ECS289: Scalable Machine Learning

Cho-Jui Hsieh
UC Davis

Oct 27, 2015

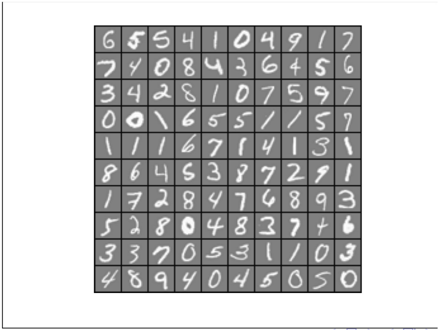
Outline

- One versus all/One versus one
- Ranking loss for multiclass/multilabel classification
- Scaling to millions of labels

Multiclass Learning

- n data points, L labels, d features
- Input: training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$:
  - Each $\boldsymbol{x}_i$ is a d dimensional feature vector
  - Each $y_i \in \{1, \ldots, L\}$ is the corresponding label
  - Each training data belongs to one category
- Goal: find a function to predict the correct label

$$f(\boldsymbol{x}) \approx y$$

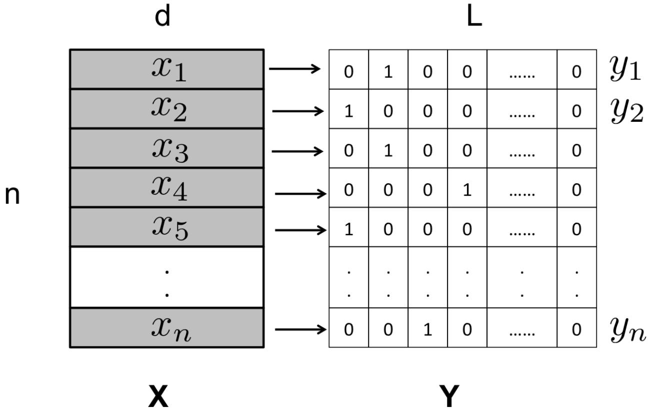


Multi-label Problems

- n data points, L labels, d features
- Input: training data $\{\boldsymbol{x}_i, \mathbf{y}_i\}_{i=1}^n$:
  - Each $\boldsymbol{x}_i$ is a d dimensional feature vector
  - Each $\mathbf{y}_i \in \{0, 1\}^L$ is a label vector (or $Y_i \in \{1, 2, \ldots, L\}$)
    Example: $\mathbf{y}_i = [0, 0, 1, 0, 0, 1, 1]$ (or $Y_i = \{3, 6, 7\}$)
  - Each training data can belong to multiple categories
- Goal: Given a testing sample $\boldsymbol{x}$, predict the correct labels

Document 1	{Sports, Politics}
Document 2	{Science, Politics}
.	.
.	.
.	.
Document n	{Environment}

Illustration



- Multiclass: each row of L has exact one "1"
- Multilabel: each row of L can have multiple ones

Measure the accuracy (multi-class)

- Let $\{\boldsymbol{x}_i, y_i\}_{i=1}^m$ be a set of testing data for multi-class problems
- Let $z_1, \ldots, z_m$ be the prediction for each testing data
- Accuracy:

$$\frac{1}{m}\sum_{i=1}^{m} I(y_i = z_i)$$

Measure the accuracy (multi-class)

- Let $\{\boldsymbol{x}_i, y_i\}_{i=1}^m$ be a set of testing data for multi-class problems
- Let $z_1, \ldots, z_m$ be the prediction for each testing data
- Accuracy:

$$\frac{1}{m}\sum_{i=1}^{m} I(y_i = z_i)$$

- If the algorithm outputs a set of k potential labels for each sample:

$$Z_1, Z_2, \ldots, Z_m$$

  Each Z_i is a set of k labels
- Precision@k:

$$\frac{1}{m}\sum_{i=1}^{m} I(y_i \in Z_i)$$

Example (multiclass)

x_1
x_2
x_3
x_4
x_5
x_n

X

0	1	0	0	y_1
1	0	0	0	y_2
0	1	0	0	
0	0	0	1	
1	0	0	0	
0	0	1	0	y_n

Y

23	49	10	20
13	52	24	17
71	62	53	46
11	8	13	14
50	60	70	10
48	51	61	97

Z(prediction)

Example (multiclass)

x_1
x_2
x_3
x_4
x_5
x_n

X

0	1	0	0	y_1
1	0	0	0	y_2
0	1	0	0	
0	0	0	1	
1	0	0	0	
0	0	1	0	y_n

Y

23	49	10	20
13	52	24	17
71	62	53	46
11	8	13	14
50	60	70	10
48	51	61	97

Z(prediction)

Accuracy = Precision@1 = 2/6

Example (multiclass)

x_1
x_2
x_3
x_4
x_5
x_n

X

0	1	0	0	y_1
1	0	0	0	y_2
0	1	0	0	
0	0	0	1	
1	0	0	0	
0	0	1	0	y_n

Y

23	49	10	20
13	52	24	17
71	62	53	46
11	8	13	14
50	60	70	10
48	51	61	97

Z(prediction)

Precision@2 = 4/6

Example (multiclass)

x_1
x_2
x_3
x_4
x_5
x_n

X

0	1	0	0	y_1
1	0	0	0	y_2
0	1	0	0	
0	0	0	1	
1	0	0	0	
0	0	1	0	y_n

Y

23	49	10	20
13	52	24	17
71	62	53	46
11	8	13	14
50	60	70	10
48	51	61	97

Z(prediction)

Precision@3 = 5/6

Measure the accuracy (multi-label)

- Let $\{\boldsymbol{x}_i, \mathbf{y}_i\}_{i=1}^m$ be a set of testing data for multi-label problems
- For each testing data, the classifier outputs $\boldsymbol{z}_i \in \{0,1\}^L$
- We use $Y_i = \{t : (\mathbf{y}_i)_t \neq 0\}$ and $Z_i = \{t : (\boldsymbol{z}_i)_t \neq 0\}$ to denote the subset of real and predictive labels for the i-th testing data.
  - If each Z_i has k elements, then Precision@k is defined by:

$$\frac{1}{m}\sum_{i=1}^{m}\frac{Z_i \cap Y_i}{k}$$

  - Hamming loss: overall classification performance

$$\frac{1}{m}\sum_{i=1}^{m} d(\mathbf{y}_i, \boldsymbol{z}_i),$$

    where $d(\mathbf{y}_i, \boldsymbol{z}_i)$ measures the number of places where $\mathbf{y}_i$ and $\boldsymbol{z}_i$ differ.
  - ROC curve (assume the classifier predicts a ranking of labels for each data point)

Example (multilabel)

x_1
x_2
x_3
x_4
x_5
x_n

X

0	1	1	0	y_1
1	0	0	1	y_2
0	1	0	0	
1	1	0	1	
1	0	1	1	
0	0	1	0	y_n

Y

23	49	10	20
13	52	24	17
71	62	53	46
11	8	13	14
50	60	70	10
48	51	61	97

Z(prediction)

Example (multilabel)

x_1
x_2
x_3
x_4
x_5
x_n

X

0	1	1	0	y_1
1	0	0	1	y_2
0	1	0	0	
1	1	0	1	
1	0	1	1	
0	0	1	0	y_n

Y

23	49	10	20
13	52	24	17
71	62	53	46
11	8	13	14
50	60	70	10
48	51	61	97

Z(prediction)

Precision@1 = 3/6

Example (multilabel)

x_1
x_2
x_3
x_4
x_5
x_n

X

0	1	1	0	y_1
1	0	0	1	y_2
0	1	0	0	
1	1	0	1	
1	0	1	1	
0	0	1	0	y_n

Y

23	49	10	20
13	52	24	17
71	62	53	46
11	8	13	14
50	60	70	10
48	51	61	97

Z(prediction)

Precision@2 = 5/12

Traditional Approach

- Many algorithms for binary classification
- Idea: transform multi-class or multi-label problems to multiple binary classification problems
- Two approaches:
  - One versus All (OVA)
  - One versus One (OVO)

One Versus All (OVA)

- Multi-class/multi-label problems with L categories
- Build L different binary classifiers
- For the t-th classifier:
  - Positive samples: all the points in class t $(\{\boldsymbol{x}_i : t \in \mathbf{y}_i\})$
  - Negative samples: all the points not in class t $(\{\boldsymbol{x}_i : t \notin \mathbf{y}_i\})$
  - $f_t(\boldsymbol{x})$: the decision value for the t-th classifier

    (larger $f_t \Rightarrow$ higher probability that $\boldsymbol{x}$ in class t)
- Prediction:

$$f(\boldsymbol{x}) = \arg\max_t f_t(\boldsymbol{x})$$

- Example: using SVM to train each binary classifier.

One Versus One (OVO)

- Multi-class/multi-label problems with L categories
- Build $L(L-1)$ different binary classifiers
- For the (s,t)-th classifier:
  - Positive samples: all the points in class s ($\{\boldsymbol{x}_i : s \in \mathbf{y}_i\}$)
  - Negative samples: all the points in class t ($\{\boldsymbol{x}_i : t \in \mathbf{y}_i\}$)
  - $f_{s,t}(\boldsymbol{x})$: the decision value for this classifier
    (larger $f_{s,t}(\boldsymbol{x}) \Rightarrow$ label s has higher probability than label t)
  - $f_{t,s}(\boldsymbol{x}) = -f_{s,t}(\boldsymbol{x})$
- Prediction:

$$f(\boldsymbol{x}) = \arg\max_s \left(\sum_t f_{s,t}(\boldsymbol{x}) \right)$$

- Example: using SVM to train each binary classifier.
- Not for multilabel problems.

OVA vs OVO

- Prediction accuracy: depends on datasets
- Computational time:

  OVA needs to train L classifiers

  OVO needs to train $L(L-1)/2$ classifiers
- Is OVA always faster than OVO?

OVA vs OVO

- Prediction accuracy: depends on datasets
- Computational time:

  OVA needs to train L classifiers

  OVO needs to train $L(L-1)/2$ classifiers
- Is OVA always faster than OVO?

  NO, depends on the time complexity of the binary classifier
  - If the binary classifier requires $O(n)$ time for n samples:
    OVA and OVO have similar time complexity
  - If the binary classifier requires $O(n^{1.xx})$ time:
    OVO is faster than OVA

OVA vs OVO

- Prediction accuracy: depends on datasets
- Computational time:

  OVA needs to train L classifiers

  OVO needs to train $L(L-1)/2$ classifiers
- Is OVA always faster than OVO?

  NO, depends on the time complexity of the binary classifier
  - If the binary classifier requires $O(n)$ time for n samples:

    OVA and OVO have similar time complexity
  - If the binary classifier requires $O(n^{1.xx})$ time:

    OVO is faster than OVA
- LIBSVM (kernel SVM solver): OVO
- LIBLINEAR (linear SVM solver): OVA

Comparisons (accuracy)

For kernel SVM with RBF kernel

	One-against-one		DAG		One-against-all		[25], [27]		C&S	
Problem	(C,γ)	rate	(C,γ)	rate	(C,γ)	rate	(C,γ)	rate	(C,γ)	rate
iris	$(2^{12},2^{-9})$	**97.333**	$(2^{12},2^{-8})$	96.667	$(2^{9},2^{-3})$	96.667	$(2^{12},2^{-8})$	**97.333**	$(2^{10},2^{-7})$	**97.333**
wine	$(2^{7},2^{-10})$	**99.438**	$(2^{6},2^{-9})$	98.876	$(2^{7},2^{-6})$	98.876	$(2^{0},2^{-2})$	98.876	$(2^{1},2^{-3})$	98.876
glass	$(2^{11},2^{-2})$	71.495	$(2^{12},2^{-3})$	**73.832**	$(2^{11},2^{-2})$	71.963	$(2^{9},2^{-4})$	71.028	$(2^{4},2^{1})$	71.963
vowel	$(2^{4},2^{0})$	**99.053**	$(2^{2},2^{2})$	98.674	$(2^{4},2^{1})$	98.485	$(2^{3},2^{0})$	98.485	$(2^{1},2^{3})$	98.674
vehicle	$(2^{9},2^{-3})$	86.643	$(2^{11},2^{-5})$	86.052	$(2^{11},2^{-4})$	**87.470**	$(2^{10},2^{-4})$	86.998	$(2^{9},2^{-4})$	86.761
segment	$(2^{6},2^{0})$	97.403	$(2^{11},2^{-3})$	97.359	$(2^{7},2^{0})$	97.532	$(2^{5},2^{0})$	**97.576**	$(2^{0},2^{3})$	97.316
dna	$(2^{3},2^{-6})$	95.447	$(2^{3},2^{-6})$	95.447	$(2^{2},2^{-6})$	95.784	$(2^{4},2^{-6})$	95.616	$(2^{1},2^{-6})$	**95.869**
satimage	$(2^{4},2^{0})$	91.3	$(2^{4},2^{0})$	91.25	$(2^{2},2^{1})$	91.7	$(2^{3},2^{0})$	91.25	$(2^{2},2^{2})$	**92.35**
letter	$(2^{4},2^{2})$	**97.98**	$(2^{4},2^{2})$	**97.98**	$(2^{2},2^{2})$	97.88	$(2^{1},2^{2})$	97.76	$(2^{3},2^{2})$	97.68
shuttle	$(2^{11},2^{3})$	99.924	$(2^{11},2^{3})$	99.924	$(2^{9},2^{4})$	99.910	$(2^{9},2^{4})$	99.910	$(2^{12},2^{4})$	**99.938**

(See “A comparison of methods for multiclass support vector machines”, 2002)

Comparisons (training time)

For kernel SVM (with RBF kernel)

Problem	One-against-one training testing	#SVs	DAG training testing	#SVs	One-against-all training testing	#SVs	[25], [27] training testing	#SVs	C&S training testing	#SVs
iris	**0.04**	16.9	**0.04**	*15.6*	0.10	16.0	0.15	16.2	16.84	27.8
wine	**0.12**	56.3	0.13	56.5	0.20	*29.2*	0.28	54.5	0.39	41.6
glass	**2.42**	*112.5*	2.85	114.2	10.00	129.0	7.94	124.1	7.60	143.3
vowel	**2.63**	345.3	3.98	365.1	9.28	392.6	14.05	*279.4*	20.54	391.0
vehicle	**19.73**	302.4	35.18	293.1	142.50	343.0	88.61	*264.2*	1141.76	264.9
segment	**17.10**	442.4	23.25	*266.8*	68.85	446.3	66.43	358.2	192.47	970.3
dna	**10.60**	967	10.74	967	23.47	1152	13.5	951	16.27	*945*
	6.91		**6.30**		8.43		6.91		6.39	
satimage	**24.85**	1611	25.1	1611	136.42	2170	48.21	*1426*	89.58	2670
	13.23		12.67		19.22		**11.89**		23.61	
letter	**298.08**	8931	298.62	8931	1831.80	10129	8786.20	7627	1227.12*	*6374*
	126.10		**92.8**		146.43		142.75		110.39	
shuttle	170.45	301	**168.87**	301	202.96	330	237.80	202	2205.78*	*198*
	6.99		5.09		5.99		4.64		**4.26**	

*: stopping tolerance $\epsilon = 0.1$ is used.

(See "A comparison of methods for multiclass support vector machines", 2002)

Methods Using Instance-wise Ranking Loss

Main idea

- OVA and OVO: decompose the problem by labels
- However, the ranking of the labels for **a testing sample** is important
  - For multiclass classification, the score of y_i should be larger than other labels
  - For multilabel classification, the score of Y_i should be larger than other labels
- Both OVA and OVO decompose the problem into individual labels

  $\Rightarrow$ they cannot capture the ranking information
- Solve **one combined optimization problem** by minimizing the ranking loss

Main idea

- For simplicity, we assume a linear model
- Model parameters: $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L$
- For each data point $\boldsymbol{x}$, compute the decision value for each label:

$$\boldsymbol{w}_1^T \boldsymbol{x}, \quad \boldsymbol{w}_2^T \boldsymbol{x}, \quad \ldots, \quad \boldsymbol{w}_L^T \boldsymbol{x}$$

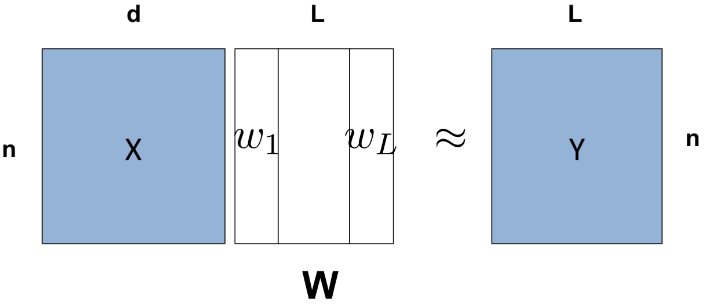
- Prediction:

$$y = \arg\max_t \boldsymbol{w}_t^T \boldsymbol{x}$$

- For training data $\boldsymbol{x}_i$, y_i is the true label, so we want

$$y_i \approx \arg\max_t \boldsymbol{w}_t^T \boldsymbol{x}_i \quad \forall i$$

Main idea



Question: how to define the distance between XW and Y

Weston-Watkins Formulation

- Proposed in Weston and Watkins, "Multi-class support vector machines". In ESANN, 1999.

$$\min_{\{\boldsymbol{w}_t\},\{\xi_i^t\}} \quad \frac{1}{2}\sum_{t=1}^{L}\|\boldsymbol{w}_t\|^2 + C\sum_{i=1}^{n}\sum_{t=1}^{L}\xi_i^t$$

$$\text{s.t.} \quad \boldsymbol{w}_{y_i}^T\boldsymbol{x}_i - \boldsymbol{w}_t^T\boldsymbol{x}_i \geq 1 - \xi_i^t, \quad \xi_i^t \geq 0 \;\; \forall t \neq y_i, \; \forall i = 1, \ldots, n$$

- If point i is in class y_i, for any other labels ($t \neq y_i$), we want

$$\boldsymbol{w}_{y_i}^T\boldsymbol{x}_i - \boldsymbol{w}_t^T\boldsymbol{x}_i \geq 1$$

  or we pay a penalty ξ_i^t
- Prediction:

$$f(\boldsymbol{x}) = \arg\max_t \boldsymbol{w}_t^T\boldsymbol{x}_i$$

Weston-Watkins Formulation (dual)

- The dual problem of Weston-Watkins formulation:

$$\min_{\{\boldsymbol{\alpha}_t\}} \frac{1}{2} \sum_{t=1}^{L} \|\boldsymbol{w}_t(\boldsymbol{\alpha})\| + \sum_{i=1}^{n} \sum_{t \neq y_i} \alpha_i^t$$

$$\text{s.t.} 0 \leq \alpha_i^t \leq C, \quad \forall t \neq y_i, \quad \forall i = 1, \ldots, n$$

- $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_L \in \mathbb{R}^n$: dual variables for each label
- $\boldsymbol{w}_t(\boldsymbol{\alpha}) = -\sum_i \alpha_i^t \boldsymbol{x}_i$, and $\alpha_i^{y_i} = -\sum_{t \neq y_i} \alpha_i^t$
- Can be solved by dual (block) coordinate descent

Crammer-Singer Formulation

- Proposed in Carmmer and Singer, "On the algorithmic implementation of multiclass kernel-based vector machines". JMLR, 2001.

$$\begin{aligned} \min_{\{\boldsymbol{w}_t\},\{\xi_i^t\}} \quad & \frac{1}{2}\sum_{t=1}^{L} \|\boldsymbol{w}_t\|^2 + C\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & \boldsymbol{w}_{y_i}^T \boldsymbol{x}_i - \boldsymbol{w}_t^T \boldsymbol{x}_i \geq 1 - \xi_i, \quad \forall t \neq y_i, \; \forall i = 1, \dots, n \\ & \xi_i \geq 0 \quad \forall i = 1, \dots, n \end{aligned}$$

- If point i is in class y_i, for any other labels $(t \neq y_i)$, we want

$$\boldsymbol{w}_{y_i}^T \boldsymbol{x}_i - \boldsymbol{w}_t^T \boldsymbol{x}_i \geq 1$$

- For each point i, we only pay the largest penalty
- Prediction:

$$f(\boldsymbol{x}) = \arg\max_t \boldsymbol{w}_t^T \boldsymbol{x}_i$$

Crammer-Singer Formulation (dual)

- The dual problem of Crammer-Singer formulation:

$$\min_{\{\boldsymbol{\alpha}_t\}} \frac{1}{2} \sum_{t=1}^{L} \|\boldsymbol{w}_t(\boldsymbol{\alpha})\| + \sum_{i=1}^{n} \sum_{t \neq y_i} \alpha_i^t$$

$$\text{s.t.} \left(\alpha_i^t \leq C, \quad \forall t, \sum_t \alpha_i^t = 0 \right), \quad \forall i = 1, 2, \ldots, n$$

- $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_L \in \mathbb{R}^n$: dual variables for each label
- $\boldsymbol{w}_t(\boldsymbol{\alpha}) = -\sum_i \alpha_i^t \boldsymbol{x}_i$, and $\alpha_i^{y_i} = -\sum_{t \neq y_i} \alpha_i^t$
- Can be solved by dual (block) coordinate descent

Comparisons

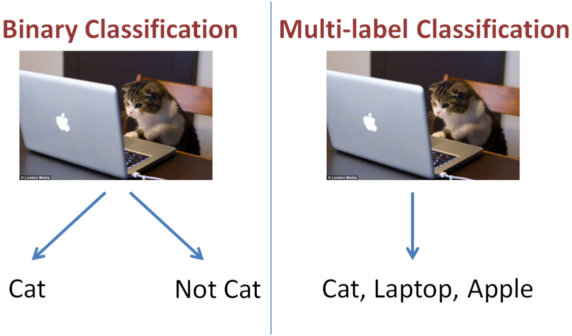
Dataset	OVR	CS	WW
NEWS20	85.39 ± 0.37	85.26 ± 0.37	85.10 ± 0.43
SECTOR	94.83 ± 0.56	95.17 ± 0.62	94.37 ± 0.57
MNIST	91.46 ± 0.23	92.50 ± 0.20	91.84 ± 0.21
RCV1	90.85 ± 0.10	91.19 ± 0.07	91.10 ± 0.11
COVER	71.35 ± 0.33	72.31 ± 0.37	72.40 ± 0.25

(See "A Sequential Dual Method for Large Scale Multi-Class Linear SVMs", KDD 2008)

Scaling to huge number of labels

Challenges: large number of categories

- Multi-label (or multi-class) classification with large number of labels
- Image classification—> 10000 labels
- Recommending tags for articles: millions of labels (tags)



Challenges: large number of categories

- Consider a problem with 1 million labels ($L = 1,000,000$)
- One versus all: reduce to 1 million binary problems
- Training: 1 million binary classification problems.

  Need 694 days if each binary problem can be solved in 1 minute
- Model size: 1 million models.

  Need 1 TB if each model requires 1MB.
- Prediction one testing data: 1 million binary prediction

  Need 1000 secs if each binary prediction needs 10^{-3} secs.

Several Approaches

- Label space reduction by Compressed Sensing (CS)
- Feature space reduction (PCA)
- Supervised latent feature model by matrix factorization

Compressed Sensing

- If $A \in \mathbb{R}^{n \times d}$, $\boldsymbol{w} \in \mathbb{R}^d$, $\mathbf{y} \in \mathbb{R}^n$, and

$$A\boldsymbol{w} = \mathbf{y}$$

- Given A and $\mathbf{y}$, can we recover $\boldsymbol{w}$?

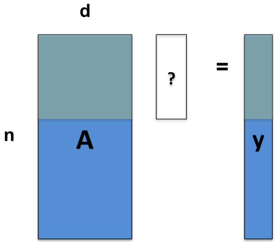
Compressed Sensing

- If $A \in \mathbb{R}^{n \times d}$, $\boldsymbol{w} \in \mathbb{R}^d$, $\mathbf{y} \in \mathbb{R}^n$, and

$$A\boldsymbol{w} = \mathbf{y}$$

- Given A and $\mathbf{y}$, can we recover $\boldsymbol{w}$?
- When $n \gg d$:

  Usually yes, because number of constraints $\gg$ number of variable (over-determined linear system)

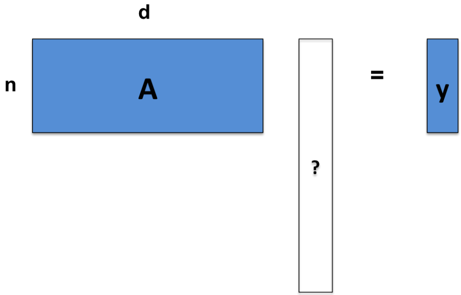


Compressed Sensing

- If $A \in \mathbb{R}^{n \times d}$, $\boldsymbol{w} \in \mathbb{R}^d$, $\mathbf{y} \in \mathbb{R}^n$, and

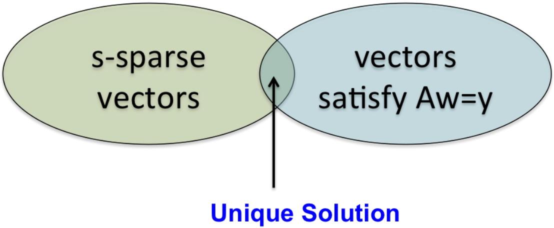
$$A\boldsymbol{w} = \mathbf{y}$$

- Given A and $\mathbf{y}$, can we recover $\boldsymbol{w}$?
- When $n \ll d$:
  No, because number of constraints $\ll$ number of variable
  (high dimensional regression, under-determined linear system, ...)



Compressed Sensing

- However, if we know $\boldsymbol{w}$ is a sparse vector (with $\leq s$ nonozero elements), and A satisfies certain condition, then we can recover $\boldsymbol{w}$ even in the high dimensional setting.



Compressed Sensing

- Conditions: $\boldsymbol{w}$ is s-sparse and A satisfies Restricted Isometry Property (RIP)
  - Example: all entries of A are generated i.i.d. from Gaussian $N(0, \sigma)$
  - ...
- $\boldsymbol{w}$ can be recoverd by $O(s \log d)$ samples
  - Lasso (ℓ_1-regularized linear regression)
  - Orthogonal Mathcing Pursuit (OMP)
  - ...
- How is this related to multilabel classification?

Label Space Reduction by Compressed Sensing

- Proposed in Hsu et al., "Multi-Label Prediction via Compressed Sensing". In NIPS 2009.
- Main idea: reduce the label space from $\mathbb{R}^L$ to $\mathbb{R}^k$, where $k \ll L$

$$\boldsymbol{z}_i = M\mathbf{y}_i \quad \text{where } M \in \mathbb{R}^{k \times L}$$

- If we can recover $\mathbf{y}_i$ by knowing $\boldsymbol{z}_i$ and M, then:

  we only need to learn a function to map $\boldsymbol{x}_i$ to $\boldsymbol{z}_i$:

$$f(\boldsymbol{x}_i) \approx \boldsymbol{z}_i$$

Label Space Reduction by Compressed Sensing

- Proposed in Hsu et al., "Multi-Label Prediction via Compressed Sensing". In NIPS 2009.
- Main idea: reduce the label space from $\mathbb{R}^L$ to $\mathbb{R}^k$, where $k \ll L$

$$\boldsymbol{z}_i = M\mathbf{y}_i \quad \text{where } M \in \mathbb{R}^{k \times L}$$

- If we can recover $\mathbf{y}_i$ by knowing $\boldsymbol{z}_i$ and M, then:

  we only need to learn a function to map $\boldsymbol{x}_i$ to $\boldsymbol{z}_i$:

$$f(\boldsymbol{x}_i) \approx \boldsymbol{z}_i$$

- By **compressed sensing**:
  - $\mathbf{y}_i$ can be recovered given $\boldsymbol{x}_i$ and M even when $k = O(s \log L)$
  - s is the number of nonzeroes in $\mathbf{y}_i$: usually very small in practice

Label Space Reduction by Compressed Sensing

Training:

Step 1: Construct M by i.i.d. Gaussian

Step 2: Compute $\boldsymbol{z}_i = M\mathbf{y}_i$ for all $i = 1, 2, \ldots, n$

Step 3: Learn a function f such that

$$f(\boldsymbol{x}_i) \approx \boldsymbol{z}_i$$

Prediction for a test sample $\boldsymbol{x}$

Step 1: compute $\boldsymbol{z} = f(\boldsymbol{x})$

Step 2: compute $\mathbf{y} \in \mathbb{R}^L$ by solving a Lasso problem

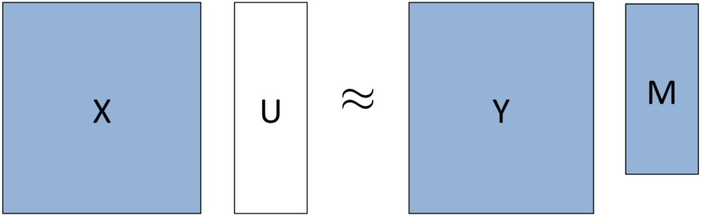
Step 3: Threshold $\mathbf{y}$ to give the prediction

Label Space Reduction by Compressed Sensing

- Reduce the label size from L to $O(s \log L)$
- Drawbacks:
  1. Slow prediction time (need to solve a Lasso problem for every prediction)
  2. Large error in practice

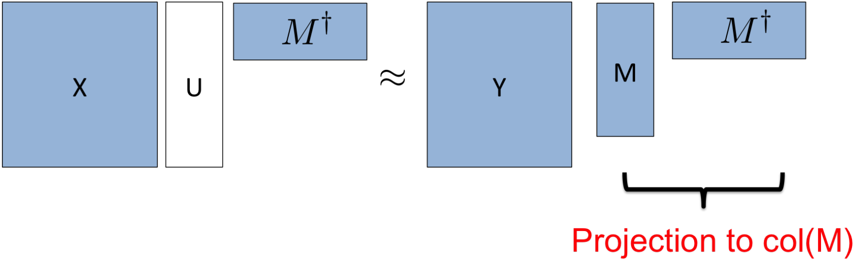
Another way to understand this algorithm

- $X \in \mathbb{R}^{n\times d}$: data matrix, each row is a data point $\boldsymbol{x}_i$
- $Y \in \mathbb{R}^{n\times L}$: label matrix, each row is $\mathbf{y}_i$
- $M \in \mathbb{R}^{L\times k}$: Gaussian random matrix
- Goal: find a U matrix such that $XU \approx YM$



Another way to understand this algorithm

- $X \in \mathbb{R}^{n \times d}$: data matrix, each row is a data point $\boldsymbol{x}_i$
- $Y \in \mathbb{R}^{n \times L}$: label matrix, each row is $\mathbf{y}_i$
- $M \in \mathbb{R}^{L \times k}$: Gaussian random matrix
- $M^\dagger$: psuedo inverse of M
- Goal: find a U matrix such that $XUM^\dagger \approx YMM^\dagger$



Feature space reduction

- Another way to improve speed: reduce the feature space
- Dimension reduction from d dimensional space to k dimensional space

$$\boldsymbol{x}_i \to N\boldsymbol{x}_i = \bar{\boldsymbol{x}}_i,$$

  where $N \in \mathbb{R}^{k \times d}$ and $k \ll d$
- Matrix form:

$$\bar{X} = XN^T$$

- Multilabel learning on the reduced feature space:

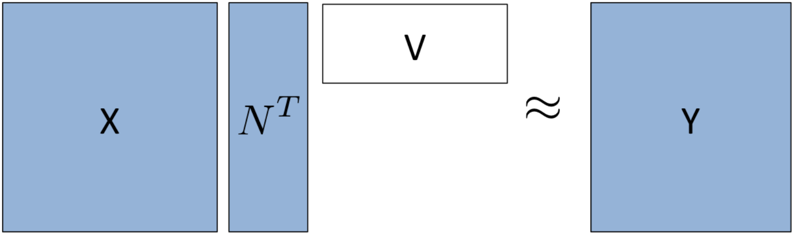
  Learn V such that

$$\bar{X}V \approx Y$$

- Time complexity: reduced by a factor of n/k
- Several ways to choose N: PCA, random projection, . . .

Another way to understand this algorithm

- $X \in \mathbb{R}^{n \times d}$: data matrix, each row is a data point $\boldsymbol{x}_i$
- $Y \in \mathbb{R}^{n \times L}$: label matrix, each row is $\mathbf{y}_i$
- $N \in \mathbb{R}^{k \times d}$: matrix for dimensional reduction
- Goal: find a V matrix such that $XN^T V \approx Y$



Supervised latent space model (by matrix factorization)

- Label space reduction: find U such that

$$X\color{blue}{U}V^T \approx Y$$

- Feature space reduction: find V such that

$$XU\color{blue}{V}^T \approx Y$$

- How to improve?

  Find best U and V simultaneously

- Proposed in
  - Chen and Lin, "Feature-aware label space dimension reduction for multi-label classification". In NIPS 2012.
  - Xu et al., "Speedup Matrix Completion with Side Information: Application to Multi-Label Learning". In NIPS 2013.
  - Yu et al., "Large-scale Multi-label Learning with Missing Labels". In ICML 2014.

Low Rank Modeling for Multilabel Classification

- Let $X \in \mathbb{R}^{n \times d}$ be the data matrix, $Y \in \mathbb{R}^{n \times L}$ be the 0-1 label matrix
- Find $U \in \mathbb{R}^{d \times k}$ and $V \in \mathbb{R}^{L \times k}$ such that

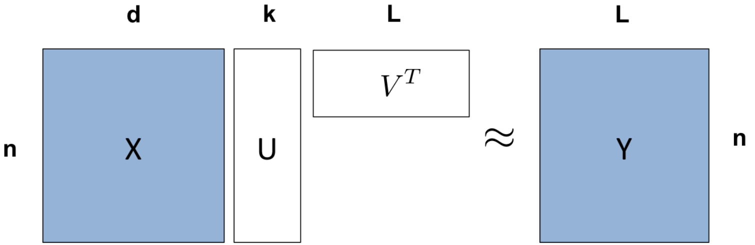
$$Y \approx XUV^T$$

- Obtain U, V by solving the following optimization problem:

$$\min_{U \in \mathbb{R}^{d \times k}, V \in \mathbb{R}^{L \times k}} \|Y - XUV^T\|_F^2 + \lambda_1 \|U\|_F^2 + \lambda_2 \|V\|_F^2$$

  where λ_1, λ_2 are the regularization parameters
- Solve by alternating minimization.

Low Rank Modeling for Multilabel Classification



Time complexity:

- $O(nkL)$ for updating V
- $O(ndk)$ for updating U
- Overall: $O(nkL + ndk)$ per iteration
- Original time complexity: $O(ndL)$ per iteration

Extensions

- Extend to general loss function:

$$\min_{U \in \mathbb{R}^{d \times k}, V \in \mathbb{R}^{L \times k}} \sum_{i,j} \text{dis}(Y_{ij}, (XUV^T)_{ij}) + \lambda_1 \|U\|_F^2 + \lambda_2 \|V\|_F^2$$

- Can handle problems with missing data:

$$\min_{U \in \mathbb{R}^{d \times k}, V \in \mathbb{R}^{L \times k}} \sum_{i,j \in \Omega} \text{dis}(Y_{ij}, (XUV^T)_{ij}) + \lambda_1 \|U\|_F^2 + \lambda_2 \|V\|_F^2$$

- Is it similar to inductive matrix completion?
- What's the time complexity for prediction?

Coming up

- Next class: paper presentations on mutlilabel classification and matrix completion

Questions?