Methods Deep in the Output Space

Jesse Read
Outline

1 Introduction

2 Multi-Output Methods

3 Deep in the Output Space
Classification

We want a model $h$, which can take inputs in $\mathcal{X}$ and provide a suitable output in $\mathcal{Y}$ (under some suitable loss metric).

Binary classification

$$\mathcal{Y} = \{\text{non\_sunset, sunset}\}$$

$$\hat{y} = h(x), \text{ where } \hat{y} \in \mathcal{Y}$$

e.g., $\hat{y} = \text{sunset}$. 
Classification

We want a model $h$, which can take inputs in $\mathcal{X}$ and provide a suitable output in $\mathcal{Y}$ (under some suitable loss metric).

$$x = \text{Multi-class classification}$$

$$\mathcal{Y} = \{\text{sunset, people, foliage, beach, urban}\}$$

$$\hat{y} = h(x), \quad \text{where } \hat{y} \in \mathcal{Y}$$

e.g., $\hat{y} = \text{sunset}$. 

Multi-class classification
Classification

We want a model $h$, which can take inputs in $\mathcal{X}$ and provide a suitable output in $\mathcal{Y}$ (under some suitable loss metric).

$x = \text{Multi-label classification}$

$\mathcal{Y} = \{\text{sunrise, people, foliage, beach, urban}\}$

$\hat{y} = h(x), \text{ where } \hat{y} \subseteq \mathcal{Y}$

e.g., $\hat{y} = \{\text{sunrise, foliage}\} \iff \hat{y} = [1, 0, 1, 0, 0]$ where $\hat{y} \in \{0, 1\}^2$. i.e., multiple labels per instance instead of a single label.
### Single-label vs. Multi-label

#### Single-label Problem \( Y \in \{0, 1\} \)

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( X_4 )</th>
<th>( X_5 )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>3</td>
<td>A</td>
<td>NO</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.9</td>
<td>1</td>
<td>C</td>
<td>YES</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0.0</td>
<td>1</td>
<td>A</td>
<td>NO</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
<td>2</td>
<td>B</td>
<td>YES</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.0</td>
<td>2</td>
<td>B</td>
<td>YES</td>
<td>0</td>
</tr>
</tbody>
</table>

\( 0 \) 0.0 3 A YES ?

#### Multi-label Problem \( Y \subseteq \{\lambda_1, \ldots, \lambda_L\} \)

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( X_4 )</th>
<th>( X_5 )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>3</td>
<td>A</td>
<td>NO</td>
<td>{\lambda_2, \lambda_3}</td>
</tr>
<tr>
<td>0</td>
<td>0.9</td>
<td>1</td>
<td>C</td>
<td>YES</td>
<td>{\lambda_1}</td>
</tr>
<tr>
<td>0</td>
<td>0.0</td>
<td>1</td>
<td>A</td>
<td>NO</td>
<td>{\lambda_2}</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
<td>2</td>
<td>B</td>
<td>YES</td>
<td>{\lambda_1, \lambda_4}</td>
</tr>
<tr>
<td>1</td>
<td>0.0</td>
<td>2</td>
<td>B</td>
<td>YES</td>
<td>{\lambda_4}</td>
</tr>
</tbody>
</table>

\( 0 \) 0.0 3 A YES ?
# Single-label vs. Multi-label

## Single-label Problem

\[ Y \in \{0, 1\} \]

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( X_4 )</th>
<th>( X_5 )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>3</td>
<td>A</td>
<td>NO</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.9</td>
<td>1</td>
<td>C</td>
<td>YES</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0.0</td>
<td>1</td>
<td>A</td>
<td>NO</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
<td>2</td>
<td>B</td>
<td>YES</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.0</td>
<td>2</td>
<td>B</td>
<td>YES</td>
<td>0</td>
</tr>
</tbody>
</table>

| 0       | 0.0     | 3       | A       | YES     | ?       |

## Multi-label Problem

\[ [Y_1, \ldots, Y_L] \in \{0, 1\}^L \]

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( X_4 )</th>
<th>( X_5 )</th>
<th>( Y_1 )</th>
<th>( Y_2 )</th>
<th>( Y_3 )</th>
<th>( Y_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>3</td>
<td>A</td>
<td>NO</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.9</td>
<td>1</td>
<td>C</td>
<td>YES</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0</td>
<td>1</td>
<td>A</td>
<td>NO</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
<td>2</td>
<td>B</td>
<td>YES</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.0</td>
<td>2</td>
<td>B</td>
<td>YES</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

| 0       | 0.0     | 3       | A       | YES     | ?       | ?       | ?       | ?       |


Text Categorization and Tag Recommendation

For example, the IMDb dataset: Textual movie plot summaries associated with genres (labels).

Also: Bookmarks, Bibtex, del.icio.us datasets. E-mail classification, document classification, . . . .
Labelling Images

Images are labelled to associated *Scenes* with e.g.,
\( \subseteq \{\text{beach, sunset, foliage, field, mountain, urban}\} \)
Labelling Audio

For example, labelling music with emotions, concepts, etc.

\[ \in \{ \text{amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely, angry-aggressive} \} \]
Multi-output Learning

We can generalize to multi-class multi-label (multi-output classification):

<table>
<thead>
<tr>
<th>X_1</th>
<th>X_2</th>
<th>X_3</th>
<th>X_4</th>
<th>X_5</th>
<th>type</th>
<th>gender</th>
<th>group</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>1</td>
<td>M</td>
<td>2</td>
</tr>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>4</td>
<td>F</td>
<td>2</td>
</tr>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>2</td>
<td>?</td>
<td>1</td>
</tr>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>3</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Multi-output Learning

We can generalize to multi-class multi-label (multi-output classification):

<table>
<thead>
<tr>
<th>x_1</th>
<th>x_2</th>
<th>x_3</th>
<th>x_4</th>
<th>x_5</th>
<th>type</th>
<th>gender</th>
<th>group</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>1</td>
<td>M</td>
<td>2</td>
</tr>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>4</td>
<td>F</td>
<td>2</td>
</tr>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>2</td>
<td>?</td>
<td>1</td>
</tr>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>3</td>
<td>M</td>
<td>1</td>
</tr>
</tbody>
</table>

Or to continuous outputs (multi-output regression):

<table>
<thead>
<tr>
<th>x_1</th>
<th>x_2</th>
<th>x_3</th>
<th>x_4</th>
<th>x_5</th>
<th>amount</th>
<th>age</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>37.00</td>
<td>25</td>
<td>0.88</td>
</tr>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>-22.88</td>
<td>22</td>
<td>0.22</td>
</tr>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>19.21</td>
<td>12</td>
<td>0.25</td>
</tr>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td>88.23</td>
<td>11</td>
<td>0.77</td>
</tr>
</tbody>
</table>

Or, a mixture of both nominal and continuous values.
Multi-output Learning

We can generalize to multi-class multi-label (multi-output classification):

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
<th>type</th>
<th>gender</th>
<th>group</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>1</td>
<td>M</td>
<td>2</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>4</td>
<td>F</td>
<td>2</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>2</td>
<td>?</td>
<td>1</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>3</td>
<td>M</td>
<td>1</td>
</tr>
</tbody>
</table>

Or to continuous outputs (multi-output regression):

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
<th>amount</th>
<th>age</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>37.00</td>
<td>25</td>
<td>0.88</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>-22.88</td>
<td>22</td>
<td>0.22</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>19.21</td>
<td>12</td>
<td>0.25</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>88.23</td>
<td>11</td>
<td>0.77</td>
</tr>
</tbody>
</table>

Or, a mixture of both nominal and continuous values.
What’s the big deal?

Can’t we just build a separate model for each label separately? (Why should I care about multi-label/multi-output learning?)

– You can build independent models for each output, but with multi-label/multi-output methods, you can achieve:
  - Better predictive performance (up to 20%)
  - Better computational performance (up to orders of magnitude)
  - Discover interesting relationships
  - Find applications in structured-output prediction tasks (e.g., sequence prediction),

But we already have models for this (deep neural nets, CNNs, LSTMs, PGMs, . . . ) . . . – You may be able to make them better! (and they can make multi-label learning better)
What’s the big deal?

Can’t we just build a separate model for each label separately?
(Why should I care about multi-label/multi-output learning?)

– You can build independent models for each output, but with multi-label/multi-output methods, you can achieve
  - Better predictive performance (up to 20%)
  - Better computational performance (up to orders of magnitude)
  - Discover interesting relationships among labels
  - Find applications in structured-output prediction tasks (e.g., sequence prediction),
What’s the big deal?

Can’t we just build a separate model for each label separately? (Why should I care about multi-label/multi-output learning?)

– You can build independent models for each output, but with multi-label/multi-output methods, you can achieve
  - Better predictive performance (up to 20%)
  - Better computational performance (up to orders of magnitude)
  - Discover interesting relationships among labels
  - Find applications in structured-output prediction tasks (e.g., sequence prediction),
    - But we already have models for this (deep neural nets, CNNs, LSTMs, PGMs, . . .) . . .
      – You may be able to make them better!
      (and they can make multi-label learning better)
Structured Output Prediction

In structured output prediction: assume a particular structure among outputs, e.g., time, pixels, coordinates, hierarchy, graphs.

In the basic sense: structured output = multi-label with many labels but we may not be able to assume a particular dependence.
Outline

1. Introduction
2. Multi-Output Methods
3. Deep in the Output Space
Individual Classifiers

\[
\hat{y}_j = h_j(x) = \underset{y_j \in \{0, 1\}}{\text{argmax}} P(y_j | x) \quad \triangleright \text{for index } j = 1, \ldots, L
\]

and then,

\[
\hat{y} = h(x) = [\hat{y}_1, \ldots, \hat{y}_4]
\]

\[
= \left[ \underset{y_1 \in \{0, 1\}}{\text{argmax}} P(y_1 | x), \ldots, \underset{y_4 \in \{0, 1\}}{\text{argmax}} P(y_4 | x) \right]
\]

\[
= \left[ h_1(x), \ldots, h_4(x) \right]
\]

Also known as the binary relevance method (BR) when \( y_j \in \{0, 1\} \).
Why not individual classifiers?

There may be label dependence, i.e.,

\[ P(y|x) \neq \prod_{j=1}^{L} P(y_j|x) \]

- usually an appropriate assumption
- usually loss function is non-decomposable, e.g., 0/1 loss (exact match), Jaccard index, rank loss, . . . .

Table: Average predictive performance (5 fold CV, Exact Match) from Read et al. 2015. Binary relevance vs Monte-carlo classifier chains.

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>BR</th>
<th>MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Music</td>
<td>6</td>
<td>0.30</td>
<td>0.37</td>
</tr>
<tr>
<td>Scene</td>
<td>6</td>
<td>0.54</td>
<td>0.68</td>
</tr>
<tr>
<td>Yeast</td>
<td>14</td>
<td>0.14</td>
<td>0.23</td>
</tr>
<tr>
<td>Genbase</td>
<td>27</td>
<td>0.94</td>
<td>0.96</td>
</tr>
<tr>
<td>Medical</td>
<td>45</td>
<td>0.58</td>
<td>0.62</td>
</tr>
<tr>
<td>Enron</td>
<td>53</td>
<td>0.07</td>
<td>0.09</td>
</tr>
<tr>
<td>Reuters</td>
<td>101</td>
<td>0.29</td>
<td>0.37</td>
</tr>
</tbody>
</table>
Classifier Chains

Classifier Chains\(^1\) for modelling label dependence,

\[
p(y|x) = p(y_1|x) \prod_{j=2}^{L} p(y_j|x, y_1, \ldots, y_{j-1})
\]

\[
\hat{y} = \arg\max_{y \in \{0,1\}^L} p(y|x)
\]

- Training: Build \(L\) binary base classifiers \(h_1, \ldots, h_L\).
- Prediction: Each classifier provides \(\hat{y}_j = h_j(x)\), which can then be used as an additional attribute: \(h_{j+1}(x, \hat{y}_1, \ldots, \hat{y}_j)\)

\(^1\)Read et al. 2009; Dembczyński, Cheng, and Hüllermeier 2010; Read et al. 2011; Read, Martino, and Luengo 2014.
Making Predictions

Instead of exploring all paths $y \in \{0,1\}^L$, can use some tree search (beam search, Monte Carlo samples, A* search, ...), and then:

$$\text{return } \arg\max_{y \subseteq \{y_t\}_{t=1}^T} P(y|x)$$

where $T \ll 2^L$.
Or, simply greedy (a single path: fast, but prone to error propagation).
Improvements:

- Hill climbing the chain order/structure space
- Large ensembles of random structures/label-subspaces
- Try different base learners

... Huge search spaces. But why does it work?
Improvements:

- Hill climbing the chain order/structure space
- Large ensembles of random structures/label-subspaces
- Try different base learners

... Huge search spaces. But why does it work?

- Label dependence?
  - Not the full answer: difficult to map dependence to good models/interpretations if using very approximate inference such as greedy inference;
  - Appears to work even knowledge of label dependence is theoretically unnecessary (e.g., Hamming loss)
Outline

1. Introduction
2. Multi-Output Methods
3. Deep in the Output Space
Probabilistic graphical model:

vs Neural network \((z_j\text{'s just carry forward input, i.e., delay nodes}),\) i.e., using the greedy inference:
Connection to Deep Learning

Classifier chains (left) vs ‘standard’ neural network\(^2\) (right):

Just apply ‘off-the-shelf’ [deep] neural net?
- Dependence is modelled in the latent layer(s)
- Well-established, popular (again), competitive but requires more parametrization, training iterations.
- In classifier chains, the ‘hidden’ nodes come ‘for free’

\(^2\)e.g., MLP; but note: final layer is not a softmax!
Deep in the Label Space

Using other labels as input

- Allows more powerful (non-linear) decision boundaries . . . even with relatively simple classifiers ($\approx$ activation functions)
- Works well with smaller training datasets, less parameterization/iterations.

So using labels as inputs, helps predicting other labels... Where can we get more labels from?
Meta Labels

We can get labels from other labels\(^3\), e.g., \(y_{S_k} \in S_k \subset \mathcal{Y}\); Or, prune to binary:

\[
z_k = 1 \iff y_{S_k} = s^{(k)}
\]

which decodes easily (via voting/weights) back to labels.

\(^3\)Read, Puurula, and Bifet 2014; Read, Martino, and Hollmén 2017.
Synthetic Labels

We can make up our own labels\(^4\) or use the same labels again:

\[
\begin{align*}
   x_1, x_2, x_3, x_4, x_5 \\
   z_1, z_2 \\
   y_1, y_2
\end{align*}
\]

- Synthetic labels \(\approx\) cascaded basis function expansion
- This can be combined with the meta labels
- Can embed these into deep neural networks
- Can include skip layer, hidden layers (latent variables), etc.

---

\(^4\)Read and Hollmén 2014; Read and Hollmén 2017, and related work Spyromitros-Xioufis et al. 2016; Cisse, Al-Shedivat, and Bengio 2016
**Table**: Exact Match, base classifier = logistic regression, except BR$_{RF}$ (random forest)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BR</th>
<th>BR$_{RF}$</th>
<th>CC</th>
<th>...</th>
<th>CC$\ddot{s}$L</th>
<th>...</th>
<th>DNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical</td>
<td>0.52</td>
<td>1.00</td>
<td>2</td>
<td>0.64</td>
<td>8</td>
<td>...</td>
<td>1.00</td>
</tr>
<tr>
<td>Music</td>
<td>0.23</td>
<td>0.25</td>
<td>5</td>
<td>0.25</td>
<td>4</td>
<td>...</td>
<td>0.26</td>
</tr>
<tr>
<td>Scene</td>
<td>0.47</td>
<td>0.48</td>
<td>7</td>
<td>0.55</td>
<td>5</td>
<td>...</td>
<td>0.58</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.14</td>
<td>0.10</td>
<td>9</td>
<td>0.18</td>
<td>3</td>
<td>...</td>
<td>0.18</td>
</tr>
<tr>
<td>Medical</td>
<td>0.45</td>
<td>0.68</td>
<td>4</td>
<td>0.46</td>
<td>6</td>
<td>...</td>
<td>0.68</td>
</tr>
<tr>
<td>Enron</td>
<td>0.11</td>
<td>0.12</td>
<td>6</td>
<td>0.12</td>
<td>5</td>
<td>...</td>
<td>0.13</td>
</tr>
<tr>
<td>Reuters</td>
<td>0.45</td>
<td>0.47</td>
<td>4</td>
<td>0.47</td>
<td>3</td>
<td>...</td>
<td>0.47</td>
</tr>
<tr>
<td>Ohsumed</td>
<td>0.15</td>
<td>0.17</td>
<td>2</td>
<td>0.15</td>
<td>3</td>
<td>...</td>
<td>0.15</td>
</tr>
<tr>
<td>M.Mill</td>
<td>0.09</td>
<td>0.12</td>
<td>2</td>
<td>0.12</td>
<td>3</td>
<td>...</td>
<td>0.11</td>
</tr>
<tr>
<td>Bibtex</td>
<td>0.10</td>
<td>0.10</td>
<td>7</td>
<td>0.11</td>
<td>4</td>
<td>...</td>
<td>0.16</td>
</tr>
<tr>
<td>Corel5k</td>
<td>0.01</td>
<td>0.01</td>
<td>5</td>
<td>0.01</td>
<td>4</td>
<td>...</td>
<td>0.02</td>
</tr>
<tr>
<td><strong>avg rank</strong></td>
<td>6.95</td>
<td>4.82</td>
<td>4.36</td>
<td>...</td>
<td>2.91</td>
<td>...</td>
<td>5.82</td>
</tr>
</tbody>
</table>

We (CCSL) outperform baselines, random-forest baseline, and ‘deep neural network’ (DNN; two hidden layers).
LSHTC4: Large scale text classification

A Kaggle Challenge based on a large dataset created from Wikipedia. The dataset is multi-class, multi-label and hierarchical. The number of categories is roughly 325,000 and number of the documents is 2,400,000, described by about 1,600,000 features.

Winning solution\(^a\) was much faster and higher-performing than employing separate models (ignoring the hierarchy).

\(^a\)Puurula, Read, and Bifet 2014.
Demand Prediction

Outputs represent the demand at multiple points.

Inputs: time, day, etc., earlier demand.
Route/Destination Forecasting

Personal nodes of a traveller and predicted trajectory;
Output: predicted trajectory (time steps \times \text{waypoints})\textsuperscript{\text{a}}.

\textsuperscript{\text{a}}Read, Martino, and Hollmén 2017.
Missing-data imputation (multiple values)

Form multi-output datasets, train, and predict (input) missing values\textsuperscript{a}.

\textsuperscript{a}Montiel et al. 2018.
Reinforcement learning

An agent can carry out multiple actions, model state and reward across multiple timesteps, etc.
Summary

Multi-output methods which are deep in the output space.

- Predicting multiple outputs simultaneously
- Interconnections with other areas (probabilistic graphical models, neural networks, structured-output prediction, transfer learning, . . .)
- Can perform well, and perform robustly with minimal fiddling/expertise/prior knowledge
- Many applications
Methods Deep in the Output Space

Jesse Read