Multi-task learning

Multitask learning, 
Rich Caruana

Learning multiple tasks with kernel methods, 
Evgeniou, Micchelli and Pontil

Multitask Learning (MTL)

- What is learned for each task can help other tasks be learned better
- Improves generalization
- Trains tasks in parallel
- Use info obtained in the training of similar tasks

For example, object recognition – train the machine to simultaneously recognize shapes, textures, reflections, regions, shadows, text … etc, instead of one at a time.
The inductive bias

Also known as the multitask bias.
MTL incurs a bias in the learning that induces the model to prefer one learning function to another.
The learning hypothesis is biased to prefer a function useful across multiple tasks.
It is this bias that gives MLT its edge in generalizability.
3 MTL applies in backprop nets

- ALVINN – a road image simulator used to generated simulated data for a road following domain
- Object recognition - doorknobs
- Pneumonia prediction

Backprop nets refer to backpropagation neural networks.
Artificial neural networks (ANN) – interconnected artificial neurons (programming units), a simplified model of the brain
Eg. RBF network; two processing layers – inputs and RBF mapping. Typically trained using a maximum likelihood approach (maximize the probability of the data given the model)
Backpropagation algorithm – minimize training error in a NN using steepest descent, and use training errors to weight the neurons at each level

ALVINN generator – 1D and 2D

- Synthetic road images generated based on a number of user-defined parameters: road width, camera angle view, number of lanes … etc.
- 1D computationally easier than 2D
- Goal: predict steering direction
### 1D-ALVINN results

Table 1. Performance of STL and MTL with one hidden layer on tasks in the 1D-ALVINN domain. The bold entries in the STL columns are the STL runs that performed best. Differences statistically significant at 0.05 or better are marked with an *. 

<table>
<thead>
<tr>
<th>TASK</th>
<th>Single 2HU</th>
<th>Single 4HU</th>
<th>Single 8HU</th>
<th>Single 16HU</th>
<th>MTL 16HU</th>
<th>Change MTL to Best STL</th>
<th>Change MTL to Mean STL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 or 2 Lanes</td>
<td>.201</td>
<td>.209</td>
<td>.207</td>
<td>.178</td>
<td>.156</td>
<td>-12.4% *</td>
<td>-21.5% *</td>
</tr>
<tr>
<td>Left Edge</td>
<td>.069</td>
<td>.071</td>
<td>.073</td>
<td>.073</td>
<td>.062</td>
<td>-10.1% *</td>
<td>-13.3% *</td>
</tr>
<tr>
<td>Right Edge</td>
<td>.076</td>
<td>.062</td>
<td>.058</td>
<td>.056</td>
<td>.051</td>
<td>-8.9% *</td>
<td>-19.0% *</td>
</tr>
<tr>
<td>Line Center</td>
<td>.153</td>
<td>.152</td>
<td>.152</td>
<td>.151</td>
<td>-0.7%</td>
<td>-0.8%</td>
<td></td>
</tr>
<tr>
<td>Road Center</td>
<td>.038</td>
<td>.037</td>
<td>.039</td>
<td>.042</td>
<td>.034</td>
<td>-8.1% *</td>
<td>-12.8% *</td>
</tr>
<tr>
<td>Road Greylevel</td>
<td>.054</td>
<td>.055</td>
<td>.055</td>
<td>.054</td>
<td>.038</td>
<td>-20.6% *</td>
<td>-30.3% *</td>
</tr>
<tr>
<td>Edge Greylevel</td>
<td>.037</td>
<td>.038</td>
<td>.039</td>
<td>.038</td>
<td>.038</td>
<td>2.7%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Line Greylevel</td>
<td>.054</td>
<td>.054</td>
<td>.054</td>
<td>.054</td>
<td>.054</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Steering</td>
<td>.093</td>
<td>.069</td>
<td>.087</td>
<td>.072</td>
<td>.058</td>
<td>-15.9% *</td>
<td>-27.7% *</td>
</tr>
</tbody>
</table>

8 tasks plus steering ... Numbers indicate generalization errors. HU refers to hidden units in the NN.

---

### 1D doors

10 tasks ... overall goal - recognize doorknobs in images. Again, MTL outperforms STL.

- horizontal location of doorknob
- horizontal location of doorway center
- horizontal location of left door jamb
- width of left door jamb
- horizontal location of left edge of door
- single or double door
- width of doorway
- horizontal location of right door jamb
- width of right door jamb
- horizontal location of right edge of door
Pneumonia prediction

- Goal: determine how much risk the illness poses to a patient
- 14,199 pneumonia cases, 1542 died
- Data consisted of:
  - Patient survival or time of death
  - Sex, age, pulse, blood count or blood gases
- 35 lab tests available only after patients were hospitalized
  - Use MTL to benefit from future lab results

Rankprop

Rankprop ranks the patients in order of risk without learning to predict mortality. Rankprop is short for backpropogation using sum-of-squares errors on repeatedly re-estimated ranks. MTL outperforms STL by 5-10%
Kernel methods in MTL

- Goal: extend Single-task kernel learning to Multi-task kernel learning using an appropriate kernel method
- MTL applications:
  - Financial predictions using multiple related indicators
  - Bioinformatics: multiple micro-arrays, multiple diseases used to predict tumors

MTL – the linear case

\( \{ (x_i, y_i) : i \in \mathbb{N}_m \} \) \( m \) examples corresponding to the \( i \)th task

**Goal:** Estimate the vector of parameters \( u \) as the minimizer of the regularization function below

\[
R(u) := \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} L(y_{ij}, u_i x_{ij}) + \gamma J(u)
\]

\( J(u) = u^T E u \)

Linear function of \( x \), \( u \) is a vector

\( E \)-matrix that captures relations between the tasks

**Linear MTL kernel:** Using this kernel, can transform a STL problem to a MTL

\[
K((x, \ell), (t, q)) = x^T B \ell^T, \quad x, t \in \mathbb{R}^d, \quad \ell, q \in \mathbb{N}_n
\]

\( B \)-defined from the \( E \) matrix that represents task relationships

More details on \( B \) and its relation with \( E \) in paper (Prop 1)
\[ J(\theta) = u' E u \quad \text{Equation 7} \]

**Proposition 1** If the feature matrix \( B \) is full rank and we define the matrix \( E \) in Equation (7) as to
be \( E = (BF)^{-1} \) then we have that

\[ S(w) = R(w), \quad w \in \mathbb{R}^d. \quad \text{(12)} \]

Conversely, if we choose a symmetric and positive definite matrix \( E \) in Equation (7) and \( T \) is a
squared root of \( E \) then for the choice of \( B = T'E^{-1} \) Equation (12) holds true.

\[
S(w) := \frac{1}{nm} \sum_{i \in \mathcal{N}_m} \sum_{\ell \in \mathcal{N}_n} L(\gamma_{i\ell}u', w' B_{i\ell} x_{i\ell}) + \gamma_0 u' w, \quad w \in \mathbb{R}^d
\]

---

**MTL SVM for binary classification**

**Goal:** learn all \( r \) functions from the available examples

\[
\min \left\{ \sum_{i \in \mathcal{N}_m} \sum_{\ell \in \mathcal{N}_n} \xi_{i\ell} + \frac{1}{2} ||w||^2 \right\}
\]

subject, for all \( i \in \mathbb{N}_m \) and \( \ell \in \mathbb{N}_n \), to the constraints that

\[
\gamma_{i\ell} u' B_{i\ell} x_{i\ell} \geq 1 - \xi_{i\ell} \\
\xi_{i\ell} \geq 0.
\]

\[ u' x = w' B_{i\ell} x, \quad x \in \mathbb{R}^d, \quad \ell \in \mathbb{N}_n \]
Experiments

2 scenarios ...

1. **Identity**: Use kernel defined by (Eqns 25 and 22)

\[ K((x,l),(t,g)) = (1 - \lambda + \lambda n\delta_{ij})x^TQ_{ij}, \quad \epsilon, g \in \mathbb{N}, x, t \in \mathbb{R}^n \]

\[ Q = \text{identity matrix} \]

Choice of \( \lambda \) dependent on number of tasks.
(See results details)
As \( \lambda \to 1 \), we approach STL

2. **PCA**: Use PCA to define \( Q \) using previously learned task parameters 
(online learning of the task kernel using PCA)

Experimental question: how does MTL compare to STL as
- the number of tasks changes?
- the number of data points per task changes?

Customer data experiments

**Goal**: find a function that predicts a customer’s choices based on the data available

200 customers
120 data points each (products)
multiple attributes per data point (eg. Color, size, price)

Model presented as a binary classification problem: will the customer choose object \( x \) as a future purchase, or not?

The 200 customers are considered to be individual tasks
\( \rightarrow \) 200 tasks to be learned
Each task has 120 data points which can be split into training/test datasets
\( \rightarrow \) training set sizes of 20, 30, 60, 90
Customer results

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Data</th>
<th>One SVM</th>
<th>Indiv SVM</th>
<th>Identity</th>
<th>PCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>20</td>
<td>41.97</td>
<td>29.36</td>
<td>28.72</td>
<td>29.16</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>41.41</td>
<td>29.36</td>
<td>28.80</td>
<td>29.26</td>
</tr>
<tr>
<td>200</td>
<td>20</td>
<td>40.08</td>
<td>29.36</td>
<td>27.79</td>
<td>28.33</td>
</tr>
<tr>
<td>50</td>
<td>30</td>
<td>40.73</td>
<td>26.84</td>
<td>25.53</td>
<td>25.65</td>
</tr>
<tr>
<td>100</td>
<td>30</td>
<td>40.66</td>
<td>26.84</td>
<td>25.28</td>
<td>24.79</td>
</tr>
<tr>
<td>200</td>
<td>30</td>
<td>39.43</td>
<td>26.84</td>
<td>25.16</td>
<td>24.13</td>
</tr>
<tr>
<td>50</td>
<td>60</td>
<td>40.33</td>
<td>22.84</td>
<td>22.06</td>
<td>21.88</td>
</tr>
<tr>
<td>100</td>
<td>60</td>
<td>40.02</td>
<td>22.84</td>
<td>22.27</td>
<td>20.79</td>
</tr>
<tr>
<td>200</td>
<td>60</td>
<td>39.74</td>
<td>22.84</td>
<td>21.86</td>
<td>20.00</td>
</tr>
<tr>
<td>50</td>
<td>90</td>
<td>38.51</td>
<td>19.84</td>
<td>19.68</td>
<td>18.45</td>
</tr>
<tr>
<td>100</td>
<td>90</td>
<td>38.97</td>
<td>19.84</td>
<td>19.34</td>
<td>18.08</td>
</tr>
<tr>
<td>200</td>
<td>90</td>
<td>38.77</td>
<td>19.84</td>
<td>19.27</td>
<td>17.53</td>
</tr>
</tbody>
</table>

Table 1: Comparison of Methods as the number of data per task and the number of tasks changes. “One SVM” stands for training one SVM with all the data from all the tasks. “Indiv SVM” stands for training for each task independently. “Identity” stands for the multi-task SVM with the identity matrix, and “PCA” is the multi-task SVM using the PCA approach. Misclassification errors are reported. Best performance(s) at the 5% significance level is in bold.

- When there are few data per task (20, 30, or 60), both multi-task SVMs significantly outperform the single-task SVM.
- As the number of tasks increases the advantage of multi-task learning increases – for example for 20 data per task, the improvement in performance relative to single-task SVM is 1.14, 1.56, and 2.07 percent for the 50, 100, and 200 tasks respectively.
- When we have many data per task (90), the simple multi-task SVM does not provide any advantage relative to the single-task SVM. However, the PCA based multi-task SVM significantly outperforms the other two methods.
- When there are few data per task, the simple multi-task SVM performs better than the PCA multi-task SVM. It may be that in this case the PCA multi-task SVM overfits the data.
Changing number of tasks

Figure 1: The horizontal axis is the number of tasks used. The vertical axis is the total test misclassification error among the tasks. There are 20 training points per task. We also show the performance of a single-task SVM (dashed line) which, of course, is not changing as the number of tasks increases.

Blue line = MTL for \( \lambda = 0.6 \)

---

Figure 2: The horizontal axis is the parameter \( \lambda \) for the simple multi-task method with the identity matrix kernel (22). The vertical axis is the total test misclassification error among the tasks. There are 200 tasks with 20 training points and 100 test points per task. Left is for 10 tasks, and right is for 200 tasks.

For a small number of tasks, best choose a largish \( \lambda \) for optimal MTL performance.
School data experiment

Data provided by Inner London Education Authority and studied in another paper (Bakker and Heskes, 2003).

15362 students, 139 schools

**Goal**: predict student exam performance based on a number of attributes
- (gender, ethnic group, year of the exam, eligible for free school meals, school gender …)

Categorical variables represented by binary vectors
- each student represented by a binary vector of 27 inputs

Each school was considered to be a task
- 139 tasks

Cross validation 10 random splits of the data into 75% training, 25% testing

**Method**: soft-margin SVR in a MTL setting to predict exam scores,
- simple MTL with identity matrix Q

---

![Graph](image.png)

**Figure 3**: Performance on the school data. The horizontal axis is the parameter \( \lambda \) for the simple multi-task method with the identity matrix while the vertical is the explained variance (percentage) on the test data. The solid line is the performance of the proposed approach while the dashed line is the best performance reported in (Bakker and Heskes, 2003).

Y-axis = explained variance, that is, a percentage measure for the mean-squared error … (total variance of the data minus the sum-squared error on the test set as a percentage of the total data variance

---

P. Agius – Spring 2008
Open questions

- Learning a multi-task kernel
- Bounds on generalization error and relationship to $\lambda$
- Computational issues
- MTL feature selection
- Online MTL
- Extending MTL beyond classification and regression