Learning to Label Graph with Kernel Methods

Hongyu Su

Department of Information and Computer Science
Aalto University, School of Science and Technology
hongyu.su@aalto.fi

February 28, 2015

Outline

Ensemble Learning
- Bagging
- Boosting

Multi-Label/Task Classification
- via Single-Task Classifier
- Multi-Task Feature Learning
- Max-Margin Conditional Random Field

What is Ensemble System?

- Classical learning approach
  - Choose a model class (Bayesian, Random Field, Max-Margin, etc.)
  - Fit a single model to training data.
  - Predict on test data.
- An ensemble system
  - Choose a model class
  - Construct multiple models on training data (base learners)
  - Generate predictions on test data (base hypotheses)
  - Combine the predictions
- The idea of *ensemble learning* is to employ multiple learners, that do better than coin-toss, to get more accurate prediction.

Ensemble Methods

- Manipulating the training examples
  - Multiple weak hypotheses are generated by training individual classifier on different datasets obtained from common training data.
  - Bagging: sampling
  - Boosting: reweighting
- Manipulating the output target
  - The output targets are encoded with a bit vector, and individual base learner is trained to predict each of the bit in the code word.
  - Error-correcting code [Ghani, 2000]
- Manipulating the input features: random subspace method [Ho, 1998]
- Modifying the learning parameters

Bootstrap aggregating (bagging) [Breiman, 1994]

- Basic idea: Build different experts and let them vote for the finial prediction
- Sample N training examples from the total N, with replacement. The selecting probability for individual example is $\frac{1}{N}$.
  - Original dataset $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)\}$
  - Bootstrap dataset $\{(x_1, y_1), (x_3, y_3), (x_3, y_3), (x_5, y_5), (x_5, y_5)\}$

Bootstrap aggregating (bagging)

Algorithm Framework

Algorithm 1 Bagging

Require: Required ensemble size T, Training dataset $S = \{(x_1, y_1), \cdots, (x_N, y_N)\}$

1: **for** $t = 1$**to** T **do**

2: Build a dataset S_t by sampling N items, uniform at random with replacement from S

3: Train a weak hypothesis h_t by S_t, and add it to ensemble collection

4: **end for**

5: For a new testing point (x', y')

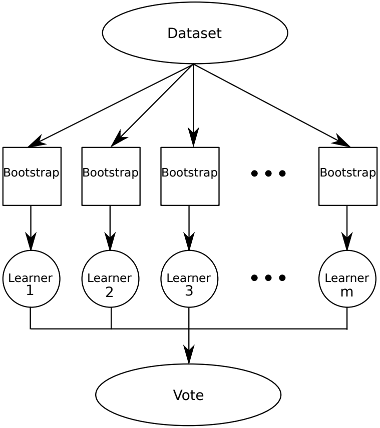
6: Get predictions from ensemble components.

7: Combine predictions to get a finial prediction.

Bootstrap aggregating (bagging) (cont.)

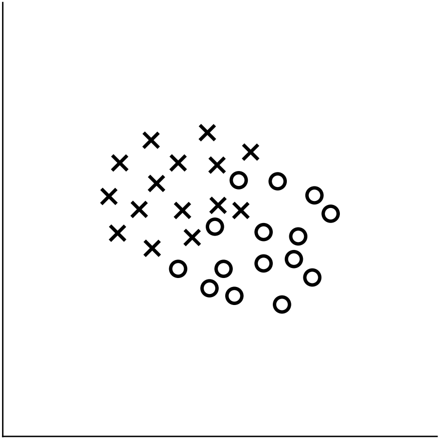
Algorithm Framework

- Diagram



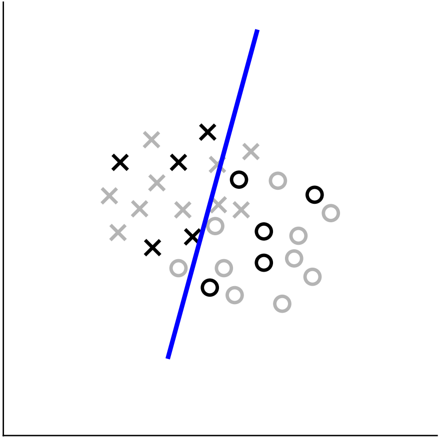
Bootstrap aggregating (bagging)

Example



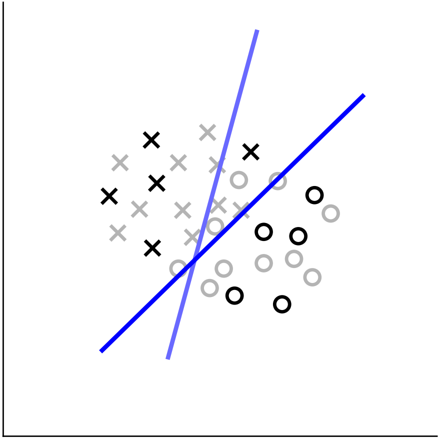
Bootstrap aggregating (bagging)

Example



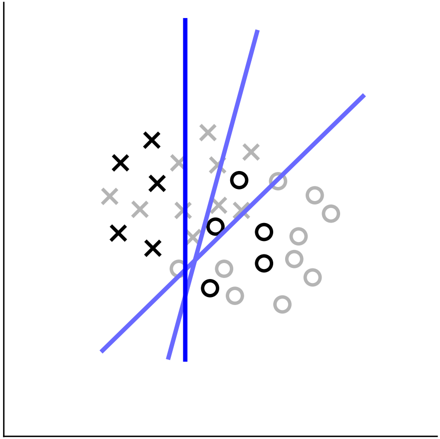
Bootstrap aggregating (bagging)

Example



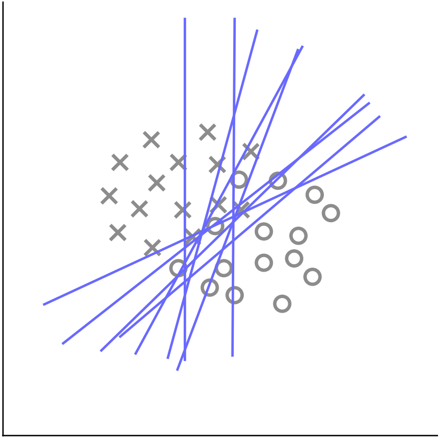
Bootstrap aggregating (bagging)

Example



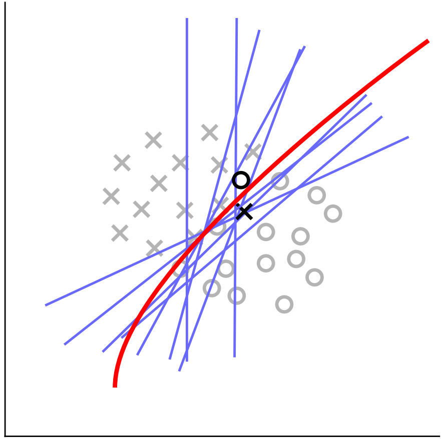
Bootstrap aggregating (bagging)

Example



Bootstrap aggregating (bagging)

Example



Bootstrap aggregating (bagging)

Why bagging works?

- Suppose there are 25 base learners
- Each has an error rate $\eta = 0.35$
- Assume independence among classifiers
- Probability that the ensemble makes a mistake

$$\sum_{i=13}^{25} \binom{25}{i} \eta^i (1-\eta)^{25-i} = 0.06$$

Bootstrap aggregating (bagging)

Bias-Variance Decomposition

- Error of learning algorithm on example x comes from three source
  - Noise, measure error / uncertainty for true label of x
  - Bias, how close/good, is the algorithm to optimal prediction
  - Variance, how much does prediction change if when training data change
- Error can be decomposed as:

$$error(x) = noise(x) + bias(x) + variance(x)$$

- Bias-variance trade-off
  - high bias $\mapsto$ low variance
  - low bias $\mapsto$ high variance

Bootstrap aggregating (bagging) (cont.)

Bias-Variance Decomposition

- Averaging can decrease variance while have the same bias

$$Var(\hat{x}) = \frac{Var(x)}{N}$$

- Bagging can be viewed as averaging.
- Bagging typically helps
  - Over-fitted base model: high variance & low bias
  - Unstable model: highly dependent on training data (decision tree, KNN, etc)

Boosting

- Boosting:
  - Base learner is just better than random prediction(coin toss)
  - Incrementally build an ensemble.
  - Each new model instant emphasizes the training instances that previous model misclassified.
- Adaptive Boosting (Adaboost) (■ See additional material) [Freund, 1992, Schapire and Singer, 1998]
- Boosting can:
  - Reduce variance, like bagging
  - Eliminate the effect the high bias of the weak learner [Freund and Schapire, 1996]

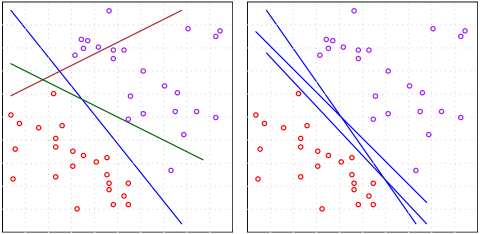
Outline

Ensemble Learning
- Bagging
- Boosting

Multi-Label/Task Classification
- via Single-Task Classifier
- Multi-Task Feature Learning
- Max-Margin Conditional Random Field

Support Vector Machine Revisit

- Learning Task
  - Single task, binary classification
  - m examples $\{(x_1, y_1), \cdots, (x_m, y_m)\}, x_i \in \mathcal{R}^d, y_i \in \{+, -\}$
- Linear separator of the form $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$
- Principal:*Empirical risk minimization*
  - minimize training error
- Hyperplanes with same empirical risk



Support Vector Machine Revisit

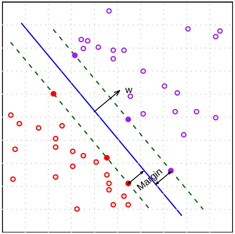
Max-Margin Hyperplane

- Maximum margin hyperplane (hard-Margin).
  - Robustness: small perturbation in training data does not change the classification
  - Performance: a large margin leads to low error on unseen data
- Hard-Margin SVM takes the form:

$$(\mathbf{w}, b) = \underset{\mathbf{w},b}{\mathbf{argmin}} \frac{1}{2}||\mathbf{w}||^2,$$

$$\text{s.t. } y_i\,(\mathbf{w}^T\mathbf{x}_i + b) \geq 1,$$

$$1 \leq i \leq n.$$



Support Vector Machine Revisit

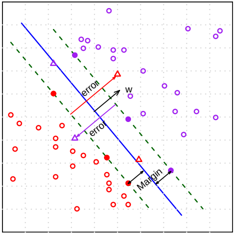
Soft-Margin SVM [Cortes and Vapnik, 1995]

- In practice, training data cannot be perfectly classified by a hyperplane
- Soft-Margin hyperplane allows training points to have smaller margin, subject to a penalty ξ
- Soft-Margin SVM takes the form

$$(\mathbf{w}, b) = \underset{\mathbf{w},b}{\mathbf{argmin}} \left(\frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{m} \xi_i \right),$$

$$\text{s.t. } y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) \geq 1 - \xi_i,$$

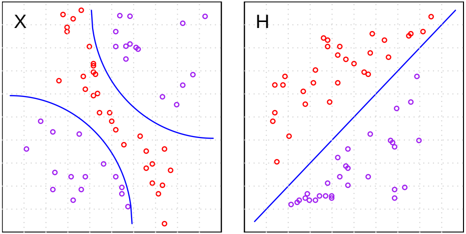
$$\xi_i \geq 0,\ 1 \leq i \leq n.$$



Support Vector Machine Revisit

Non-linear SVM

- Training data is not linear separable.



- Map data to high dimensional space $\mathcal{H}$,
  $\mathbf{x}_i = (x_1, x_2, \cdots, x_n) \xrightarrow{\varphi} \varphi(\mathbf{x}_i) = (\varphi_1, \varphi_2, \cdots, \varphi_k)$.

Support Vector Machine Revisit (cont.)

Non-linear SVM

- Dual is given by

$$
\begin{aligned}
\tilde{\mathcal{L}}(\alpha) &= \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\varphi(\mathsf{x}_i) \cdot \varphi(\mathsf{x}_j)) \\
&= \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathsf{x}_i, \mathsf{x}_j)
\end{aligned}
$$

- kernel trick: mapping observation to inner product space without computing the mappping explicitly.
- Example

Support Vector Machine Revisit (cont.)

Non-linear SVM

- Assume we have variables X and Y in n dimensional feature space $X, Y \in \mathbb{R}^n$

$$\phi(X) = (x_1, \ldots, x_n) \tag{1}$$
$$\phi(Y) = (y_1, \ldots, y_n) \tag{2}$$

Linear kernel corresponds to a dot product denoted by

$$K_{linear}(X, Y) = \phi(X) \cdot \phi(Y) \tag{3}$$
$$= \sum_i^n x_i y_i \tag{4}$$

A more interesting kernel can be constructed by constructing a kernel from two different feature sets $\phi_A(X) \in \mathbb{R}^a$ and

Support Vector Machine Revisit (cont.)

Non-linear SVM

$\phi_B(X) \in \mathbb{R}^b$. We can combine the two feature sets into $a+b$ dimensional vector $\phi_{A,B}(X) = [\phi_A(X), \phi_B(X)]$ and computing the kernel $K_{A,B}(X,Y)$ directly as $\phi_{A,B}(X) \cdot \phi_{A,B}(Y)$.
However, by definition $K_A(X,Y) + K_B(X,Y) = K_{A,B}(X,Y)$.
Thus, we can either concatenate features before hand and then compute a single kernel, or compute the individual kernels and sum them.
By taking an element-wise product $K_A \cdot K_B$ between kernels K_A and K_B we get features of type $\phi_A(X)_i \phi_B(X)_j$ for all $i \in [1, \ldots, a]$ and $j \in [1, \ldots, b]$.
For quadratic kernels, we have

Support Vector Machine Revisit (cont.)

Non-linear SVM

$$
\begin{aligned}
K_{quadratic}(X, Y) &= (K_{linear}(X, Y) + C)^2 && (5)\\
&= \left(\sum_{i}^{n} x_i y_i + C\right)^2 && (6)\\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j y_i y_j + 2\sum_{i=1}^{n} x_i y_i + C^2 && (7)\\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j y_i y_j + \sum_{i=1}^{n} \sqrt{2} x_i \sqrt{2} y_i + C^2 && (8)\\
&= \phi'(X) \cdot \phi'(Y), && (9)
\end{aligned}
$$

where the expanded feature spaces $\phi'(X)$ and $\phi'(Y)$ are

Support Vector Machine Revisit (cont.)

Non-linear SVM

$$\phi'(X) = (x_1x_1, \ldots, x_1x_n, x_2x_1, \ldots, x_nx_n; \sqrt{2}x_1, \ldots, \sqrt{2}x_n, C) \tag{10}$$

$$\phi'(Y) = (y_1y_1, \ldots, y_1y_n, y_2y_1, \ldots, y_ny_n; \sqrt{2}y_1, \ldots, \sqrt{2}y_n, C) \tag{11}$$

Structured Multi-Task Classification

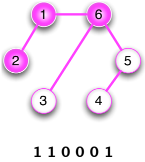
- m training data $\{(x_i, \mathbf{y}_i)\}_{i=1}^m$
- d dimension feature representation for each example $x \in \mathcal{R}^d$
- T targets/labels, $\mathbf{y} \in \{+,-\}^T$
- General form:

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x_{i1}} & \mathbf{x_{i2}} & \dots & \mathbf{x_{id}} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{md} \end{bmatrix}, \quad Y = \begin{bmatrix} y_{11} & y_{12} & \dots & y_{1T} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y_{i1}} & \mathbf{y_{i2}} & \dots & \mathbf{y_{iT}} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & \dots & y_{mT} \end{bmatrix}$$

$$x_j = \begin{pmatrix} x_{j1} & x_{j2} & \dots & x_{jm} \end{pmatrix}, \qquad y_j = \begin{pmatrix} ? & ? & \cdots & ? \end{pmatrix}$$

Structured Multi-Task Classification (cont.)

- Output graph: a structure $\mathcal{G} = (E, V)$ on multiple targets
  - A node $v \in V$ corresponds to target/label
  - An edge $e \in E$ represents dependency between pair of targets
- A prediction of multiple targets/labels can be seen as a labeling of the output graph.

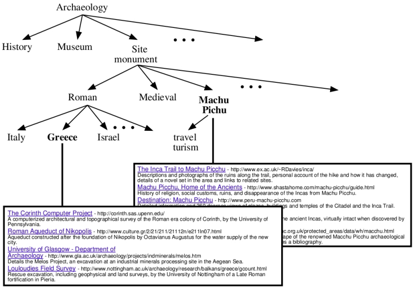


$\mathbf{y} = (y_1, y_2, y_3, y_4, y_5, y_6)$

Application Areas

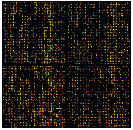
- Hierarchical document classification [Veeramachaneni et al., 2005, Rousu et al., 2006a]

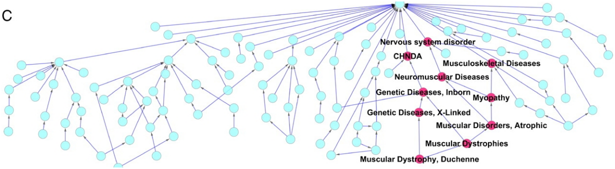
Given news articles x and the classification hierarchy $\mathcal{G}$, predict the multiple categories of the new article.



Application Areas (cont.)

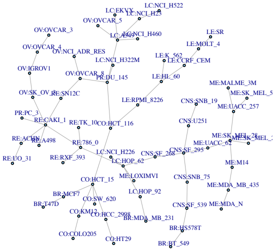
- Medical diagnosis [Huang et al., 2010]
  Given microarray profile of tissue sample x and the disease hierarchy $\mathcal{G}$, predict multiple labels in the hierarchy.



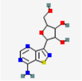


Application Areas (cont.)

- Drug bioactivity prediction [Su et al., 2010]
  Given molecular structure x and relation of cancer cell lines, predict the anti-cancer potentials of the molecules.



$$x \xrightarrow{\textit{predict}} \mathbf{y} = \{y_1, y_2, \cdots, y_{60}\}, y_i \in \{0, 1\}.$$



1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1	0	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	0	1	1	1	1	1	1

blocks correspond to cell lines

A Collection of Single-Task Classifiers

- Reduce multiple tasks to a a bag of single takes and solved by a collection of single-task classifiers (e.g. SVM)

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{im} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix}, \quad Y = \begin{bmatrix} y_{11} & y_{12} & \dots & y_{1k} \\ \vdots & \vdots & \ddots & \vdots \\ y_{i1} & y_{i2} & \dots & y_{ik} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \dots & y_{nk} \end{bmatrix}$$

$$\{y_1, y_2, \cdots, y_T\} = F(x)$$

- Drawbacks:
  - Train a collection of classifiers separately
  - Assume tasks are independent

A Collection of Single-Task Classifiers

- Reduce multiple tasks to a a bag of single takes and solved by a collection of single-task classifiers (e.g. SVM)

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{im} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix}, \quad Y = \begin{bmatrix} y_{11} & \dots & \dots & \dots \\ \vdots & \vdots & \ddots & \vdots \\ y_{i1} & \dots & \dots & \dots \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & \dots & \dots & \dots \end{bmatrix}$$

$$\{y_1, \cdots, \cdots, \cdots\} = \{\mathbf{f_1}(\mathbf{x}), \cdots, \cdots, \cdots\}$$

- Drawbacks:
  - Train a collection of classifiers separately
  - Assume tasks are independent

A Collection of Single-Task Classifiers

- Reduce multiple tasks to a a bag of single takes and solved by a collection of single-task classifiers (e.g. SVM)

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{im} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix}, \quad Y = \begin{bmatrix} \dots & y_{12} & \dots & \dots \\ \vdots & \vdots & \ddots & \vdots \\ \dots & y_{i2} & \dots & \dots \\ \vdots & \vdots & \ddots & \vdots \\ \dots & y_{n2} & \dots & \dots \end{bmatrix}$$

$$\{\cdots, y_2, \cdots, \cdots\} = \{\cdots, \mathbf{f_2}(\mathbf{x}), \cdots, \cdots\}$$

- Drawbacks:
  - Train a collection of classifiers separately
  - Assume tasks are independent

A Collection of Single-Task Classifiers

- Reduce multiple tasks to a a bag of single takes and solved by a collection of single-task classifiers (e.g. SVM)

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{im} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix}, \quad Y = \begin{bmatrix} \dots & \dots & y_{1j} & \dots \\ \vdots & \vdots & \ddots & \vdots \\ \dots & \dots & y_{ij} & \dots \\ \vdots & \vdots & \ddots & \vdots \\ \dots & \dots & y_{nj} & \dots \end{bmatrix}$$

$$\{\cdots, \cdots, y_j, \cdots\} = \{\cdots, \cdots, \mathbf{f_j}(\mathbf{x}) \cdots\}$$

- Drawbacks:
  - Train a collection of classifiers separately
  - Assume tasks are independent

A Collection of Single-Task Classifiers

- Reduce multiple tasks to a a bag of single takes and solved by a collection of single-task classifiers (e.g. SVM)

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{im} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix}, \quad Y = \begin{bmatrix} \dots & \dots & \dots & y_{1T} \\ \vdots & \vdots & \ddots & \vdots \\ \dots & \dots & \dots & y_{jT} \\ \vdots & \vdots & \ddots & \vdots \\ \dots & \dots & \dots & y_{nT} \end{bmatrix}$$

$$\{\cdots, \cdots, \cdots, y_T\} = \{\cdots, \cdots, \cdots, \mathbf{f_T}(\mathbf{x})\}$$

- Drawbacks:
  - Train a collection of classifiers separately
  - Assume tasks are independent

Multi-Task Feature Learning [Argyriou et al., 2007]

- Sparse dimensionality reduction
  - Sparse PCA
  - Sparse ICA

Multi-Task Feature Learning

- Inspired by sparse modeling.
- Learning multiple related tasks *vs.* learning independently.
- Assumptions:
  - Few data per task, pooling data across related tasks.
  - Common underlying representation across tasks.
  - A small set of shared features across tasks (task relations).

Multi-Task Feature Learning

- Learning Paradigm
  - Task index $t = 1, \cdots, T$
  - m examples for each task $(x_1, y_{t,1}), \cdots, (x_m, y_{t,m}) \in \mathcal{R}^d \times \mathcal{R}$
  - d-dimension features for each example
    $H(x) = h_1(x), \cdots, h_d(x)$
- Predict via a linear function $f_t : \mathcal{R}^d \to \mathcal{R}$ with the form

$$f_t(x) = \sum_{i=1}^{d} a_{i,t} h_i(x)$$

MTL - Coefficient Matrix

- Task specific coefficients can be arranged as a matrix

$$A = \begin{pmatrix} a_{1,1} & \cdots & a_{1,T} \\ \vdots & \ddots & \vdots \\ a_{d,1} & \cdots & a_{d,T} \end{pmatrix}$$

- $a_{i,t}$ reveals feature importance and task organization
- Desiderata
  - a low dimensional data representation shared across tasks (few features)
  - importance of each feature is also preserved across tasks (similar feature weight)

MTL - Regularization

- A should be a matrix with property
  - most elements being zero (sparsity)
  - for task t, $|a_{i,t}|$ being similar (uniformity)
- $(2,1)$-norm regularization

$$||A||_{2,1} = \sum_{i=1}^{d} \sqrt{\sum_{t=1}^{T} a_{i,t}^2}$$

  - first compute l_2 norm of the rows $||\mathbf{a_1}||_2, \cdots, ||\mathbf{a_d}||_2$
  - then compute the sum of result vectors $\sum_{i=1}^{d} ||\mathbf{a_i}||_2$

MTL - Learning Objective

- (2, 1)-norm $||A||_{2,1} = \sum_{i=1}^{d} \sqrt{\sum_{t=1}^{T} a_{i,t}^2}$ to be small
  - small l_2 norm favors uniformity
  - small l_1 norm favors sparsity
- Learning objective function

$$\min \left\{ \sum_{t=1}^{T} \sum_{j=1}^{m} \mathcal{L}(y_{t,j}, \sum_{i=1}^{d} a_{i,t} h_i(x_{t,j})) + \gamma ||A||_{2,1}^2 \right\}$$

  - minimize error ($\mathcal{L}$ is loss function)
  - minimize $||A||_{2,1}^2$
  - the number of features decreases with γ

MTL - Single Task variation

- Single task variation

$$\min \left\{ \sum_{j=1}^{m} \mathcal{L}(y_j, \sum_{i=1}^{d} a_i h_i(x_j)) + \gamma ||\mathbf{a}||_1^2 \right\}$$

  - $||\mathbf{a}_1||$ is l_1 norm, number of non-zero entries of $\mathbf{a}$
  - Many entries will be 0 (sparse model)

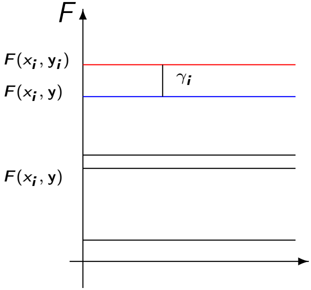
Max-Margin Condition Random Field [Rousu et al., 2006b]

- What is margin
  - Confidence of assigning an example to a label.
- Maximum margin learning
  - Define a compatibility score $F(x_i, \mathbf{y}_i; \mathbf{w})$
  - Define a separation margin for $(x_i, \mathbf{y}_i)$

$$\gamma_i = F(x_i, \mathbf{y}_i; \mathbf{w}) - \max_{\mathbf{y} \neq \mathbf{y}_i} F(x_i, \mathbf{y}; \mathbf{w})$$

  - Maximize the minimum margin

$$\max \min_i \gamma_i$$

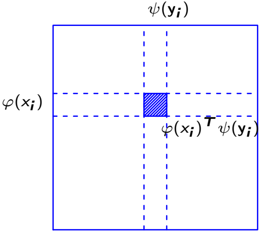


Max-Margin Conditional Random Field

Learning Paradigm

- Learning Paradigm
  - Data from a domain $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k, \mathcal{Y}_j \in \{+1, -1\}$.
    - Training data is given as $\{(x_i, \mathbf{y}_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$.
    - A *pseudo-example* is $(x_i, \mathbf{y})$ where $\mathbf{y}$ is an arbitrary multilabel.
  - Markov network $\mathcal{G} = (\mathcal{E}, \mathcal{V})$ over multiple labels.
- Joint feature map
- Map input and output to a *joint feature map* $\phi(x, \mathbf{y})$:

$$\phi(x, \mathbf{y}) = \varphi(x) \otimes \psi(\mathbf{y}).$$



Max-Margin Conditional Random Field

Compatibility Function

- Model family
  - Exponential family defined on Markov network $\mathcal{G} = (\mathcal{E}, \mathcal{V})$:

$$P(\mathbf{y}|x) = \frac{1}{Z(x, \mathbf{w})} \prod_{e \in \mathcal{E}} \exp(\mathbf{w}_e^T \phi_e(x, \mathbf{y}_e)),$$

    which leads to a log linear model:

$$\log P(\mathbf{y}|x) = \mathbf{w}^T \phi(x_i, \mathbf{y}_i) - \log Z(x, \mathbf{w}).$$

Max-Margin Conditional Random Field (cont.)

Compatibility Function

- Margin-based learning:
  - Use margin-based learning to avoid partition function $Z(\mathbf{w}, x)$
  - Related to odds-ratio learning:

$$\frac{\log P(\mathbf{y}_i|x_i)}{\log P(\mathbf{y}|x_i)} = \mathbf{w}^T\phi(x_i, \mathbf{y}_i) - \mathbf{w}^T\phi(x_i, \mathbf{y})$$

  Maximize the margin between real example $\phi(x_i, \mathbf{y}_i)$ and all the pseudo-example $\phi(x_i, \mathbf{y})$, according to $\ell_\Delta(\mathbf{y}_i, \mathbf{y})$.

- Compatibility Function

$$F(x_i, \mathbf{y}_i; \mathbf{w}) = \mathbf{w}^T\phi(x_i, \mathbf{y}_i)$$

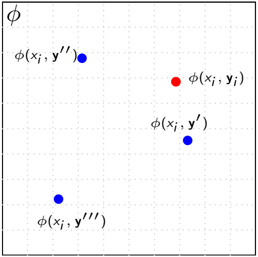
Max-Margin Conditional Random Field

Maximum Margin Learning

- Maximize the margin of difference in compatibility score between correct pair and incorrect pair
- Optimization problem (quadratic programming)

$$(\mathbf{w}) = \underset{w,\xi \geq 0}{\mathbf{argmin}} \left(\frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i \right)$$

$$\text{s.t. } \mathbf{w}^T \phi(x_i, \mathbf{y}_i) - \mathbf{w}^T \phi(x_i, \mathbf{y}) \geq \ell_\Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i, \ \forall x_i, \mathbf{y}.$$



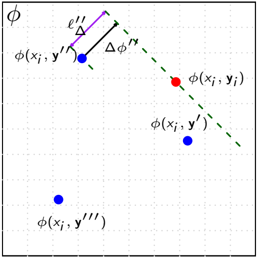
Max-Margin Conditional Random Field

Maximum Margin Learning

- Maximize the margin of difference in compatibility score between correct pair and incorrect pair
- Optimization problem (quadratic programming)

$$(\mathbf{w}) = \underset{w,\xi\geq 0}{\mathbf{argmin}} \left(\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \right)$$

$$\textbf{s.t. } \mathbf{w}^T\phi(x_i,\mathbf{y}_i) - \mathbf{w}^T\phi(x_i,\mathbf{y}) \geq \ell_{\Delta}(\mathbf{y}_i,\mathbf{y}) - \xi_i,\ \forall x_i, \mathbf{y}.$$



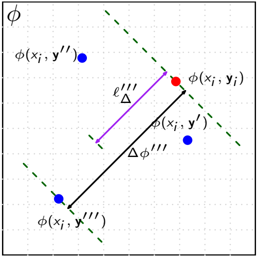
Max-Margin Conditional Random Field

Maximum Margin Learning

- Maximize the margin of difference in compatibility score between correct pair and incorrect pair
- Optimization problem (quadratic programming)

$$
(\mathbf{w}) = \underset{w,\xi \geq 0}{\mathbf{argmin}} \left(\frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i \right)
$$

$$
\textbf{s.t. } \mathbf{w}^T \phi(x_i, \mathbf{y}_i) - \mathbf{w}^T \phi(x_i, \mathbf{y}) \geq \ell_\Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i, \ \forall x_i, \mathbf{y}.
$$



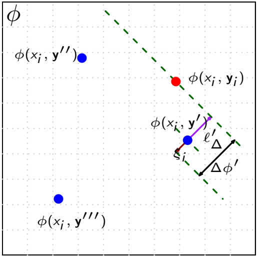
Max-Margin Conditional Random Field

Maximum Margin Learning

- Maximize the margin of difference in compatibility score between correct pair and incorrect pair
- Optimization problem (quadratic programming)

$$(\mathbf{w}) = \underset{w,\xi \geq 0}{\mathbf{argmin}} \left(\frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i \right)$$

$$\mathbf{s.t.}\ \mathbf{w}^T \phi(x_i, \mathbf{y}_i) - \mathbf{w}^T \phi(x_i, \mathbf{y})$$
$$\geq \ell_\Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i,\ \forall x_i, \mathbf{y}.$$

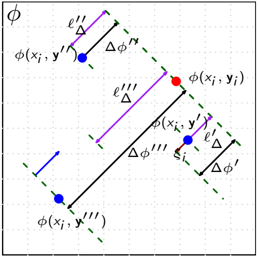


- Maximize the margin of difference in compatibility score between correct pair and incorrect pair
- Optimization problem (quadratic programming)

$$(\mathbf{w}) = \underset{w,\xi \geq 0}{\mathbf{argmin}} \left(\frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i \right)$$

$$\textbf{s.t. } \mathbf{w}^T \phi(x_i, \mathbf{y}_i) - \mathbf{w}^T \phi(x_i, \mathbf{y})$$

$$\geq \ell_\Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i, \; \forall x_i, \mathbf{y}.$$



Max-Margin Conditional Random Field

Prediction

- Prediction is made by

$$\mathbf{y}^* = \underset{\mathbf{y} \in \mathcal{Y}}{\mathbf{argmax}}\, \mathbf{w}^T \phi(x_i, \mathbf{y}_i)$$

Bibliography

Argyriou, A., Evgeniou, T., and Pontil, M. (2007).
Multi-task feature learning.
In *Advances in Neural Information Processing Systems 19*. MIT Press.

Breiman, L. (1994).
Bagging predictors.
Technical Report 421, Department of Statistics, UC Berkeley.
ftp://ftp.stat.berkeley.edu/pub/tech-reports/421.ps.Z.

Cortes, C. and Vapnik, V. (1995).
Support vector networks.
Machine Learning, 20:273 – 297.

Bibliography (cont.)

Freund, Y. (1992).
An improved boosting algorithm and its implications on learning complexity.
In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 391 – 398. ACM Press, New York, NY.

Freund, Y. and Schapire, R. E. (1996).
Experiments with a new boosting algorithm.
In *Proceedings of the International Conference on Machine Learing*, pages 148 – 146. Morgan Kaufmann Publishers.

Ghani, R. (2000).
Using error-correcting codes for text classification.
In *Proc. 17th International Conf. on Machine Learning*, pages 303 – 310. Morgan Kaufmann, San Francisco, CA.

Bibliography (cont.)

Ho, T. K. (1998).
The random subspace method for constructing decision forests.
Pattern Analysis and Machine Intelligence, IEEE Transactions on, 20(8):832 –844.

Huang, H., Liu, C.-C., and Zhou, X. J. (2010).
Bayesian approach to transforming public gene expression repositories into disease diagnosis databases.
Proceedings of the National Academy of Sciences, 107(15):6823–6828.

Bibliography (cont.)

Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J. (2006a).
Kernel-based learning of hierarchical multilabel classi?cation models.
Journal of Machine Learning Research, 7:1601–1626.

Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J. (2006b).
Kernel-Based Learning of Hierarchical Multilabel Classification Models.
The Journal of Machine Learning Research, 7:1601–1626.

Bibliography (cont.)

Schapire, R. and Singer, Y. (1998).
Improved boosting algorithms using confidence-rated predictions.
In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 80 – 91, New York. ACM Press.

Su, H., Heinonen, M., and Rousu, J. (2010).
Structured output prediction of anti-cancer drug activity.
In *Proceedings of the 5th IAPR international conference on Pattern recognition in bioinformatics*, PRIB'10, pages 38–49, Berlin, Heidelberg. Springer-Verlag.

Bibliography (cont.)

Veeramachaneni, S., Sona, D., and Avesani, P. (2005).
Hierarchical dirichlet model for document classification.
In *Proceedings of the 22nd international conference on Machine learning*, pages 928–935. ACM.