Methods Deep in the Output Space

Jesse Read

Outline

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).

$\mathbf{x} =$ 

Binary classification

$$\mathcal{Y} = \{\texttt{non\_sunset}, \texttt{sunset}\}$$
$$\hat{y} = h(\mathbf{x}), \quad \text{where } \hat{y} \in \mathcal{Y}$$

e.g., $\hat{y} =$ 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).

$\mathbf{x} =$ 

Multi-*class* classification

$$\mathcal{Y} = \{\texttt{sunset}, \texttt{people}, \texttt{foliage}, \texttt{beach}, \texttt{urban}\}$$
$$\hat{y} = h(\mathbf{x}), \quad \text{where } \hat{y} \in \mathcal{Y}$$

e.g., $\hat{y} =$ 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).

$\mathbf{x} =$ 

Multi-label classification

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

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

e.g., $\hat{y} = \{\text{sunset}, \text{foliage}\} \Leftrightarrow \widehat{\mathbf{y}} = [1, 0, 1, 0, 0]$ where $\widehat{\mathbf{y}} \in \{0, 1\}^2$.
i.e., multiple labels per instance instead of a single label.

Single-label vs. Multi-label

Single-label Problem $Y \in \{0, 1\}$

X_1	X_2	X_3	X_4	X_5	Y
1	0.1	3	A	NO	0
0	0.9	1	C	YES	1
0	0.0	1	A	NO	0
1	0.8	2	B	YES	1
1	0.0	2	B	YES	0
0	0.0	3	A	YES	?

Multi-label Problem $Y \subseteq \{\lambda_1, \ldots, \lambda_L\}$

X_1	X_2	X_3	X_4	X_5	Y
1	0.1	3	A	NO	$\{\lambda_2, \lambda_3\}$
0	0.9	1	C	YES	$\{\lambda_1\}$
0	0.0	1	A	NO	$\{\lambda_2\}$
1	0.8	2	B	YES	$\{\lambda_1, \lambda_4\}$
1	0.0	2	B	YES	$\{\lambda_4\}$
0	0.0	3	A	YES	?

Single-label vs. Multi-label

Single-label Problem $Y \in \{0, 1\}$

X_1	X_2	X_3	X_4	X_5	Y
1	0.1	3	A	NO	0
0	0.9	1	C	YES	1
0	0.0	1	A	NO	0
1	0.8	2	B	YES	1
1	0.0	2	B	YES	0
0	0.0	3	A	YES	?

Multi-label Problem $[Y_1, \ldots, Y_L] \in \{0, 1\}^L$

X_1	X_2	X_3	X_4	X_5	Y_1	Y_2	Y_3	Y_4
1	0.1	3	A	NO	0	1	1	0
0	0.9	1	C	YES	1	0	0	0
0	0.0	1	A	NO	0	1	0	0
1	0.8	2	B	YES	1	0	0	1
1	0.0	2	B	YES	0	0	0	1
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.,
$\subseteq \{\texttt{beach}, \texttt{sunset}, \texttt{foliage}, \texttt{field}, \texttt{mountain}, \texttt{urban}\}$

Labelling Audio

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

e.g., $\subseteq \{$ `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):

X_1	X_2	X_3	X_4	X_5	type	gender	group
x_1	x_2	x_3	x_4	x_5	1	M	2
x_1	x_2	x_3	x_4	x_5	4	F	2
x_1	x_2	x_3	x_4	x_5	2	?	1
x_1	x_2	x_3	x_4	x_5	3	M	1
x_1	x_2	x_3	x_4	x_5	?	?	?

Multi-output Learning

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

X_1	X_2	X_3	X_4	X_5	type	gender	group
x_1	x_2	x_3	x_4	x_5	1	M	2
x_1	x_2	x_3	x_4	x_5	4	F	2
x_1	x_2	x_3	x_4	x_5	2	?	1
x_1	x_2	x_3	x_4	x_5	3	M	1
x_1	x_2	x_3	x_4	x_5	?	?	?

Or to continuous outputs (multi-output regression):

X_1	X_2	X_3	X_4	X_5	amount	age	percent
x_1	x_2	x_3	x_4	x_5	37.00	25	0.88
x_1	x_2	x_3	x_4	x_5	-22.88	22	0.22
x_1	x_2	x_3	x_4	x_5	19.21	12	0.25
x_1	x_2	x_3	x_4	x_5	88.23	11	0.77
x_1	x_2	x_3	x_4	x_5	?	?	?

Multi-output Learning

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

X_1	X_2	X_3	X_4	X_5	type	gender	group
x_1	x_2	x_3	x_4	x_5	1	M	2
x_1	x_2	x_3	x_4	x_5	4	F	2
x_1	x_2	x_3	x_4	x_5	2	?	1
x_1	x_2	x_3	x_4	x_5	3	M	1
x_1	x_2	x_3	x_4	x_5	?	?	?

Or to continuous outputs (multi-output regression):

X_1	X_2	X_3	X_4	X_5	amount	age	percent
x_1	x_2	x_3	x_4	x_5	37.00	25	0.88
x_1	x_2	x_3	x_4	x_5	-22.88	22	0.22
x_1	x_2	x_3	x_4	x_5	19.21	12	0.25
x_1	x_2	x_3	x_4	x_5	88.23	11	0.77
x_1	x_2	x_3	x_4	x_5	?	?	?

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?**)

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?

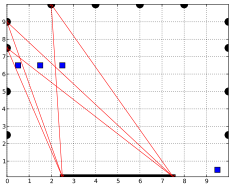
Can't we just build a separate model for each label separately?
(**Why should I care about multi-label/multi-output learning?**)

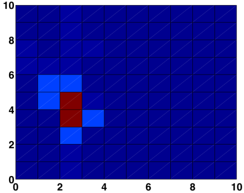
– You *can* build independent models for each output, but with multi-label/multi-output methods, you can achieve

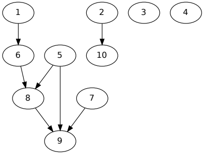
- Better predictive performance (up to 20%)
- Better computational performance (up to orders of magnitude)
- Discover interesting relationships among labels
- Find applications in structured-output prediction tasks (e.g., sequence prediction),
  - **But we already have models for this (deep neural nets,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

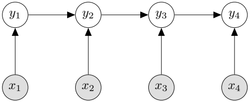
Structured Output Prediction

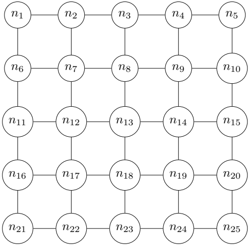
In structured output prediction: assume a particular structure amoung outputs, e.g., time, pixels, coordinates, hierarchy, graphs.







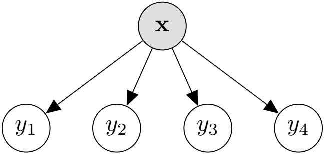




In the basic sense: structured output = multi-label with many labels but we may not be able to assume a particular dependence.

Outline

Individual Classifiers



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

and then,

$$\begin{aligned}
\widehat{\mathbf{y}} = \mathbf{h}(\mathbf{x}) &= [\hat{y}_1, \ldots, \hat{y}_4] \\
&= \left[\underset{y_1 \in \{0,1\}}{\operatorname{argmax}} P(y_1|\mathbf{x}), \cdots, \underset{y_4 \in \{0,1\}}{\operatorname{argmax}} P(y_4|\mathbf{x}) \right] \\
&= \left[h_1(\mathbf{x}), \cdots, h_4(\mathbf{x}) \right]
\end{aligned}$$

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.,

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

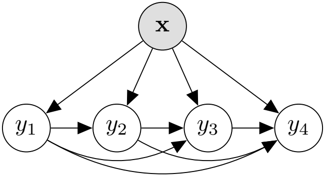
- usually an appropriate assumption
- usually loss function is non-decomposable, e.g., 0/1 loss (exact match), Jaccard index, rank loss, ...

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

	L	BR	MCC
Music	6	0.30	**0.37**
Scene	6	0.54	**0.68**
Yeast	14	0.14	**0.23**
Genbase	27	0.94	**0.96**
Medical	45	0.58	**0.62**
Enron	53	0.07	**0.09**
Reuters	101	0.29	**0.37**

Classifier Chains

Classifier Chains[1] for modelling label dependence,



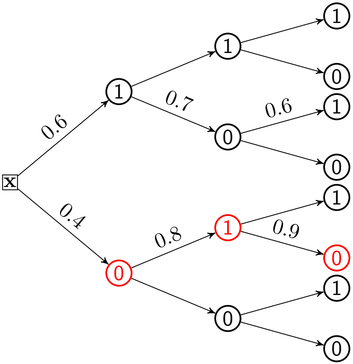
$$p(\mathbf{y}|\mathbf{x}) = p(y_1|\mathbf{x}) \prod_{j=2}^{L} p(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})$$

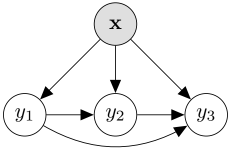
$$\widehat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y} \in \{0,1\}^L} p(\mathbf{y}|\mathbf{x})$$

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

[1] Read et al. 2009; Dembczyński, Cheng, and Hüllermeier 2010; Read et al. 2011; Read, Martino, and Luengo 2014.

Making Predictions



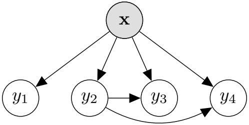


Instead of exploring all paths $\mathbf{y} \in \{0,1\}^L$, can use some tree search (beam search, Monte Carlo samples, A* search, ...), and then.

$$\text{return} \operatorname*{argmax}_{\mathbf{y} \in \{\mathbf{y}_t\}_{t=1}^{T}} P(\mathbf{y}|\mathbf{x})$$

where $T \ll 2^L$.

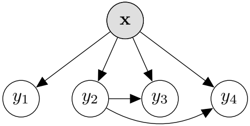
Or, simply greedy (a single path: fast, but prone to error propagation).

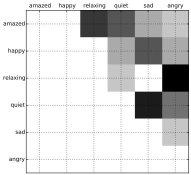


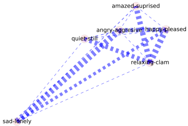
Improvements:

- 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:

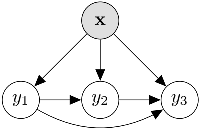
- Hill climbing the chain order/structure space
- Large ensembles of random structures/label-subspaces
- Try different base learners

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

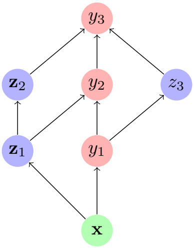
- Label dependence?
  - Not the full answer: difficult to map dependence to good models/interpretations if using very approximate inference such as greedy inference;
  - Appears to work even knowledge of label dependence is theoretically unnecessary (e.g., Hamming loss)

Outline

Probabilistic graphical model:

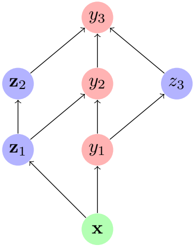


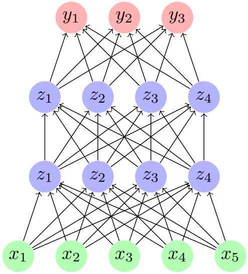
vs Neural network (z_js just carry forward input, i.e., delay nodes), i.e., using the greedy inference:



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.

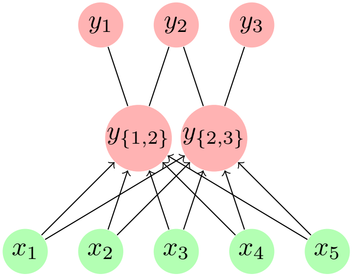
So using labels as inputs, helps predicting other labels... Where can we get more labels from?

Meta Labels

We can get labels from other labels[3], e.g., $\mathbf{y}_{S_k} \in S_k \subset \mathcal{Y}$; Or, prune to binary:

$$z_k = 1 \Leftrightarrow \mathbf{y}_{S_k} = \mathbf{s}^{(k)}$$

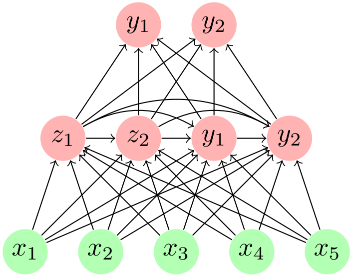
which decodes easily (via voting/weights) back to labels.



[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:



- Synthetic labels ≈ 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

Results

Table: Exact Match, base classifier = logistic regression, except $\mathtt{BR}_{RF}$ (random forest)

Dataset	BR	$\mathtt{BR}_{RF}$	CC	...	CCSL	...	DNN
Logical	0.52 9	1.00 2	0.64 8	...	1.00 2	...	0.83 6
Music	0.23 8	0.25 5	0.25 4	...	0.26 1	...	0.25 3
Scene	0.47 8	0.48 7	0.55 5	...	0.58 1	...	0.56 2
Yeast	0.14 6	0.10 9	0.18 3	...	0.18 5	...	0.12 7
Medical	0.45 7	0.68 4	0.46 6	...	0.68 2	...	0.62 5
Enron	0.11 7	0.12 6	0.12 5	...	0.13 2	...	0.09 8
Reuters	0.45 7	0.47 4	0.47 3	...	0.47 2	...	0.38 8
Ohsumed	0.15 4	0.17 2	0.15 3	...	0.15 6	...	0.21 1
M.Mill	0.09 8	0.12 2	0.12 3	...	0.11 6	...	0.05 9
Bibtex	0.10 5	0.10 7	0.11 4	...	0.16 3	...	0.07 8
Corel5k	0.01 7	0.01 5	0.01 4	...	0.02 1	...	0.01 7
avg rank	6.95	4.82	4.36	...	2.91	...	5.82

We (CCSL) outperform baselines, random-forest baseline, and ‘deep neural network’ (DNN; two hidden layers).

More Applications

LSHTC4: Large scale text classification

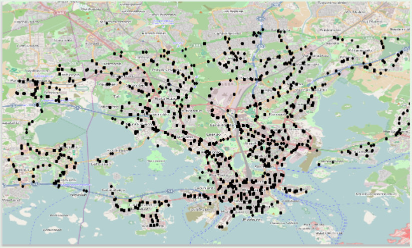
A Kaggle Challenge based on a large dataset created from Wikipedia. The dataset is multi-class, multi-label and hierarchical. The number of categories is roughly 325,000 and number of the documents is 2,400,000, described by about 1,600,000 features.

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

[a] Puurula, Read, and Bifet 2014.

Demand Prediction

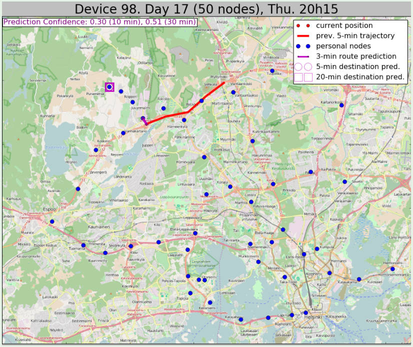
Outputs represent the demand at multiple points.



Inputs: time, day, etc., earlier demand.

Route/Destination Forecasting

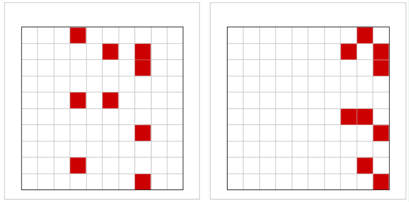
Personal nodes of a traveller and predicted trajectory;
Output: predicted trajectory (time steps $\times$ waypoints)[a].



[a] Read, Martino, and Hollmén 2017.

Missing-data imputation (multiple values)

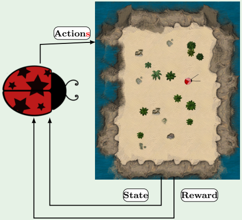
Form multi-output datasets, train, and predict (input) missing values[a].



[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