Transfer Learning Using Feature Selection

Paramveer S. Dhillon

Computer and Information Science
University of Pennsylvania, Philadelphia, PA, U.S.A

May 2, 2009

Outline

Transfer Learning

- Classical supervised learning algorithms learn a model for a task in isolation using labeled data.

Transfer Learning

- Classical supervised learning algorithms learn a model for a task in isolation using labeled data.
- But, labeled data is limited and very expensive to obtain.
- An attractive alternative is to use other sources of data to improve performance, for e.g. Semi -Supervised Learning.

Or...

Transfer Learning

- Classical supervised learning algorithms learn a model for a task in isolation using labeled data.
- But, labeled data is limited and very expensive to obtain.
- An attractive alternative is to use other sources of data to improve performance, for e.g. Semi -Supervised Learning.

Or...

- Use labeled data from other “similar” learning tasks and build shared models
- In other words “borrow strength” by sharing information between the models for related tasks – Transfer Learning (Thrun, ’96; Caruana, 97; Baxter, 97; . . .)

Our Contribution

- We provide three related ways of doing Transfer by using feature selection; i.e. sharing features across multiple related tasks.

Our Contribution

- We provide three related ways of doing Transfer by using feature selection; i.e. sharing features across multiple related tasks.
- All the three models are based on the information theoretic Minimum Description Length (MDL) principle and also share the same underlying Bayesian representation.

Our Contribution

- We provide three related ways of doing Transfer by using feature selection; i.e. sharing features across multiple related tasks.
- All the three models are based on the information theoretic Minimum Description Length (MDL) principle and also share the same underlying Bayesian representation.
- Two of the models (i.e. MIC and TPC) require the tasks to share the same set of features and do “joint feature selection” for these tasks.

Our Contribution

- We provide three related ways of doing Transfer by using feature selection; i.e. sharing features across multiple related tasks.
- All the three models are based on the information theoretic Minimum Description Length (MDL) principle and also share the same underlying Bayesian representation.
- Two of the models (i.e. MIC and TPC) require the tasks to share the same set of features and do “joint feature selection” for these tasks.
- The third model (Transfer-TPC) is more general:
  - It even works for tasks that have different amounts of labeled data.
  - It also allows us to define an arbitrary “similarity metric” between the various tasks.

Word Sense Disambiguation: An Example Problem

- WSD exhibits all three kinds of transfer.
- Each word has multiple labels (multiple senses) like:
  - For e.g. fire (a gun); fire (a person) or fire (your mouth off)
    - These are likely to share features (MIC Setting).

Word Sense Disambiguation: An Example Problem

- WSD exhibits all three kinds of transfer.
- Each word has multiple labels (multiple senses) like:
  - For e.g. fire (a gun); fire (a person) or fire (your mouth off)
    - These are likely to share features (MIC Setting).
- Each word has multiple feature classes:
  - Preceeding word; preceeding POS (Part of Speech); Topic of the document etc.
  - Features in the same class are likely to be in or out of the model together (TPC Setting)

Word Sense Disambiguation: An Example Problem

- WSD exhibits all three kinds of transfer.
- Each word has multiple labels (multiple senses) like:
  - For e.g. fire (a gun); fire (a person) or fire (your mouth off)
    - These are likely to share features (MIC Setting).
- Each word has multiple feature classes:
  - Preceeding word; preceeding POS (Part of Speech); Topic of the document etc.
  - Features in the same class are likely to be in or out of the model together (TPC Setting)
- Different words have different amounts of data and share some senses [Ando CoNLL'06]:
  - fire/discharge; fire/dismiss
  - These are likely to share similar features (Transfer-TPC Setting).

General Methodology

- All three models use MDL based coding scheme to specify a penalized likelihood method.
- The Description Length can be written as:

$$S = S_E + S_M$$

$S_E \longmapsto$ # Bits for encoding the residual errors given the model.
$S_M \longmapsto$ # Bits for encoding the model

General Methodology

- All three models use MDL based coding scheme to specify a penalized likelihood method.
- The Description Length can be written as:

$$S = S_E + S_M$$

$S_E \longmapsto$ # Bits for encoding the residual errors given the model.
$S_M \longmapsto$ # Bits for encoding the model

- Reduction in Description Length by adding a feature 'j' to the model:

$$\Delta S_j = \Delta S_{jE} - \Delta S_{jM}$$

- The goal is to maximize ΔS i.e. maximize the difference between the accuracy of the model (measured by likelihood) minus the cost of adding a new feature.

Simple ℓ_0 regression - "Baseline"

- Assume a simple linear regression model: $Y = WX + \epsilon$

Simple ℓ_0 regression - "Baseline"

- Assume a simple linear regression model: $Y = WX + \epsilon$
- In this case: $S_E = -\log\left(exp(-\frac{\sum_{i=1}^{n}(y_i - wx_i)^2}{2\sigma^2})\right)$
- $S_M = log(m) + 2$ i.e. Bits to code the feature (RIC Penalty) and its coefficient (AIC Penalty).

Simple ℓ_0 regression - "Baseline"

- Assume a simple linear regression model: $Y = WX + \epsilon$
- In this case: $S_E = -\log\left(exp(-\frac{\sum_{i=1}^{n}(y_i - wx_i)^2}{2\sigma^2})\right)$
- $S_M = log(m) + 2$ i.e. Bits to code the feature (RIC Penalty) and its coefficient (AIC Penalty).
- Equivalently we can write: $-log(P(f_i = 1)) = -\log\left(\frac{1}{m}\right)$; i.e. a uniform prior on each feature.

MIC : The Main Idea

- Addresses the problem of “joint feature selection” for related tasks.
- The tasks share the same set of features.

MIC : The Main Idea

- Addresses the problem of "joint feature selection" for related tasks.
- The tasks share the same set of features.
- Puts an ℓ_0 penalty on the features and an ℓ_0 on the tasks in which that feature is included. i.e. a (ℓ_0/ℓ_0) type penalty
- MIC induces sparsity at the level of features and at the level of tasks in which that feature is included.

Coding Schemes for MIC

- The goal is to maximize $\Delta S_j^k = \Delta S_{jE}^k - \Delta S_{jM}^k$ i.e. the reduction in TDL by adding a feature 'j' to a subset *k* of *h* tasks.

Coding Schemes for MIC

- The goal is to maximize $\Delta S_j^k = \Delta S_{jE}^k - \Delta S_{jM}^k$ i.e. the reduction in TDL by adding a feature 'j' to a subset k of h tasks.
- The cost to code S_{jE}^k i.e. the increase in likelihood by adding the feature to the model of k tasks is:
- $S_E = -\log(P(Y|X,w)$
  where
  $P(Y|X,w) = \frac{1}{\left((2\pi)^h|\Sigma|\right)^{\frac{n}{2}}} \exp\left(-\frac{1}{2}\sum_{i=1}^{n}\left[(y_i - wx_i)^T\Sigma^{-1}(y_i - wx_i)\right]\right)$
  Σ is the $h \times h$ covariance matrix.

Coding Schemes for MIC contd ...

- The cost to code the model is S_{jM}^{k} is $l_l + l_H + l_\theta$
- Cost to code:
  - The feature being included: $l_l = log(m)$.
  - The coefficient of the feature being included: $l_\theta = 2$.
  - How many and which of the h tasks have that feature $l_H = log(h) + log\binom{h}{k}$

Variations of MIC

- We can come up with more flexible coding schemes if we know that a feature will be included in “all” or “none” of the tasks [Full MIC].

Variations of MIC

- We can come up with more flexible coding schemes if we know that a feature will be included in “all” or “none” of the tasks [Full MIC].
- In this case we can save l_H bits used to code the subset of tasks which include the feature.
- Similar to the ℓ_1/ℓ_2 normed BBLasso [Obozinski et. al. ’09].

Variations of MIC

- We can come up with more flexible coding schemes if we know that a feature will be included in “all” or “none” of the tasks [Full MIC].
- In this case we can save l_H bits used to code the subset of tasks which include the feature.
- Similar to the ℓ_1/ℓ_2 normed BBLasso [Obozinski et. al. ’09].
- OR, we can code each task in isolation from each other [Independent MIC] (Same as the baseline).
- Here also we save l_H bits as above.

Experimental Setup

Name	# Tasks h	# Obs. n	# Features m	Source
Yeast Dataset	20	104	6715	[Litvin and Pe'er '09]
Breast Cancer	5	100	5000	[van't Veer et. al. '02]

- We compared the three versions of MIC (Partial, Full and Independent) against BBLasso [Obozinski et. al. '09] and AndoZhang [Ando et. al. '05]
- For BBLasso and AndoZhang we used their standard implementations from Berkeley Transfer Learning Toolkit.

Experiments contd...

Table: 5 fold CV accuracies. *Note:* AndoZhang's NA values are due to the fact that it does not explicitly select features.

Method	Test Error	# Features Selected.
	Yeast Dataset	
Partial MIC	$\mathbf{0.38 \pm 0.04}$	4 ± 0
Full MIC	0.39 ± 0.04	3 ± 0
Independent	0.41 ± 0.05	9 ± 1
AndoZhang	0.49 ± 0.03	NA
BBLasso	0.43 ± 0.03	63 ± 14

Experiments contd...

Table: 5 fold CV accuracies. *Note:* AndoZhang's NA values are due to the fact that it does not explicitly select features.

Method	Test Error	# Features Selected.
Breast Cancer Dataset		
Partial MIC	**0.33** $\pm$ **0.08**	2 ± 0
Full MIC	0.37 ± 0.08	2 ± 0
Independent	0.36 ± 0.08	2 ± 0
AndoZhang	0.44 ± 0.03	NA
BBLasso	**0.33** $\pm$ **0.08**	12 ± 4

Conclusion

- MIC gives flexible coding schemes for doing “joint feature selection” in related tasks.
- Significantly (5% level, paired t-test) better than AndoZhang, on Yeast and Breast Cancer datasets.
- Comparable in accuracy to BBLasso, but provides simpler and sparser models.

TPC : The Main Idea

- Does “joint feature selection” like MIC, but has a different notion of sharing of features.
- Here sharing occurs between features from the same Feature Class.

TPC : The Main Idea

- Does “joint feature selection” like MIC, but has a different notion of sharing of features.
- Here sharing occurs between features from the same Feature Class.
- Feature Classes compartmentalize the feature space as shown below:

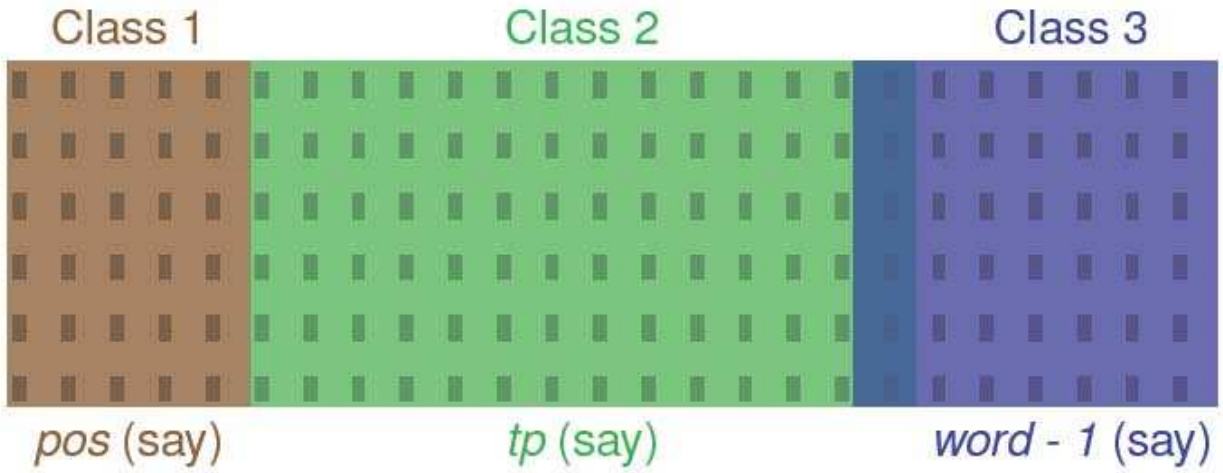


Figure: *Feature Classes in data* ($tp \longmapsto$ *topic of the document*, $pos \longmapsto$ *part of speech*, $word - 1 \longmapsto$ *previous word*)

TPC: The Main Idea

- Feature Classes differ in how dense they are in beneficial features.
- For e.g. when disambiguating various senses of the word Paris (i.e. Paris (France), Paris Hilton or Paris (Texas)), a feature class like topic of the document would contain more beneficial features than a feature class like # words in the document.

TPC: The Main Idea

- Feature Classes differ in how dense they are in beneficial features.
- For e.g. when disambiguating various senses of the word Paris (i.e. Paris (France), Paris Hilton or Paris (Texas)), a feature class like topic of the document would contain more beneficial features than a feature class like # words in the document.
- The main idea in TPC is to preferentially draw features from a feature class, once you have found it.
- As an analogy with MIC, the tendency in TPC is to include more and more features from the same feature class, as doing so is cheap.

Coding Schemes used in TPC

- $S_{jE} = -\log\left(exp(-\frac{\sum_{j=1}^{n}(y_j - wx_j)^2}{2\sigma^2})\right)$
- $S_{jM} = l_c + l_i + l_\theta$ (Three Part Coding)
- Coding cost for:
  - The index of the feature class of the feature $\longmapsto l_c$
  - The index of the feature within its feature class $\longmapsto l_i$
  - The coefficient of the feature $\longmapsto l_\theta$

TPC Coding Schemes contd ...

- If there are 'K' feature classes and 'Q' of them are currently in the model, then

$$l_C = \begin{cases} log(K) & \textit{if the feature class is not in the model} \\ log(Q) & \textit{if the feature class is already in the model} \end{cases}$$

- $l_l = log(m_k)$, where m_k = # Features in the k^{th} Feature Class (RIC or Bonferroni Penalty)
- $l_\theta = 2$ (AIC like penalty)

Analysis of TPC Scheme

- **TPC wins when most of the features lie in a small number of Feature Classes i.e. multiple 'good' features per Feature Class**.
- $[TPC - Baseline]_{bits} = (q - Q)log(\frac{K}{Q})$
  q $\longmapsto$ # Features Selected
  Q $\longmapsto$ # Feature Classes Selected
  K $\longmapsto$ Total # Feature Classes

Experiments

- WSD Datasets containing 172 verbs $\sim$ 7000 features each [Chen and Palmer '05].
- Feature Classes $\longmapsto$ *'tp', 'word-1', 'word+1' , 'pos' etc.* [# 45]

Experiments

- WSD Datasets containing 172 verbs $\sim$ 7000 features each [Chen and Palmer '05].
- Feature Classes $\longmapsto$ *'tp', 'word-1', 'word+1' , 'pos' etc.* [# 45]
- GSEA Datasets containing 5 different phenotypes and $\sim$ 10000 features each [Mootha et. al '03].
- Feature Classes $\longmapsto$ *Gene Classes* [# 318]

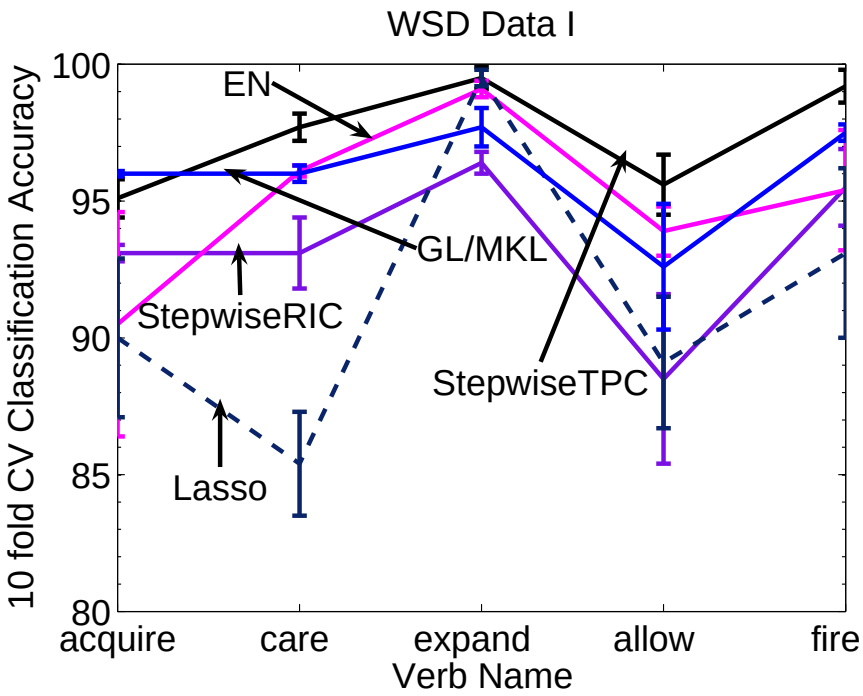
Experiments

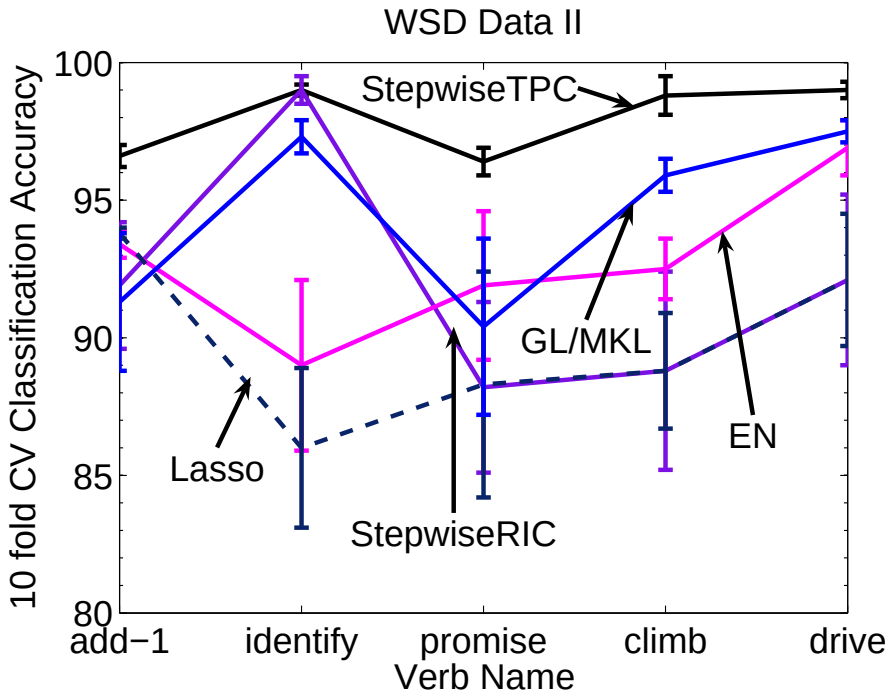
- We compare TPC to its standard counterparts i.e. Standard Stepwise Regression, Lasso [Tibshirani '96] [L_1 penalty] and Elastic Nets [Zou and Hastie '05] [$L_1 + L_2$ penalty], Group Lasso/ Multiple Kernel Learning [minimizes L_1/L_2 norm].

Experiments

- We compare TPC to its standard counterparts i.e. Standard Stepwise Regression, Lasso [Tibshirani '96] [L_1 penalty] and Elastic Nets [Zou and Hastie '05] [$L_1 + L_2$ penalty], Group Lasso/ Multiple Kernel Learning [minimizes L_1/L_2 norm].
- For Group Lasso/MKL we used their standard implementation as in SimpleMKL toolbox [Rakotomamonjy and Bach '08].
  - We used a set of 13 candidate kernels, consisting of 10 Gaussian Kernels (with bandwidths $\sigma = 0.5 - 20$) and 3 polynomial kernels (with degree 1-3) for each feature class as is done by [Rakotomamonjy et. al. '08].

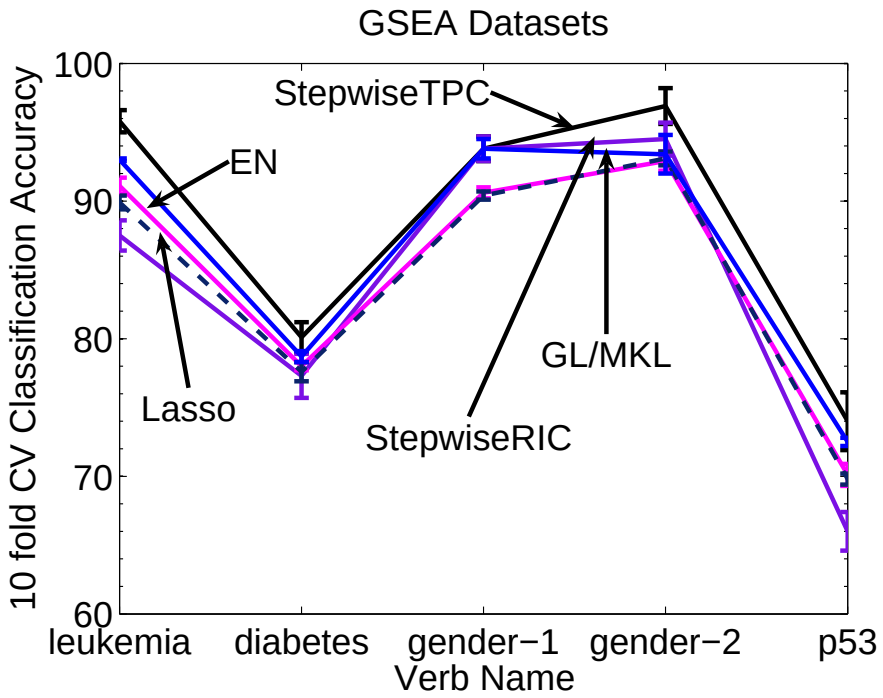
Results on WSD Data





- TPC significantly better on 7 verbs; has same accuracy as the best method on 2 verbs and slightly worse than GL/MKL on 1 ocassion.
- Overall TPC is significantly better or comparable than competing methods on 164 (160+4) /172 verbs.

Results on GSEA Data



- TPC significantly better on 4 datasets; has same accuracy as the best methods on the 5^{th} dataset.

Conclusion

- TPC is a penalized likelihood method
  - based on the MDL principle.
  - works best when good features are concentrated in a small number of features classes
  - outperforms competitors on WSD and GSEA datasets

Transfer-TPC: The main idea

- Does “sequential transfer” i.e. the task on which we want to transfer knowledge may not be known in advance.
- But we know that it is similar to other tasks according to some “similarity metric”.
- Transfer-TPC is most beneficial when we want to transfer knowledge between tasks which have unequal amounts of labeled data and they dont share the same set of features.
- As might be obvious from the name, Transfer-TPC uses TPC as a baseline model.

Transfer-TPC: The main idea

- If we know a feature is beneficial for a set of tasks; then it will also be beneficial for the (test-task) which is similar to the training tasks.
- Intuitively, we can think of putting a prior on the features and feature classes of the test-task based on evidence from similar tasks.

Transfer-TPC: The main idea

- If we know a feature is beneficial for a set of tasks; then it will also be beneficial for the (test-task) which is similar to the training tasks.
- Intuitively, we can think of putting a prior on the features and feature classes of the test-task based on evidence from similar tasks.
- As a result we will require fewer number of bits to code that feature.

Transfer-TPC formulation

- We have parameters θ_i and μ_j which denote the events of i^{th} feature class and j^{th} feature being included in standard TPC model i.e. [$\theta_i = p(fc_i = 1|\theta_i)$].
- Then posteriors over these parameters can be calculated by putting a Beta prior over them as:

$$p(\theta_i|\mathcal{D}_i) = p(\mathcal{D}_i|\theta_i) \times p(\theta_i|a, b)$$

where a, b are the hyperparameters of the Beta prior (conjugate of the Bernoulli likelihood) and $\mathcal{D}_i$ is the data for the i^{th} feature/feature class.

Transfer-TPC formulation

- We have parameters θ_i and μ_j which denote the events of i^{th} feature class and j^{th} feature being included in standard TPC model i.e. [$\theta_i = p(fc_i = 1|\theta_i)$].
- Then posteriors over these parameters can be calculated by putting a Beta prior over them as:

$$p(\theta_i|\mathcal{D}_i) = p(\mathcal{D}_i|\theta_i) \times p(\theta_i|a, b)$$

  where a, b are the hyperparameters of the Beta prior (conjugate of the Bernoulli likelihood) and $\mathcal{D}_i$ is the data for the i^{th} feature/feature class.
- The predictive distribution can be obtained as: $p(fc_i = 1|\mathcal{D}_i) = \int_0^1 p(fc_i = 1|\theta_i)p(\theta_i|\mathcal{D}_i)d\theta = \frac{k+a}{k+l+a+b}$; where k is the number of times that the i^{th} feature class/feature is selected and l is the complement of k, i.e. the number of times the i^{th} feature class/feature is not selected in the training data.

Transfer-TPC formulation

- The new cost of coding the model for the test-task is:

$$S_M = -\log\left(p(fc_i = 1|\mathcal{D}_i)\right) - \log\left(p(f_j = 1|\mathcal{D}_i)\right) + 2 \qquad (1)$$

Transfer-TPC formulation

- The new cost of coding the model for the test-task is:

$$S_M = -\log\left(p(fc_i = 1|\mathcal{D}_i)\right) - \log\left(p(f_j = 1|\mathcal{D}_i)\right) + 2 \qquad (1)$$

- If we choose the hyperparameters $a = 1$ and $b = K - 1$ where K is the total number of feature classes, then in the case of no transfer i.e. $k = l = 0$, the above equation reduces to the standard TPC scheme.

Transfer-TPC formulation

- The new cost of coding the model for the test-task is:

$$S_M = -\log\left(p(fc_i = 1|\mathcal{D}_i)\right) - \log\left(p(f_j = 1|\mathcal{D}_i)\right) + 2 \qquad (1)$$

- If we choose the hyperparameters $a = 1$ and $b = K - 1$ where K is the total number of feature classes, then in the case of no transfer i.e. $k = l = 0$, the above equation reduces to the standard TPC scheme.
- Intuitively we have relaxed the "harsh" RIC assumption of $p(f_i = 1) = \frac{1}{m}$; and replaced it with a more liberal assumption by transferring from similar tasks.

Experimental Setup

- Demonstrate Transfer-TPC by generalizing across words for Word Sense Disambiguation
- Set of 172 verbs [Schuler and Palmer '06], with varying number of senses and data.

Experimental Setup

- Demonstrate Transfer-TPC by generalizing across words for Word Sense Disambiguation
- Set of 172 verbs [Schuler and Palmer '06], with varying number of senses and data.
- We break the problem into disambiguation at the level of senses of a word rather than at the level of complete words.
- Since, most of the verb senses are singleton and may not be similar to other verb senses; we use Foreground- Background clustering [Kandylas et. al. '07] which puts all the singleton verbs into background.
- We do clustering by considering "semantic + syntactic" features, only "semantic features" and only "syntactic features".

Overview of our Approach

- Learn a separate “standard TPC” model for each sense of a word for distinguishing the different senses of that word (Note: In the baseline (TPC) case we show in experiments, the predicted sense is the one whose model gave the highest score.)
- Cluster word senses based on those features from their models that are positively correlated with those particular word senses. I.e., characterize each word sense by those features in its model that have positive regression coefficients.

Overview of our Approach

- Learn a separate "standard TPC" model for each sense of a word for distinguishing the different senses of that word (Note: In the baseline (TPC) case we show in experiments, the predicted sense is the one whose model gave the highest score.)
- Cluster word senses based on those features from their models that are positively correlated with those particular word senses. I.e., characterize each word sense by those features in its model that have positive regression coefficients.
- For each "target" verb sense to be predicted, use the selected features of other $(n-1)$ verb senses in its cluster to estimate the probability of the features being relevant for disambiguating the target verb sense.
- Combine all these "improved models" learnt at the sense level; and disambiguate the word as a whole by picking the predicted sense as the one whose model gave the best score.

Experimental Results

Table: 10-fold CV accuracies of various methods

Trans.-TPC(I),(II),(III),(IV)	Baseline TPC	SVM	Maj. Vote
86.29, 85.75, 85.11, 85.37	83.50	83.77	76.59

- In Settings 3 and 4 we did only “semantic” and only “syntactic” clustering respectively, in contrast to Settings 1 and 2 where we did (“semantic” + “syntactic”) clustering.
- In setting 2 we only transferred a prior on features, not on feature classes.

Conclusion

- Transfer -TPC provides an elegant way to introduce prior information from related tasks.
- In all the settings Transfer-TPC is significantly better than the competing methods (5% level, Paired t-test)
- The performance is affected little by introducing different linguistic effects into clustering.

Conclusion

- Transfer -TPC provides an elegant way to introduce prior information from related tasks.
- In all the settings Transfer-TPC is significantly better than the competing methods (5% level, Paired t-test)
- The performance is affected little by introducing different linguistic effects into clustering.
- There was 3.19-8.67% improvement in performance when the verb to which we were transferring had less data For example “kill” had 6 times more data than other verb senses in the same cluster i.e. “capture”, “arrest”, “strengthen” and “do” had 10 times more data than “die” and “save”.
- Even in the cases in which the verbs had similar amount of data we got 1.5- 2.5% increase in accuracy.
- In 8 out of 172 verbs we had negative transfer [Caruana ‘97].

A few anecdotes

- The positive features for the sense of word “fire” which means “to dismiss somebody from employment” were:
  {‘company’, ‘executive’, ‘friday’,‘hired’, ‘interview’, ‘job’, ‘named’,‘probably’,‘sharon’,’wally’, ’years’, ‘join’, ‘meet’, ‘replace’, ‘pay’,‘quoted’.... ‘VBD-VBN’, ‘VB-VP’,‘VB-VP-NP-PRP’, PRP’, ‘VBD-NP’... etc.}

A few anecdotes

- The positive features for the sense of word “fire” which means “to dismiss somebody from employment” were:
  {‘company’, ‘executive’, ‘friday’,‘hired’, ‘interview’, ‘job’, ‘named’,‘probably’,‘sharon’,’wally’, ’years’, ‘join’, ‘meet’, ‘replace’, ‘pay’,‘quoted’.... ‘VBD-VBN’, ‘VB-VP’,‘VB-VP-NP-PRP’, PRP’, ‘VBD-NP’... etc.}
- The positive features for the sense of word “fire” which means “to ignite something” were:
  {prehistoric’, ‘same’, ‘temperature’,‘israeli’, ‘palestinian’,‘incident’,‘months’, ‘retaliation’,‘showing’... ‘NNP-NP-S-VP-VBD’, ‘NNP-NP-S-VP-VBD’, ‘NNP’,‘NNP-NNP-VBD’,‘NNP-VBD’ ... etc.}

A few anecdotes

- A few of the clusters that we got: {‘back’, ‘stay’, ‘walk’, ‘step’}, {‘kill’, ‘capture’, ‘arrest’, ‘strengthen’, ‘release’}, {‘love’, ‘promise’, ‘turn’, ‘wear’}, {‘beat’, ‘respond’, ‘urge’} ...etc
- There were a total of 60-70 such clusters.

Connection between MIC and TPC

- Both of them do “joint feature selection” and expect the tasks to share the same set of features.
- Both put coefficients into classes, but the key difference is that in MIC the coefficient class is the set of coefficients of a single feature in all tasks, while in TPC, we have coefficient of multiple features from the same feature class.
- In MIC the tendency is to put a feature into models of more and more tasks, while in TPC the tendency is to put more and more features from the same feature class.

Connection between TPC and Transfer-TPC

- TPC is the basic building block on which Transfer-TPC is based and in the case of no transfer they are equivalent.
- Transfer-TPC generalizes the concept of TPC to incorporate priors from related tasks.

Connection between TPC and Transfer-TPC

- TPC is the basic building block on which Transfer-TPC is based and in the case of no transfer they are equivalent.
- Transfer-TPC generalizes the concept of TPC to incorporate priors from related tasks.
- If we take the Bayesian interpretation of the coding schemes, then TPC makes a very strict assumption that only 1 feature class and 1 feature from that feature class will enter the model.
- But in the case of Transfer-TPC that assumption is relaxed and we allow “similar tasks” to decide how many features and feature classes will enter the model. The hyperparameters of the prior control how much we want our estimates to deviate from those predicted by standard TPC.

Connection between MIC and Transfer-TPC

- These two are perhaps the most different of the pairs of methods.
- MIC does “simultaneous transfer” whereas Transfer-TPC does “sequential transfer” i.e. it does not expect the test task to be known in advance.
- Transfer-TPC can work in cases when tasks have unequal amounts of data; whereas MIC expects all tasks to share the same set of features.
- In MIC we assume that all tasks are related but in Transfer-TPC we explicitly look for tasks that are similar to the target task being learnt.

Connection between MIC and Transfer-TPC

- These two are perhaps the most different of the pairs of methods.
- MIC does “simultaneous transfer” whereas Transfer-TPC does “sequential transfer” i.e. it does not expect the test task to be known in advance.
- Transfer-TPC can work in cases when tasks have unequal amounts of data; whereas MIC expects all tasks to share the same set of features.
- In MIC we assume that all tasks are related but in Transfer-TPC we explicitly look for tasks that are similar to the target task being learnt.
- If we constrain all the tasks to share same set of features then we can reduce Transfer-TPC to MIC by concatenating all the tasks, sharing the same feature matrix.

Summary

- Information theory provides a flexible approach to transferring knowledge for feature selection.
- Coding penalties can easily be customized to fit the problem at hand
- They capture the spirit of Bayesian priors without the strong commitment to specific models.
- We used Information theory for:
  - Multi-label learning (MIC)
  - Multiple feature class learning (TPC)
  - Transfer between different verbs (Transfer-TPC)

Thanks