http://goo.gl/Jv7Vj9

Course website

KYOTO UNIVERSITY

Statistical Machine Learning Theory

From Multi-class Classification to Structured Output Prediction

Hisashi Kashima
kashima@i.Kyoto-u.ac.jp

DEPARTMENT OF INTELLIGENCE SCIENCE AND TECHNOLOGY

Topics of the 2nd half of this course: Advanced supervised learning and unsupervised learning

- Multi-class classification and structured output prediction
- Other variants of supervised learning problems:
  - Semi-supervised learning, active learning, & transfer learning
- On-line learning:
  - Follow the leader, on-line gradient descent, perceptron
  - Regret analysis
- Sparse modeling:
  - L_1 regularization, Lasso, & reduced rank regression
- Model evaluation

Homework

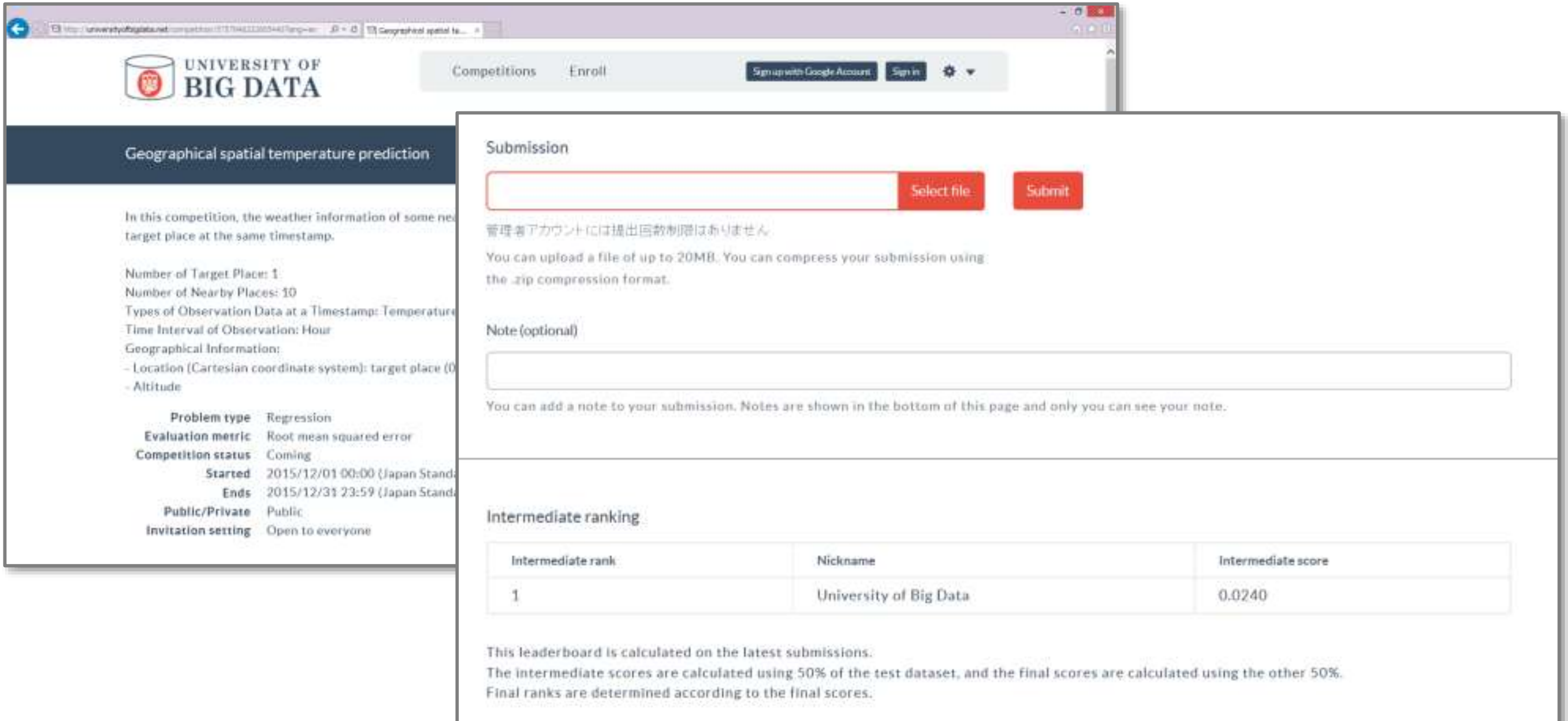
Homework:
Supervised regression

- Work on a supervised regression problem:
  1. Implement at least one linear regression by yourself
  2. Use publicly available nonlinear regression implementations
- Participate into a competition at http//universityofbigdata.net
  - Register with your Google account (if you have not yet)

  – The competition will last until Dec. 31th
  – Submit your predictions at least twice
    (once with your implementation; once with another)

Submitting your prediction: http://goo.gl/3BMpf4

- See the instructions at http//universityofbigdata.net/competition/5757948332605440?lang=en



Report submission:
Submit a report summarizing your work

- Submission:
  - Due: Jan. 7th noon, 2016
  - Send your report to kashipong+report@gmail.com with subject “SML2015 competition report” and confirm you receive an ack before 8th
- Report format:
  - Must include:
    - Brief description of your implementation (not sourcecode)
    - Your approach, analysis pipeline, results, and discussions
  - At least 3 pages, but do not exceed 6 pages in LNCS format

Topics of the 2nd half of this course: Advanced supervised learning and unsupervised learning

- ***Multi-class classification and structured output prediction***
- Other variants of supervised learning problems:
  - Semi-supervised learning, active learning, & transfer learning
- On-line learning:
  - Follow the leader, on-line gradient descent, perceptron
  - Regret analysis
- Sparse modeling:
  - L_1 regularization, Lasso, & reduced rank regression
- Model evaluation

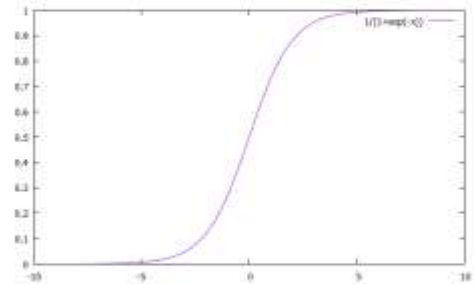
Multi-class Classification

Multi-class classification: Generalization of supervised two-class classification

- Training dataset: $\{(\boldsymbol{x}^{(1)}, y^{(1)}), \dots, (\boldsymbol{x}^{(i)}, y^{(i)}), \dots, (\boldsymbol{x}^{(N)}, y^{(N)})\}$
  - input $\boldsymbol{x}^{(i)} \in \mathcal{X} = \mathbb{R}^D$: D-dimensional real vector
  - output $y^{(i)} \in \mathcal{Y}$: one-dimensional scalar
- Estimate a *deterministic mapping* $f: \mathcal{X} \to \mathcal{Y}$ (often with a confidence value) or a *conditional probability* $P(y|\boldsymbol{x})$
- Classification
  - $\mathcal{Y} = \{+1, -1\}$: Two-class classification
  - $\mathcal{Y} = \{1, 2, \dots, K\}$: K-class classification
    - hand-written digit recognition, text classification, …

Two-class classification model: One model with one model parameter vector

- Two-class classification model
  - Linear classifier: $f(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x}) \in \{+1, -1\}$
  - Logistic regression: $P(y|\boldsymbol{x}) = \dfrac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x})}$



  - The model is specified by the parameter vector $\boldsymbol{w} = (w_1, w_2, \dots, w_D)^\top$
- Our goal is find the parameter $\widehat{\boldsymbol{w}}$ by using the training dataset $\{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \dots, (\boldsymbol{x}^{(N)}, y^{(N)})\}$
  - Generalization: accurate prediction for future data sampled from some underlying distribution $\mathcal{D}_{\boldsymbol{x},y}$

Simple approaches to multi-class classification: Reduction to two-class classification

- Reduction to a set of two-class classification problems
- Approach 1: One-versus-rest
  - Construct K two-class classifiers; each classifier $\mathrm{sign}(\boldsymbol{w}^{(k)\top}\boldsymbol{x})$ discriminates class k from the others
  - Prediction: the most probable class with the highest $\boldsymbol{w}^{(k)\top}\boldsymbol{x}$

confidence

- Approach 2: One-versus-one
  - Construct $K(K-1)/2$ two-class classifiers, each of which discriminates between a pair of two classes
  - Prediction by voting

Error Correcting Output Code (ECOC) : An approach inspired by error correcting coding

- Approach 3: Error correcting output code (ECOC)
  - Construct a set of two-class classifiers, each of which discriminates between two groups of classes, e.g. AB vs. CD
  - Prediction by finding the nearest code in terms of Hamming distance

codes

class	two-class classification problems					
	1	2	3	4	5	6
A	1	1	1	1	1	1
B	1	-1	1	-1	-1	-1
C	-1	-1	-1	1	-1	1
D	-1	1	1	-1	-1	1
prediction	**1**	**1**	**1**	**1**	**1**	**-1**

code for class A

Design of ECOC :
Code design is the key for good classification

- Codes (row) should be apart from each other in terms of Hamming distance

codes

class	two-class classification problems					
	1	2	3	4	5	6
A	1	1	1	1	1	1
B	1	-1	1	-1	-1	-1
C	-1	-1	-1	1	-1	1
D	-1	1	1	-1	-1	1

Hamming distances between codes

class	A	B	C	D
A	0	4	4	3
B		0	4	3
C			0	3
D				0

Multi-class classification model: One model parameter vector for each class

- More direct modeling of multi-class classification
  - One parameter vector $\boldsymbol{w}^{(k)}$ for each class k
  - Multi-class linear classifier: $f(\boldsymbol{x}) = \underset{k \in \mathcal{Y}}{\operatorname{argmax}} \, \boldsymbol{w}^{(k)\top}\boldsymbol{x}$
  - Multi-class logistic regression: $P(k|\boldsymbol{x}) = \dfrac{\exp(\boldsymbol{w}^{(k)\top}\boldsymbol{x})}{\sum_{k' \in \mathcal{Y}} \exp(\boldsymbol{w}^{(k')\top}\boldsymbol{x})}$
    - converts real values into positive values, and then normalizes them to obtain a probability value $\in [0,1]$

Training multi-class classifier: Constraints for correct classification

- Training multiclass linear classifier: $f(\boldsymbol{x}) = \underset{k \in \mathcal{Y}}{\operatorname{argmax}} \boldsymbol{w}^{(k)\top}\boldsymbol{x}$
  - can use the one-versus-rest method, but not perfect
- Constraints for correct classification of training data

  $\boldsymbol{w}^{(y^{(i)})\top}\boldsymbol{x}^{(i)} > \boldsymbol{w}^{(k)\top}\boldsymbol{x}^{(i)}$ for $^{\forall}k \neq y^{(i)}$

  i.e. $\boldsymbol{w}^{(y^{(i)})\top}\boldsymbol{x}^{(i)} > \underset{k \in \mathcal{Y}, k \neq y^{(i)}}{\operatorname{argmax}} \boldsymbol{w}^{(k)\top}\boldsymbol{x}^{(i)}$
  - Learning algorithms find solutions satisfying (almost all) these constraints
    - Multi-class perceptron, multi-class SVM, …

Multi-class perceptron: Incremental learning algorithm of linear classifier

- Multi-class linear perceptron trains a classifier to meet the constraints $\boldsymbol{w}^{(y^{(i)})\top}\boldsymbol{x}^{(i)} > \max_{k\in\mathcal{Y}, y\neq y^{(i)}} \boldsymbol{w}^{(k)\top}\boldsymbol{x}^{(i)}$

- Algorithm:

1. Given $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)$, make a prediction with :
$$f\left(\boldsymbol{x}^{(i)}\right) = \operatorname*{argmax}_{k\in\mathcal{Y}} \boldsymbol{w}^{(k)\top}\boldsymbol{x}^{(i)}$$

2. Update parameters only when the prediction is wrong:
    1. $\boldsymbol{w}^{(y^{(i)})} \leftarrow \boldsymbol{w}^{(y^{(i)})} + \boldsymbol{x}^{(i)}$: reinforces correct prediction
    2. $\boldsymbol{w}^{\left(f\left(\boldsymbol{x}^{(i)}\right)\right)} \leftarrow \boldsymbol{w}^{\left(f\left(\boldsymbol{x}^{(i)}\right)\right)} - \boldsymbol{x}^{(i)}$: discourages wrong prediction

Training multi-class logistic regression: (Regularized) maximum likelihood estimation

- Find the parameters that minimizes the negative log-likelihood

$$J\left(\{\boldsymbol{w}^{(y)}\}_y\right) = -\sum_{i=1,\dots,N} \log p(y^{(i)}|\boldsymbol{x}^{(i)}) + \gamma \sum_{y\in\mathcal{Y}} \| \boldsymbol{w}^{(y)} \|_2^2$$

  – $\| \boldsymbol{w}^{(y)} \|_2^2$: a regularizer to avoid overfitting

- For multi-class logistic regression $P(k|\boldsymbol{x}) = \dfrac{\exp(\boldsymbol{w}^{(k)\top}\boldsymbol{x})}{\sum_{k'\in\mathcal{Y}} \exp(\boldsymbol{w}^{(k')\top}\boldsymbol{x})}$

$$J = -\sum_i \boldsymbol{w}^{(k)\top}\boldsymbol{x}^{(i)} + \sum_i \log \sum_{k'\in\mathcal{Y}} \exp\left(\boldsymbol{w}^{(k)\top}\boldsymbol{x}^{(i)}\right) + reg.$$

  – Minimization using gradient-based optimization methods

Difference of perceptron and ML estimation: Perceptron needs only max operation; ML needs sum

- Perceptron
  - Training & prediction need only $\operatorname{argmax}_{k \in \mathcal{Y}}$ operation
  - SVM also does
- (Regularized) maximum likelihood estimation
  - Training: needs $\sum_{k' \in \mathcal{Y}}$ operation
  - Prediction: needs $\operatorname{argmax}_{k \in \mathcal{Y}}$ operation

Equivalent form of multi-class logistic regression: Representation with one (huge) parameter vector

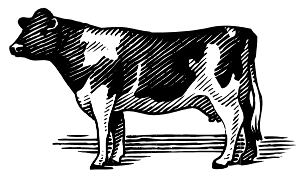
- Consider a joint feature space of $\boldsymbol{x}$ and y:
  - $\boldsymbol{\varphi}(\boldsymbol{x}, y) = (\delta(y=1)\boldsymbol{x}^\top, \delta(y=2)\boldsymbol{x}^\top, \dots, \delta(y=K)\boldsymbol{x}^\top)^\top$
  - Corresponding parameter vector:
$$\boldsymbol{w} = (\boldsymbol{w}^{(1)\top}, \boldsymbol{w}^{(2)\top}, \dots, \boldsymbol{w}^{(K)\top})^\top$$
  - KD-dimensional feature space

- Multiclass LR model: $P(y|\boldsymbol{x}) = \dfrac{\exp\left(\boldsymbol{w}^\top \boldsymbol{\varphi}(\boldsymbol{x},y)\right)}{\sum_{k' \in \mathcal{Y}} \exp(\boldsymbol{\varphi}(\boldsymbol{x},k'))}$
  - Equivalent to the previous model $P(k|\boldsymbol{x}) = \dfrac{\exp\left(\boldsymbol{w}^{(k)\top}\boldsymbol{x}\right)}{\sum_{k' \in \mathcal{Y}} \exp(\boldsymbol{w}^{(k')\top}\boldsymbol{x})}$
  - Useful when we consider structured output prediction

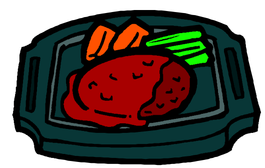
Structured Output Prediction

Ultimate predictive modeling:
Learn a mapping between general sets

- In supervised learning, what we want is a mapping $f: \mathcal{X} \to \mathcal{Y}$
  - $\mathcal{X} = \mathbb{R}^D, \mathcal{Y} = \mathbb{R}$ (regression) or a discrete set (classification)
- Ultimate predictor should take arbitrary $\mathcal{X}$ and $\mathcal{Y}$ sets



$\xrightarrow{f}$

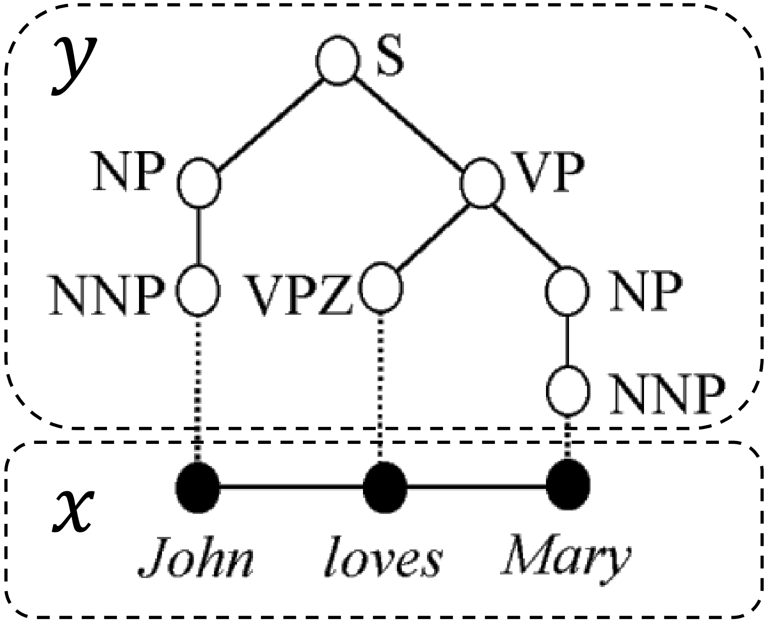


x $\qquad\qquad$ y

- We have to restrict the classes of $\mathcal{X}$ and $\mathcal{Y}$ in practice
  - Especially, cases with general output spaces are difficult to consider in the current framework
    - Classification with an infinite number of classes

Structured output prediction: Outputs are sequences, trees, and graphs

- (Inputs and) outputs have complex structures such as sequences, trees, and graphs in many applications
  - Natural language processing: texts, parse trees, …
  - Bioinformatics: sequences and structures of DNA/RNA/proteins
- Structured output prediction tasks:
  - Syntactic parsing: sequences to trees
    - $x = (John, loves, Mary)$: sequence
    - $y = (\mathrm{S}(\mathrm{NP}(\mathrm{NNP}))(\mathrm{VP}(\mathrm{VPZ})(\mathrm{NP}(\mathrm{NNP}))))$: tree



http://en.wikipedia.org/wiki/Treebank#mediaviewer/File:Example-tree.png

Sequence labeling:
Structured prediction with sequential input & output

- Sequence labeling gives a label to each element of a sequence
  - $x = (x_1, x_2, \dots, x_T)$: input sequence of length T
  - $y = (y_1, y_2, \dots, y_T)$: output sequence with the same length
  - Simplest structured prediction problem

x_1	x_2	…	x_T
y_1	y_2	…	y_T

- Example. Part-of-speech tagging gives a part-of-speech tag to each word in a sentence
  - x: sentence (a sequence of words)
  - y: Part-of-speech tags (e.g. *noun*, *verb*,…)

Sequence labeling as multi-class classification: Impossible to work with exponentially many parameters

- Formulation as T independent classification problems
  - Predict y_t using surrounding words $(\ldots, x_{t-1}, x_t, x_{t+1}, \ldots)$
    - Sometimes quite works well and efficient
  - No guarantee of consistence among predicted labels
    - Might want to include dependencies among labels such as "a verb is likely to follow nouns"
- This problem can also be considered as one multi-class classification problem with K^T classes
  - $f(x) = \operatorname*{argmax}_{k \in \mathcal{Y}} \boldsymbol{w}^{(k)\top} \boldsymbol{x}$ is almost impossible to work with exponentially many parameters

Key for solving structured output prediction: Formulation as a validation problem of in/output pairs

- Remember another form of multi-class classifier using the joint feature space
  - $P(y|x) = \dfrac{\exp\left(\boldsymbol{w}^\top \boldsymbol{\varphi}(x,y)\right)}{\sum_{k' \in \mathcal{Y}} \exp\left(\boldsymbol{\varphi}(x,k')\right)}$ or $f(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \boldsymbol{w}^\top \boldsymbol{\varphi}(x, y)$
  - They evaluate the affinity of an input-output pair
- Still the problem is not solved….
  but we can consider reducing the dimensionality of $\boldsymbol{\varphi}(x, y)$
  - Because the dimensionality of $\boldsymbol{\varphi}(x, y)$ is still huge

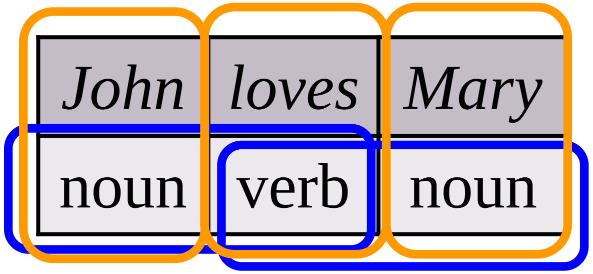
Features for sequence labeling: First-order Markov assumption gives two feature types

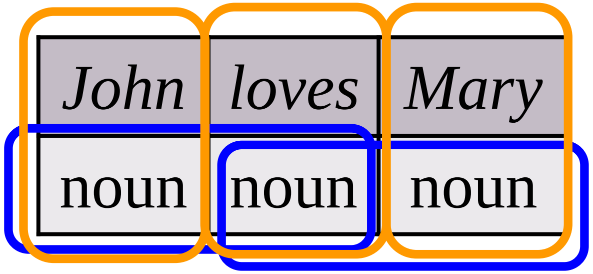
- Two types of features for sequence labeling
  1. Combination of one input label x_t and one output label y_t
     - Standard feature for multi-class classification
     - e.g. x_t="*loves*" $\wedge$ y_t="verb"
  2. Combination of two consecutive labels y_{t-1} and y_t
     - Markov assumption of output labels
     - e.g. y_{t-1}="noun" $\wedge$ y_t="verb"

x_1	x_2	…	$x_{t\text{-}1}$	x_t	…	x_T
y_1	y_2	…	$y_{t\text{-}1}$	y_t	…	y_T

Feature vector definition:
The numbers of appearance of each pattern

- Each dimension of $\boldsymbol{\varphi}(x, y)$ is defined as the number of appearance of each pattern in the joint sequence (x, y), e.g.
  - $\varphi(x, y)_1 =$ #appearance of [x_t="*loves*" $\wedge$ y_t="verb"]
  - $\varphi(x, y)_2 =$ #appearance of [y_{t-1}="noun" $\wedge$ y_t="verb"]
  - Features for all possible combination of POS tags and words





Impact of first-order Markov assumption: Reduced dimensionality of feature space

- Dimensionality of a feature vector was decreased from $O(K^T)$ to $O(K^2)$ (K is the number of labels for each position)
- Space problem was solved; we can calculate $\boldsymbol{w}^\top \boldsymbol{\varphi}(x, y)$
  - Prediction problem (i.e. $\underset{y \in \mathcal{Y}}{\operatorname{argmax}} \boldsymbol{w}^\top \boldsymbol{\varphi}(x, y)$) has not been solved
  - For sequential labeling, this can be done by using dynamic programming

Structured perceptron : Simple structured output learning algorithm

- Structured perceptron learns $\boldsymbol{w}$ satisfying

$$\boldsymbol{w}^\top \boldsymbol{\varphi}\left(x^{(i)}, y^{(i)}\right) > \max_{y \in \mathcal{Y}, y \neq y^{(i)}} \boldsymbol{w}^\top \boldsymbol{\varphi}\left(x^{(i)}, y\right)$$

- Algorithm:

1. Given $\left(x^{(i)}, y^{(i)}\right)$, make a prediction with :

$$f\left(\boldsymbol{x}^{(i)}\right) = \operatorname*{argmax}_{y \in \mathcal{Y}} \boldsymbol{w}^\top \boldsymbol{\varphi}\left(x^{(i)}, y\right)$$

2. Update parameters only when the prediction is wrong

$$\boldsymbol{w}^{\mathrm{NEW}} \leftarrow \boldsymbol{w}^{\mathrm{OLD}} + \boldsymbol{\varphi}\left(x^{(i)}, y^{(i)}\right) - \boldsymbol{\varphi}\left(x^{(i)}, f\left(x^{(i)}\right)\right)$$

– Prediction can be done in polynomial time by using dynamic programming for sequence labeling

Conditional random field: Conditional probabilistic model for structured prediction

- Conditional random filed: conditional probabilistic model

$$P(y|x) = \frac{\exp\left(\boldsymbol{w}^\top \boldsymbol{\varphi}(x, y)\right)}{\sum_{k' \in \mathcal{Y}} \exp\left(\boldsymbol{\varphi}(x, k')\right)}$$

- ML estimation needs the sum over all possible outputs

$$J = \sum_i \boldsymbol{w}^\top \boldsymbol{\varphi}\left(x^{(i)}, y^{(i)}\right) - \sum_i \log \sum_{y \in \mathcal{Y}} \boldsymbol{w}^\top \boldsymbol{\varphi}\left(x^{(i)}, y\right) + reg.$$

  – The sum can be taken with dynamic programming

Perceptron vs. CRF: Perceptron needs only max operation; ML needs sum

- Just like in multi-class classification,
  - Structured perceptron can work only with argmax operation
  - Maximum likelihood estimation also needs sum operation
- There are some structured output problems where argmax operation is easy but sum operation is difficult
  - e.g. bipartite matching