithm.
  - SVM-type algorithm.
- Combination of binary classifiers:
  - One-vs-all.
  - One-vs-one.
  - Error-correcting codes.

One-vs-All

- Technique:
  - for each class $l \in Y$ learn binary classifier $h_l = \text{sgn}(f_l)$.
  - combine binary classifiers via voting mechanism, typically majority vote: $h \colon x \mapsto \operatorname{argmax}_{l \in Y} f_l(x)$.
- Problem: poor justification.
  - calibration: classifier scores not comparable.
  - nevertheless: simple and frequently used in practice, computational advantages in some cases.

One-vs-One

- Technique:
  - for each pair $(l, l') \in Y, l \neq l'$ learn binary classifier $h_{ll'} : X \to \{0, 1\}$.
  - combine binary classifiers via majority vote:

$$h(x) = \operatorname*{argmax}_{l' \in Y} \left| \{l : h_{ll'}(x) = 1\} \right|.$$

- Problem:
  - computational: train $k(k-1)/2$ binary classifiers.
  - overfitting: size of training sample could become small for a given pair.

Computational Comparison

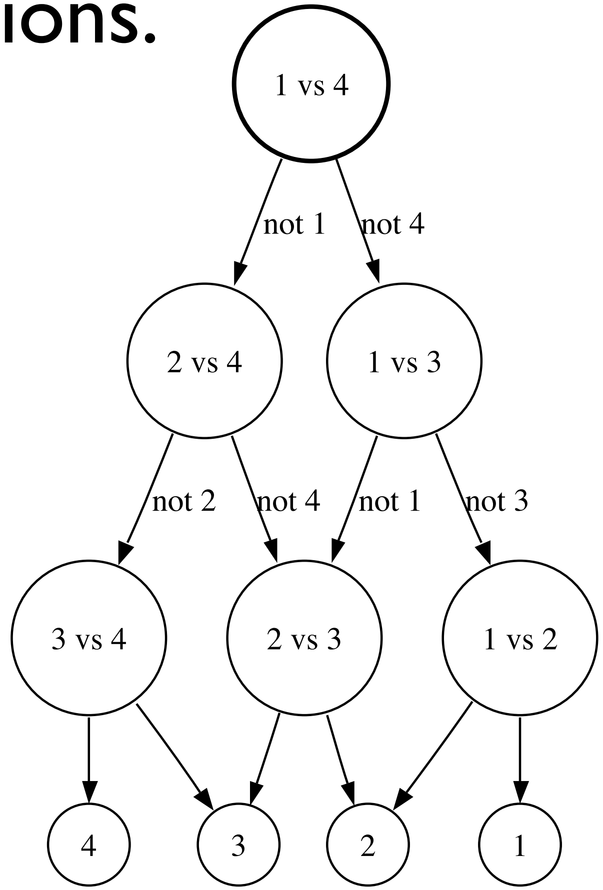
	Training	Testing
One-vs-all	$O(kB_{\text{train}}(m))$ $O(km^{\alpha})$	$O(kB_{\text{test}})$
One-vs-one	$O(k^2B_{\text{train}}(m/k))$ (on average) $O(k^{2-\alpha}m^{\alpha})$	$O(k^2B_{\text{test}})$ $smaller\ N_{SV}\ per\ B$

Time complexity for SVMs, α less than 3.

Heuristics

(Platt et al., 2000)

- Training:
  - reuse of computation between classifiers, e.g., sharing of kernel computations.
  - caching.
- Testing:
  - directed acyclic graph.
  - smaller number of tests.
  - ordering?



Error-Correcting Code Approach

(Dietterich and Bakiri, 1995)

- Technique:
  - assign F-long binary code word to each class:
    $$\longrightarrow \mathbf{M} = [\mathbf{M}_{lj}] \in \{0, 1\}^{[1,k]\times[1,F]}.$$
  - learn binary classifier $f_j\colon X \to \{0, 1\}$ for each column. Example x in class l labeled with M_{lj}.
  - classifier output: $\Big(\mathbf{f}(x) = (f_1(x), \ldots, f_F(x))\Big)$,
    $$h\colon x \mapsto \operatorname*{argmin}_{l \in Y} d_{\text{Hamming}}\Big(\mathbf{M}_l, \mathbf{f}(x)\Big).$$

Illustration

- 8 classes, code length: 6.

codes

classes	1	2	3	4	5	6
1	0	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	1	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	1	0	1	0	0

$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
0	1	1	0	1	1

new example x

Error-Correcting Codes - Design

- Main ideas:
  - independent columns: otherwise no effective discrimination.
  - distance between rows: if the minimal Hamming distance between rows is d, then the multi-class can correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors.
  - columns may correspond to features selected for the task.
  - one-vs-all and one-vs-one (with ternary codes) are special cases.

Extensions

(Allwein et al., 2000)

- Matrix entries in $\{-1, 0, +1\}$:
  - examples marked with 0 disregarded during training.
  - $\longrightarrow$ one-vs-one becomes also a special case.
- Margin loss L: function of $yf(x)$, e.g., hinge loss.
  - Hamming loss:

$$h(x) = \operatorname*{argmin}_{l \in \{1, \ldots, k\}} \sum_{j=1}^{F} \frac{1 - \operatorname{sgn}\big(\mathbf{M}_{lj} f_j(x)\big)}{2}.$$

  - Margin loss:

$$h(x) = \operatorname*{argmin}_{l \in \{1, \ldots, k\}} \sum_{j=1}^{F} L\big(\mathbf{M}_{lj} f_j(x)\big).$$

Continuous Codes

(Crammer and Singer, 2000, 2002)

- Optimization problem: ($\mathbf{M}_l$ lth row of $\mathbf{M}$)

$$\min_{\mathbf{M},\boldsymbol{\xi}} \|\mathbf{M}\|_2^2 + C\sum_{i=1}^m \xi_i$$

$$\text{subject to: } K(\mathbf{f}(x_i), \mathbf{M}_{y_i}) \geq K(\mathbf{f}(x_i), \mathbf{M}_l) + 1 - \xi_i$$

$$(i, l) \in [1, m] \times [1, k].$$

- Decision function:

$$h\colon x \mapsto \operatorname*{argmax}_{l \in \{1,\ldots,k\}} K(\mathbf{f}(x), \mathbf{M}_l).$$

Ideas

- Continuous codes: real-valued matrix.
- Learn matrix code $\mathbf{M}$.
- Similar optimization problems with other matrix norms.
- Kernel K used for similarity between matrix row and prediction vector.

Multiclass Margin

- Definition: let $H \subseteq \mathbb{R}^{X \times Y}$. The margin of the training point $(x, y) \in X \times Y$ for the hypothesis $h \in H$ is

$$\gamma_h(x, y) = h(x, y) - \max_{y' \neq y} h(x, y').$$

- Thus, h misclassifies (x, y) iff $\gamma_h(x, y) \leq 0$.

Margin Bound

(Koltchinskii and Panchenko, 2002)

- Theorem: Let $H_1 = \{x \mapsto h(x,y) \colon h \in H, y \in Y\}$ where $H \subseteq \mathbb{R}^{X \times Y}$. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds

$$\Pr[\gamma_h(x,y) \leq 0] \leq \widehat{\Pr}[\gamma_h(x,y) \leq \rho] + c\frac{k^2 \mathfrak{R}_m(H_1)}{\rho} + c' \sqrt{\frac{\log \frac{1}{\delta}}{m}},$$

for some constants $c, c' > 0$.

Applications

- One-vs-all approach is the most widely used.
- No clear empirical evidence of the superiority of other approaches (Rifkin and Klautau, 2004).
  - except perhaps on small data sets with relatively large error rate.
- Large structured multi-class problems: often treated as ranking problems (see next lecture).

References

- Erin L. Allwein, Robert E. Schapire and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113-141, 2000.
- K. Crammer and Y. Singer. Improved output coding for classification using continuous relaxation. In Proceedings of *NIPS*, 2000.
- Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.
- Koby Crammer and Yoram Singer. On the Learnability and Design of Output Codes for Multiclass Problems. *Machine Learning 47*, 2002.
- Thomas G. Dietterich, Ghulum Bakiri: Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research (JAIR) 2*: 263-286, 1995.
- John C. Platt, Nello Cristianini, and John Shawe-Taylor. Large Margin DAGS for Multiclass Classification. In *Advances in Neural Information Processing Systems 12 (NIPS 1999)*, pp. 547-553, 2000.

References

- Ryan Rifkin. "Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning." Ph.D. Thesis, MIT, 2002.

- Rifkin and Klautau. "In Defense of One-Vs-All Classification." *Journal of Machine Learning Research*, 5:101-141, 2004.

- Robert E. Schapire. The boosting approach to machine learning: An overview. In D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, B. Yu, editors, *Nonlinear Estimation and Classification*. Springer, 2003.

- Robert E. Schapire, Yoav Freund, Peter Bartlett and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5): 1651-1686, 1998.

- Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135-168, 2000.

- Jason Weston and Chist Watkins. Support Vector Machines for Multi-Class Pattern Recognition. *Proceedings of the Seventh European Symposium On Artificial Neural Networks (ESANN '99)*, 1999.