MTTTS16 Learning from Multiple Sources

5 ECTS credits

Autumn 2019, Tampere University

Lecturer: Jaakko Peltonen

Lecture 2: Basic multi-task learning

On this lecture:

- Example of canonical correlation analysis
- Basic multitask learning

Part 1: Example of CCA

Canonical Correlation Analysis, recap

Reminder: CCA finds projections of two simultaneously observed data sources (two feature sets for the same samples) so that the projections are maximally correlated.

Used in many tasks and data domains.

Canonical Correlation Analysis, recap

- For **x**, find a projection $w_{x,1}x_1 + w_{x,2}x_2 + \ldots + w_{x,K}x_K$ where $\mathbf{w_x} = [\, w_{x,1}, w_{x,2}, \ldots, w_{x,K}\,]$ is the projection basis.
- For **y**, find a projection $w_{y,1}y_1 + w_{y,2}y_2 + \ldots + w_{y,L}y_L$ where $\mathbf{w_y} = [\, w_{y,1}, w_{y,2}, \ldots, w_{y,L}\,]$ is the projection basis.
- Find the projection bases by maximizing the correlation between the projections: maximize

$$corr(\mathbf{w}_x^T\mathbf{x}, \mathbf{w}_y^T\mathbf{y}) = \frac{E[\mathbf{w}_x^T\mathbf{x}\,\mathbf{w}_y^T\mathbf{y}]}{\left(E[(\mathbf{w}_x^T\mathbf{x})^2]\,E[(\mathbf{w}_y^T\mathbf{y})^2]\right)^{1/2}}$$

with respect to $\mathbf{w_x}$ and $\mathbf{w_y}$.

This definition assumes x and y are zero-mean, otherwise substract the means as in the original correlation definition.

- For a finite data set: maximize the sample correlation

$$\hat{corr}(\mathbf{w}_x^T\mathbf{x}, \mathbf{w}_y^T\mathbf{y}) = \frac{\hat{E}_{ML}[\mathbf{w}_x^T\mathbf{x}\,\mathbf{w}_y^T\mathbf{y}]}{\left(\hat{E}_{ML}[(\mathbf{w}_x^T\mathbf{x})^2]\,\hat{E}_{ML}[(\mathbf{w}_y^T\mathbf{y})^2]\right)^{1/2}}$$

Same definition as before

$$\hat{E}_{ML}[x\,y] = \frac{1}{N}\sum_{i=1}^{N} x^i y^i$$

Canonical Correlation Analysis, recap

- CCA can be solved as a generalized eigenvalue equation

$$\hat{C}_{x,y}\hat{C}_{y}^{-1}\hat{C}_{y,x}\mathbf{w}_x=\lambda^2\hat{C}_x\mathbf{w}_x$$

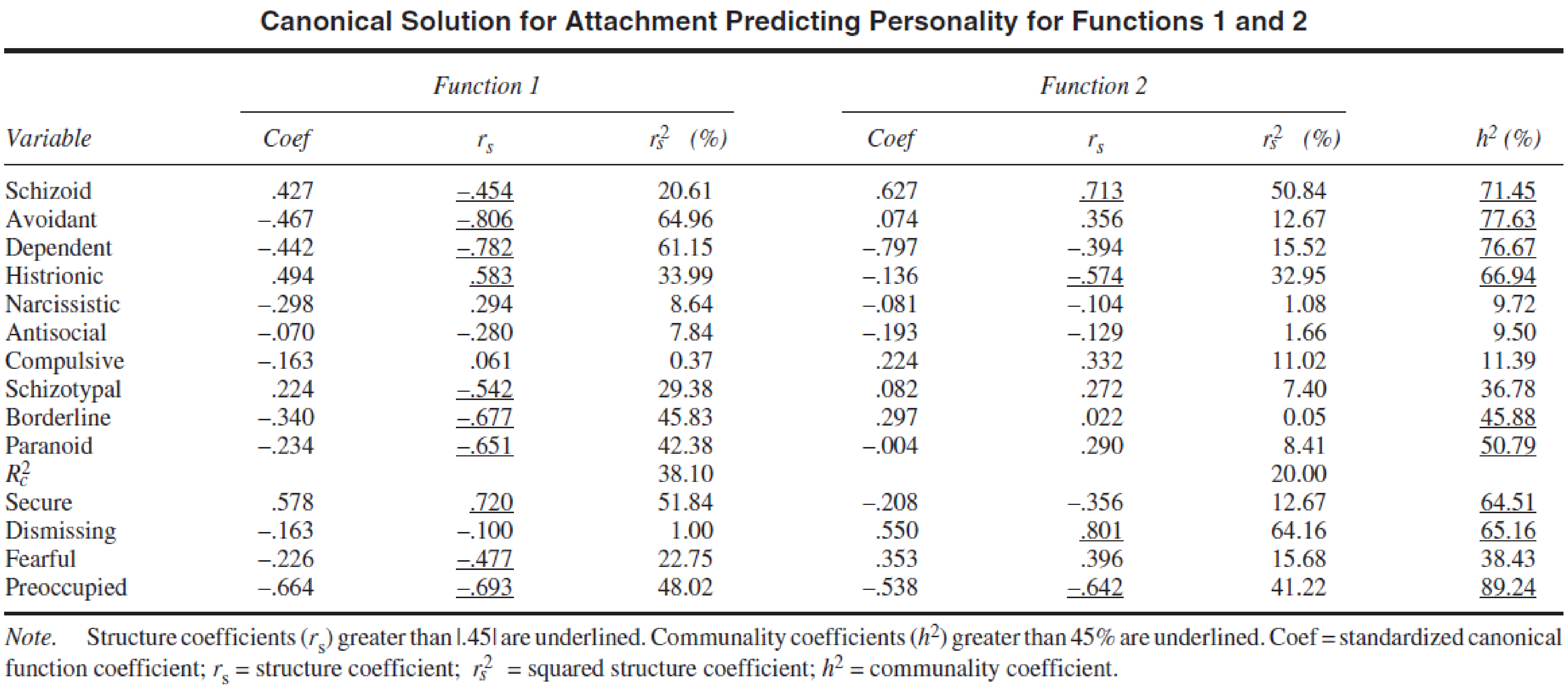
$$\mathbf{w}_y=(1/\lambda)\hat{C}_{y}^{-1}\hat{C}_{y,x}\mathbf{w}_x$$

- This is a generalized eigenvalue equation which we can solve to get $\mathbf{w}_\mathbf{x}$, and the previous equation then gives $\mathbf{w}_\mathbf{y}$ from $\mathbf{w}_\mathbf{x}$.

Canonical Correlation Analysis, example 1

- example: psychology study
- Reference: Alissa Sherry and Robin K. Henson. Conducting and Interpreting Canonical Correlation Analysis in Personality Research: A User-Friendly Primer. Journal of Personality Assessment 84(1), 37–48, 2005.
- 269 undergraduate students from three US universities
- Find correlated projections between a set of 4 **predictor variables** (relationship questionnaires assessing *secure*, *dismissing*, *fearful*, and *preoccupied* attachment) and a set of **criterion variables** (personality variables on several scales: *Schizoid*, *Avoidant*, *Dependent*, *Histrionic*, *Narcissistic*, *Antisocial*, *Compulsive*, *Schizotypal*, *Borderline*, and *Paranoid*)

Canonical Correlation Analysis, example 1



Canonical Solution for Attachment Predicting Personality for Functions 1 and 2

	Function 1			*Function 2*			
Variable	*Coef*	r_s	r_s^2 (%)	*Coef*	r_s	r_s^2 (%)	h^2 (%)
Schizoid	.427	–.454	20.61	.627	.713	50.84	71.45
Avoidant	–.467	–.806	64.96	.074	.356	12.67	77.63
Dependent	–.442	–.782	61.15	–.797	–.394	15.52	76.67
Histrionic	.494	.583	33.99	–.136	–.574	32.95	66.94
Narcissistic	–.298	.294	8.64	–.081	–.104	1.08	9.72
Antisocial	–.070	–.280	7.84	–.193	–.129	1.66	9.50
Compulsive	–.163	.061	0.37	.224	.332	11.02	11.39
Schizotypal	.224	–.542	29.38	.082	.272	7.40	36.78
Borderline	–.340	–.677	45.83	.297	.022	0.05	45.88
Paranoid	–.234	–.651	42.38	–.004	.290	8.41	50.79
R_c^2			38.10			20.00	
Secure	.578	.720	51.84	–.208	–.356	12.67	64.51
Dismissing	–.163	–.100	1.00	.550	.801	64.16	65.16
Fearful	–.226	–.477	22.75	.353	.396	15.68	38.43
Preoccupied	–.664	–.693	48.02	–.538	–.642	41.22	89.24

Note. Structure coefficients (r_s) greater than |.45| are underlined. Communality coefficients (h^2) greater than 45% are underlined. Coef = standardized canonical function coefficient; r_s = structure coefficient; r_s^2 = squared structure coefficient; h^2 = communality coefficient.

Canonical Correlation Analysis, example 2

- example 2: bioinformatics study
- Reference: Abhishek Tripathi, Arto Klami and Samuel Kaski. Simple integrative preprocessing preserves what is shared in data sources. *BMC Bioinformatics* **9**:111, 2008.
- Study shared effects in leukemia subtypes
- 22283 genes for 31 patients, divided into five subtypes of Pediatric acute lymphoblastic leukemia (ALL)
- There is an extension of CCA which works with multiple (more than two) data sources; it is used here, see the paper for details
- CCA is here used for preprocessing: the projected value(s) from each source can be treated as new features
- 11 features extracted by CCA
- The 1% of genes with the highest norm (distance from the origin along CCA axes) are selected: high distance from origin implies high total contribution to the shared variation.
- The selected genes are checked for gene ontology annotations

Canonical Correlation Analysis, example 2

- results

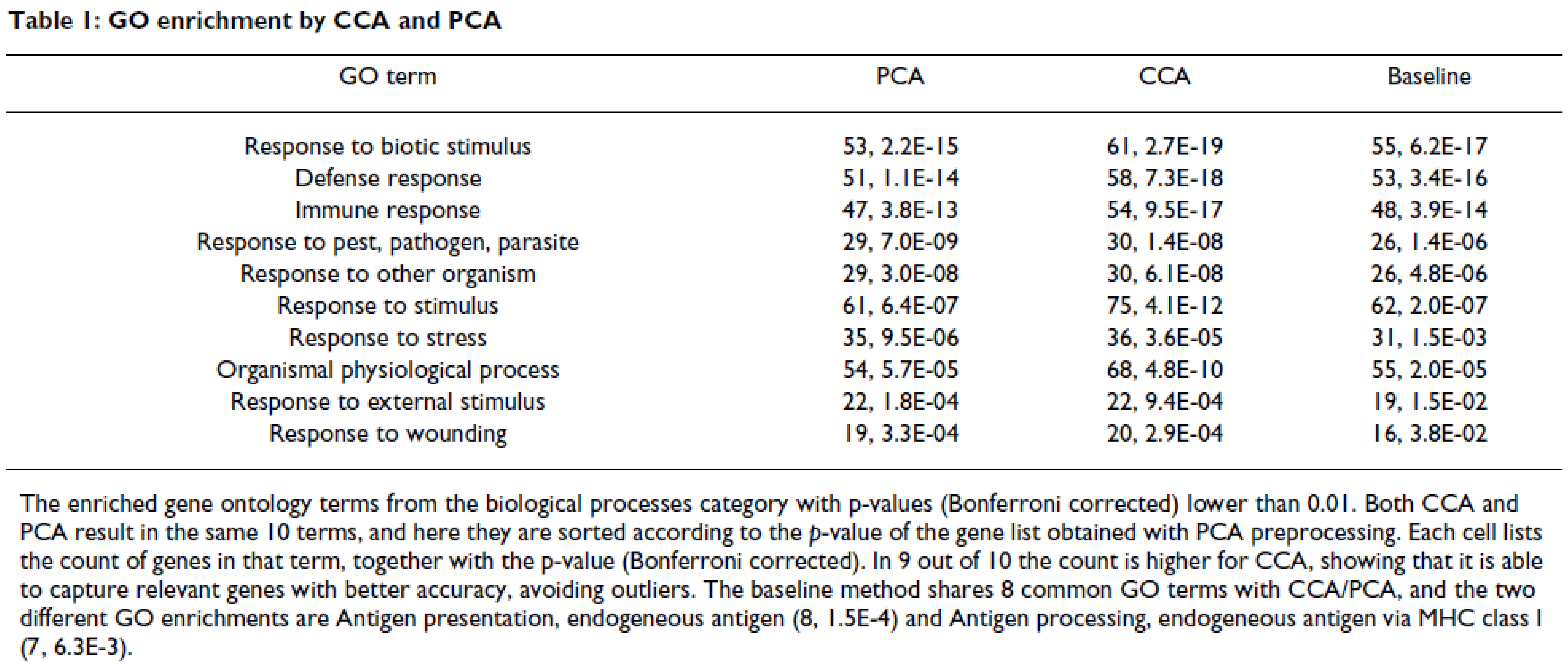


Table 1: GO enrichment by CCA and PCA

GO term	PCA	CCA	Baseline
Response to biotic stimulus	53, 2.2E-15	61, 2.7E-19	55, 6.2E-17
Defense response	51, 1.1E-14	58, 7.3E-18	53, 3.4E-16
Immune response	47, 3.8E-13	54, 9.5E-17	48, 3.9E-14
Response to pest, pathogen, parasite	29, 7.0E-09	30, 1.4E-08	26, 1.4E-06
Response to other organism	29, 3.0E-08	30, 6.1E-08	26, 4.8E-06
Response to stimulus	61, 6.4E-07	75, 4.1E-12	62, 2.0E-07
Response to stress	35, 9.5E-06	36, 3.6E-05	31, 1.5E-03
Organismal physiological process	54, 5.7E-05	68, 4.8E-10	55, 2.0E-05
Response to external stimulus	22, 1.8E-04	22, 9.4E-04	19, 1.5E-02
Response to wounding	19, 3.3E-04	20, 2.9E-04	16, 3.8E-02

The enriched gene ontology terms from the biological processes category with p-values (Bonferroni corrected) lower than 0.01. Both CCA and PCA result in the same 10 terms, and here they are sorted according to the *p*-value of the gene list obtained with PCA preprocessing. Each cell lists the count of genes in that term, together with the p-value (Bonferroni corrected). In 9 out of 10 the count is higher for CCA, showing that it is able to capture relevant genes with better accuracy, avoiding outliers. The baseline method shares 8 common GO terms with CCA/PCA, and the two different GO enrichments are Antigen presentation, endogeneous antigen (8, 1.5E-4) and Antigen processing, endogeneous antigen via MHC class I (7, 6.3E-3).

Part 2: Basic multitask learning

Basic multitask learning: introduction

- The typical approach in statistics and machine learning is to learn each task at a time.
- In some cases if there are complicated tasks they may be broken into subtasks that are learned separately.
- Suppose you have a data set of input samples: for example "standard measurements" taken from each hospital patient
- You want to predict several things from them: for example,
  - do they have a cold
  - do they have cancer
  - do they have malaria

Basic multitask learning: introduction

- It would be possible to run separate tests for each (to get specialized features indicative of each disease)
  - however this is expensive
  - and you might miss potential predictive power in the standard features
- The standard feature set might be complicated and high-dimensional (if they e.g. include measurements of gene activities)
- Given a small number of input samples, it may be hard to learn a good predictor for each disease from the input features without overlearning

Basic multitask learning: introduction

- Multitask learning is an approach to **inductive transfer** (transferring information from one learning task to another).
- It tries to avoid overlearning and therefore improve generalization.
- It uses **domain information**, contained in the training signals of several related tasks, as an **inductive bias**.
- Explicit a priori information on how the tasks are related is not needed, it is learned from the data.
- Multitask learning learns several related task **in parallel**, using a **shared representation** of what features are important for the tasks.
- The assumption is that the shared representation is rich enough to describe each task, and can be learned from the data

Basic multitask learning: introduction

- Inductive transfer is one way of causing **inductive bias**: making the learner prefer some hypotheses over others.
- Bias-free learning is impossible, must make some assumptions to learn anything (see no-free-lunch theorems)
- In Bayesian statistics the bias is caused by the selection of the model family, and selection of a prior inside a model family (more on this in later lectures)
- On this lecture the bias comes partly from the other tasks being learned (from the data of the other tasks)

Basic multitask learning: introduction

- On this lecture the bias comes partly from the other tasks being learned (from the data of the other tasks)
- Inductive bias in multitask learning: we prefer hypotheses that explain multiple tasks
- Advantages of inductive transfer:
  - better statistical learning (we focus on this)
  - potentially better computational speed
  - potentially better intelligibility of the learned models

Basic multitask learning: linear regression example

- Consider multivariate linear regression:
    - tries to find a predictive relationship between input variables and a target variable, of the form $y = a_1 x_1 + \ldots + a_K x_K + b$
    - differences between predicted and observed y are treated as noise, parameters fitted to data (optimized) to minimize remaining noise
- What if you have several target variables?
- Standard answer: learn separate predictors every time.

$$y_1 = a_{1,1} x_1 + \ldots + a_{1,K} x_K + b_1$$
$$y_2 = a_{2,1} x_1 + \ldots + a_{2,K} x_K + b_2$$

- In vector form: $\mathbf{y} = \mathbf{Ax} + \mathbf{b}$, $\mathbf{y} = [y_1 , y_2]$, $\mathbf{x} = [x_1 , \ldots , x_K]$
- The learning of such separate predictors shares no information

Basic multitask learning: linear regression example

- What if there is a shared representation?
- Suppose the predictive tasks share an intermediate representation, for example of the form $\mathbf{z} = [z_1 , \ldots , z_L]$

$$z_1 = c_{1,1}x_1 + \ldots + c_{1,K}x_K + d_1$$
$$\ldots \qquad\qquad \ldots$$
$$z_L = c_{L,1}x_1 + \ldots + c_{L,K}x_K + d_L$$

- The predictive tasks then use the shared representation:

$$y_1 = a_{1,1}z_1 + \ldots + a_{1,L}z_L + b_1$$
$$y_2 = a_{2,1}z_1 + \ldots + a_{2,L}z_L + b_2$$

- In vector form: $\mathbf{z} = \mathbf{Cx} + \mathbf{d}$, $\quad \mathbf{y} = \mathbf{Az} + \mathbf{b} = \mathbf{A}(\mathbf{Cx} + \mathbf{d}) + \mathbf{b}$
- The learning of **C** and **d** is shared

Basic multitask learning: more general example

- More generally: assume there is some function that learns the shared representation, and some functions that learn the individual tasks from the shared representation
- Shared representation: $\mathbf{z} = \mathbf{f}(\mathbf{x}; \mathbf{c})$ where the function has some known basic functional form (e.g., a linear equation) and **c** are the parameters of the form
- Learning the individual tasks: separately for each tasks from the shared representation

  $y_1 = g_1(\mathbf{z}; \mathbf{a}_1)$ where g_1 is some known functional form for y_1 and $\mathbf{a}_1$ are its parameters

  $y_2 = g_2(\mathbf{z}; \mathbf{a}_2)$ where g_2 is some known functional form for y_2 and $\mathbf{a}_2$ are its parameters

Basic multitask learning: more general example

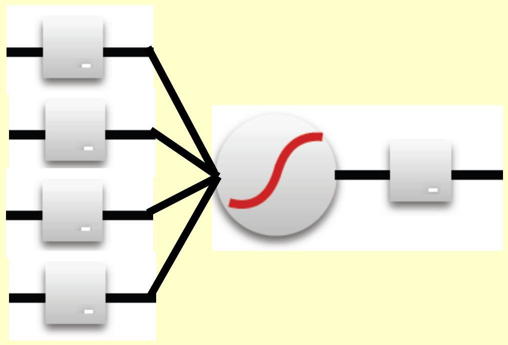
- Learning of **f** is shared, learning of the g functions is separate
- “Shared” and “separate” refers to the way statistical information is used; algorithmically everything happens at the same time, and during the algorithm the learning of **f** affects the learning of the g functions.
- Note: sometimes people use a simple preprocessing for data, (e.g. principal component analysis to reduce data dimensionality) and then learn several separate tasks from the processed data. This kind of simple preprocessing creates a shared representation for learning the tasks, but the representation is not learned for the tasks, it is just some off-the-shelf preprocessing.

Basic multitask learning: in neural networks

- One of the prominent early papers about multitask learning was in the context of learning **artificial neural networks (ANNs)**.
- ANNs are inspired by the signal processing in human neurons
- They learn a complicated nonlinear input-output function composed of many simple parts (neurons)

Basic multitask learning: in neural networks

- One of the prominent early papers about multitask learning was in the context of learning **artificial neural networks (ANNs)**.
- ANNs are inspired by the signal processing in human neurons
- They learn a complicated nonlinear input-output function composed of many simple parts (neurons)



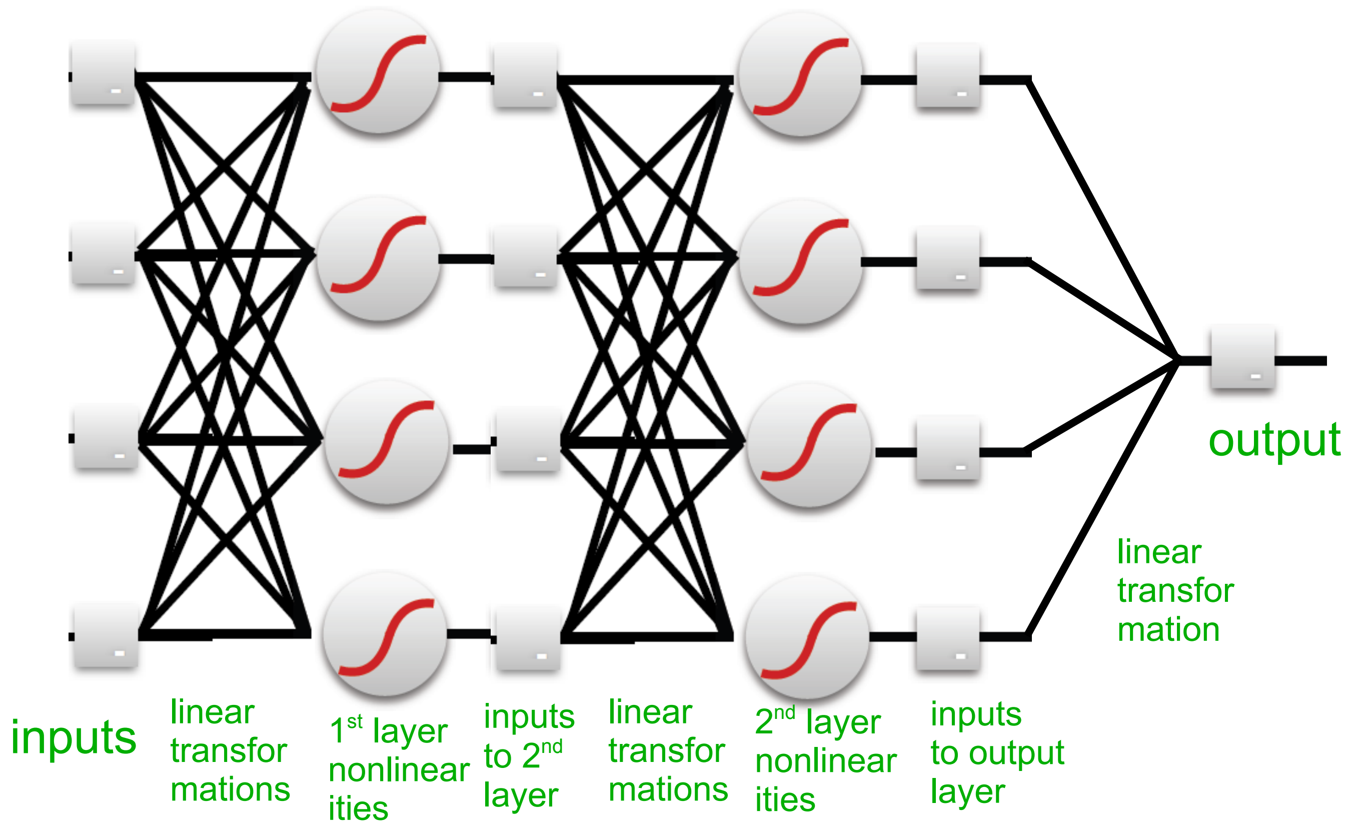
Individual neuron computes:

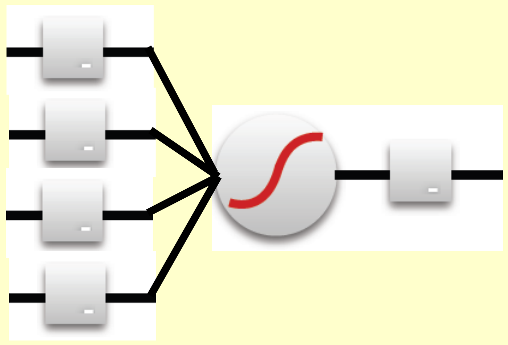
$$s=\sum_{i=1}^{d} w_i x_i + b$$

$$y=\phi(s)=\frac{1}{1+\exp(-s)}$$

Basic multitask learning: in neural networks

- One of the prominent early papers about multitask learning was in the context of learning **artificial neural networks (ANNs)**.
- ANNs are inspired by the signal processing in human neurons
- They learn a complicated nonlinear input-output function composed of many simple parts (neurons)





Individual neuron computes:

$$s = \sum_{i=1}^{d} w_i x_i + b$$

$$y = \phi(s) = \frac{1}{1+\exp(-s)}$$

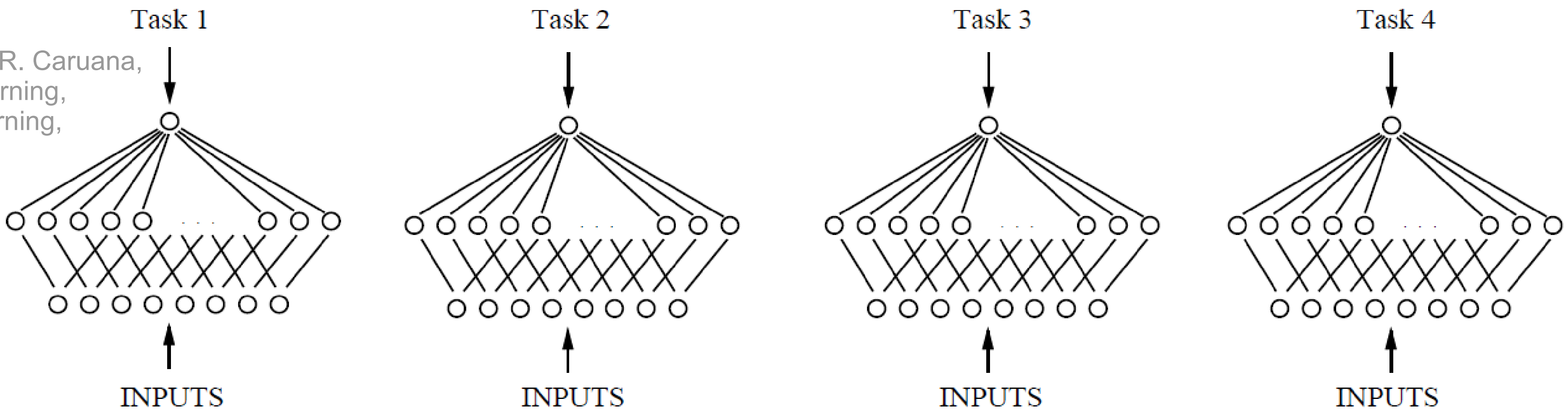
Basic multitask learning: in neural networks

- Learning the parameters of an ANN can be seen as fitting a statistical model to data, just like e.g. linear regression; only the input-output function is more complicated
- ANNs are typically fitted by “back-propagation” (a gradient descent algorithm) to minimize squared error. Same as maximum likelihood fitting of a statistical input-output model with normally distributed noise.
- An ANN trained on a single difficult task may not learn it well from limited data (e.g. recognize whether a particular type of object is present in a picture: input-output function from a 1000x1000 pixel vector to a binary variable)
- Could we train the ANN to recognize also simpler things (edges, corners, textures, ...)?

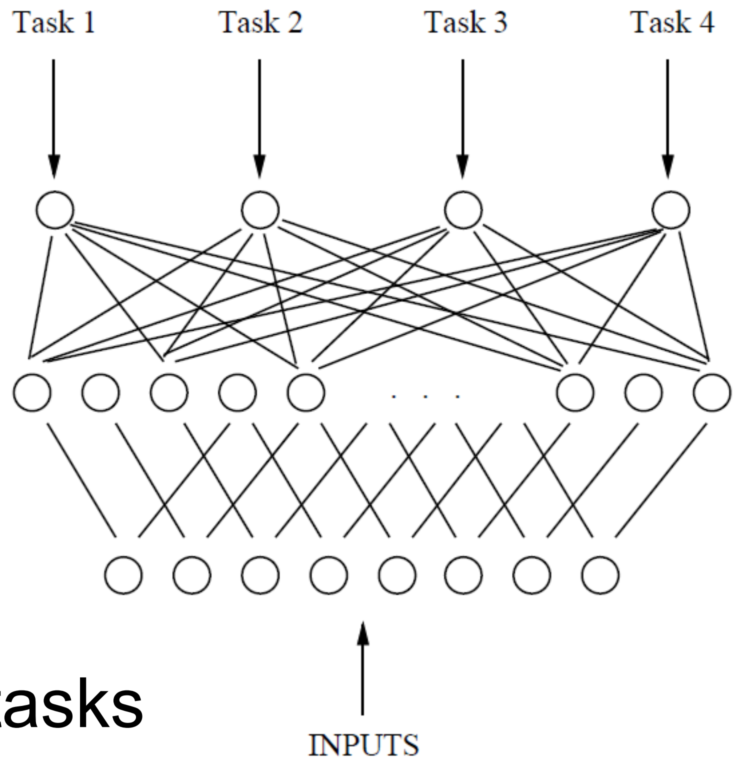
Basic multitask learning: in neural networks

- Single-task learning (STL): learn each task with a different ANN

Images from R. Caruana, Multitask Learning, Machine Learning, 1997



- Multi-task learning (MTL): use a shared intermediate representation: share the intermediate layers!
- Becomes a single ANN with multiple outputs in the output layer (one per task)
- Training is done in parallel for all tasks



Basic multitask learning: in neural networks

- Advantages of the shared training:
    - features developed for one task can be used for others
    - features may be developed that might not have been found when training any single task alone
- Features are not forced to be shared: the output layer for one task can choose to ignore some of the intermediate features.
    - this is done by setting the weight for those features near-zero for that output
- As a result, some intermediate features can become specialized
  for use in particular tasks

  In principle, it is even possible to represent completely separate subnetworks within a large enough shared network, and use one subnetwork per output--->same as STL, if that turns out to be the best representation. In practice this is unlikely to happen.
- Inductive bias due to restricted resources: limited amount of intermediate neurons ---> prefer solutions where neurons are

Basic multitask learning: in neural networks, example 1

- Example task - prediction for robot navigation planning. Models image properties.
- Main task: given robot's camera image, predict whether the image contains a doorknob. Learn several other tasks simultaneously.

Example images:

Tasks to learn:

- 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

Images from R. Caruana, Multitask Learning, Machine Learning, 1997

Basic multitask learning: in neural networks, example 1

- Train with a training set annotated with the outputs for all tasks to learn.
- Use backpropagation = gradient descent, to minimize squared error between predictions and outputs)
- Test on a separate test set not used in training.
- Result: multi-task learning outperforms single-task learning on the test set

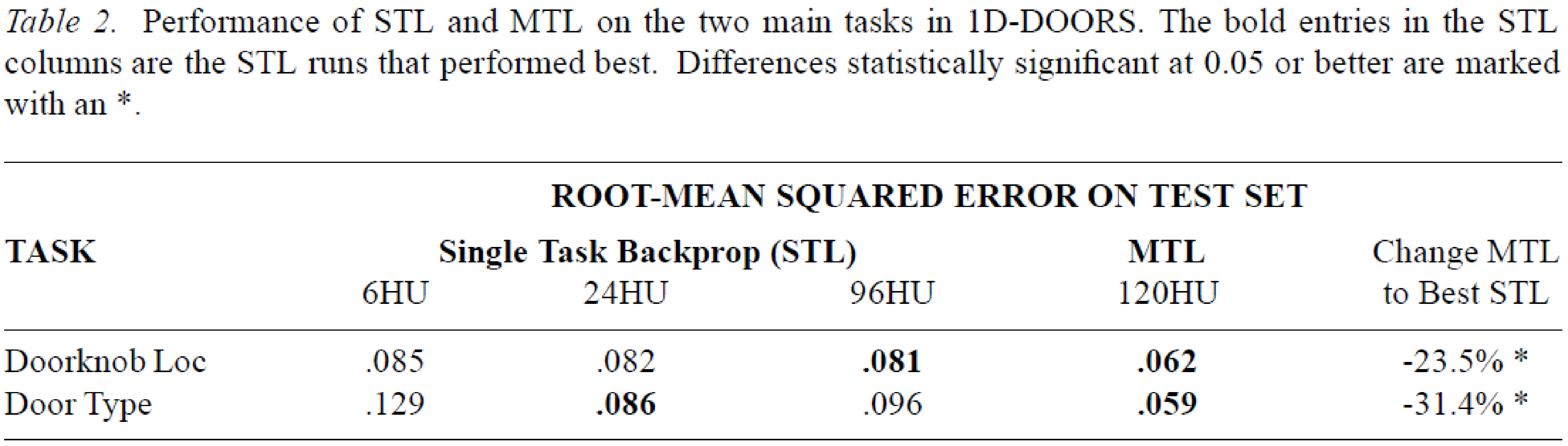


Table 2. Performance of STL and MTL on the two main tasks in 1D-DOORS. The bold entries in the STL columns are the STL runs that performed best. Differences statistically significant at 0.05 or better are marked with an *.

	ROOT-MEAN SQUARED ERROR ON TEST SET				
TASK	**Single Task Backprop (STL)**			**MTL**	Change MTL
	6HU	24HU	96HU	120HU	to Best STL
Doorknob Loc	.085	.082	**.081**	**.062**	-23.5% *
Door Type	.129	**.086**	.096	**.059**	-31.4% *

Images from R. Caruana, Multitask Learning, Machine Learning, 1997

Basic multitask learning: in neural networks, example 2

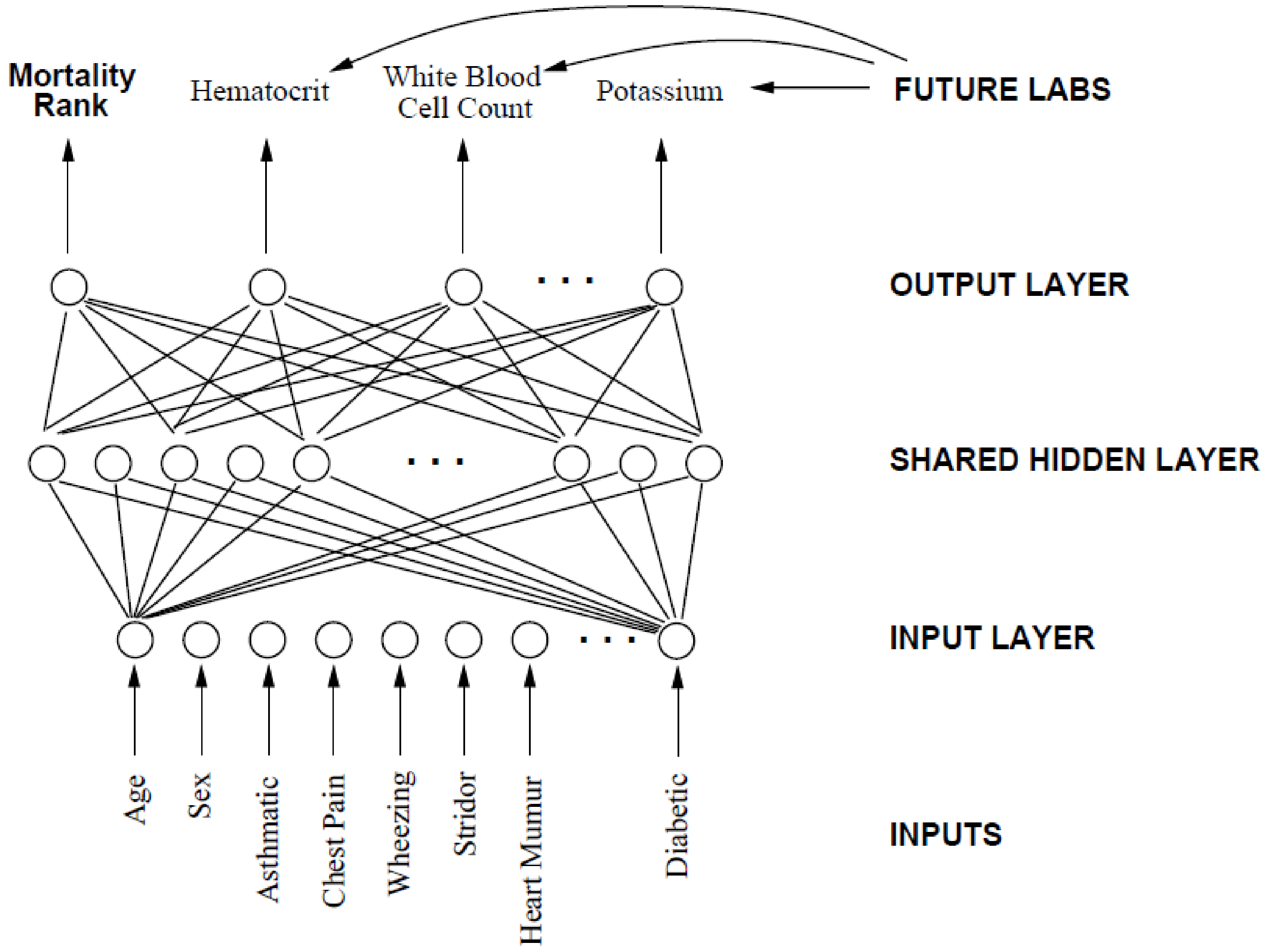
- Medis Pneumonia Database has 14199 pneumonia cases from 78 hospitals in 1989
- 65 measurements available for most patients: 30 basic measurements measured prior to hospitalization (age, sex, pulse, etc.), and 35 lab results (blood counts, blood gases etc.) usually available only after hospitalization.
- Training data includes knowledge which patients died.
- Main task: rank neumonia patients by probability of death.
- More precisely, find the 10%, 20%, 30%, 40% and 50% of the population at least risk, minimize error rate at each fraction of the population (FOP).
- Multitask setup:use the 35 lab results as extra targets, to be predicted from the 30 basic measurements

Images from R. Caruana, Multitask Learning, Machine Learning, 1997

Basic multitask learning: in neural networks, example 2

- Multitask setup:use the 35 lab results as extra targets, to be predicted from the 30 basic measurements

Mortality rank = patient is in 10%/20%/30%/40%/50% highest riskpatients or not



Lab results may not be available when risk is predicted, that's why they are used as extra targets instead of inputs

Images from R. Caruana, Multitask Learning, Machine Learning, 1997

Basic multitask learning: in neural networks, example 2

- Multitask results are better than single-task results for some of the population fractions

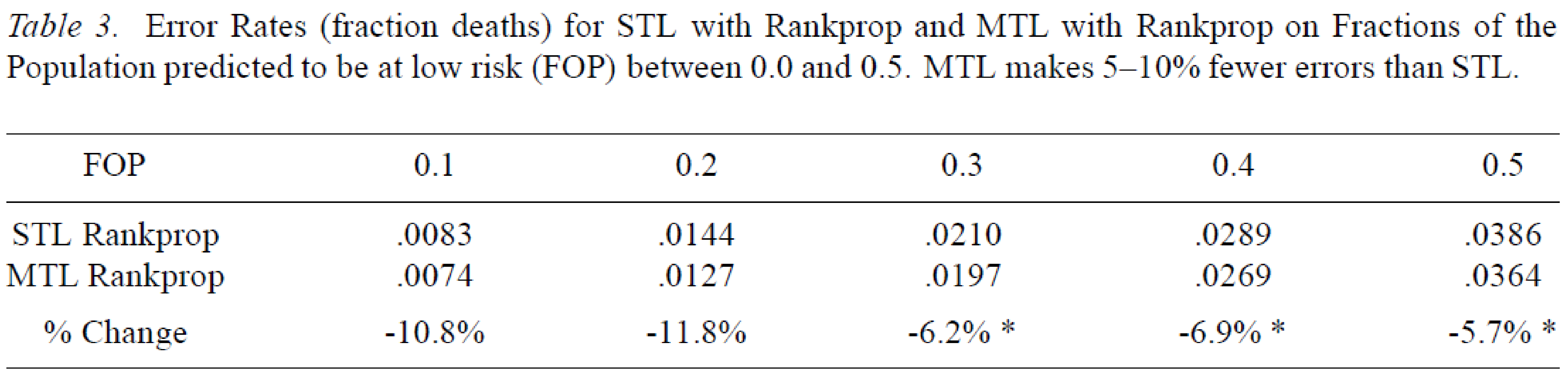


Table 3. Error Rates (fraction deaths) for STL with Rankprop and MTL with Rankprop on Fractions of the Population predicted to be at low risk (FOP) between 0.0 and 0.5. MTL makes 5–10% fewer errors than STL.

FOP	0.1	0.2	0.3	0.4	0.5
STL Rankprop	.0083	.0144	.0210	.0289	.0386
MTL Rankprop	.0074	.0127	.0197	.0269	.0364
% Change	-10.8%	-11.8%	-6.2% *	-6.9% *	-5.7% *

Images from R. Caruana, Multitask Learning, Machine Learning, 1997

Basic multitask learning with neural networks, mechanisms

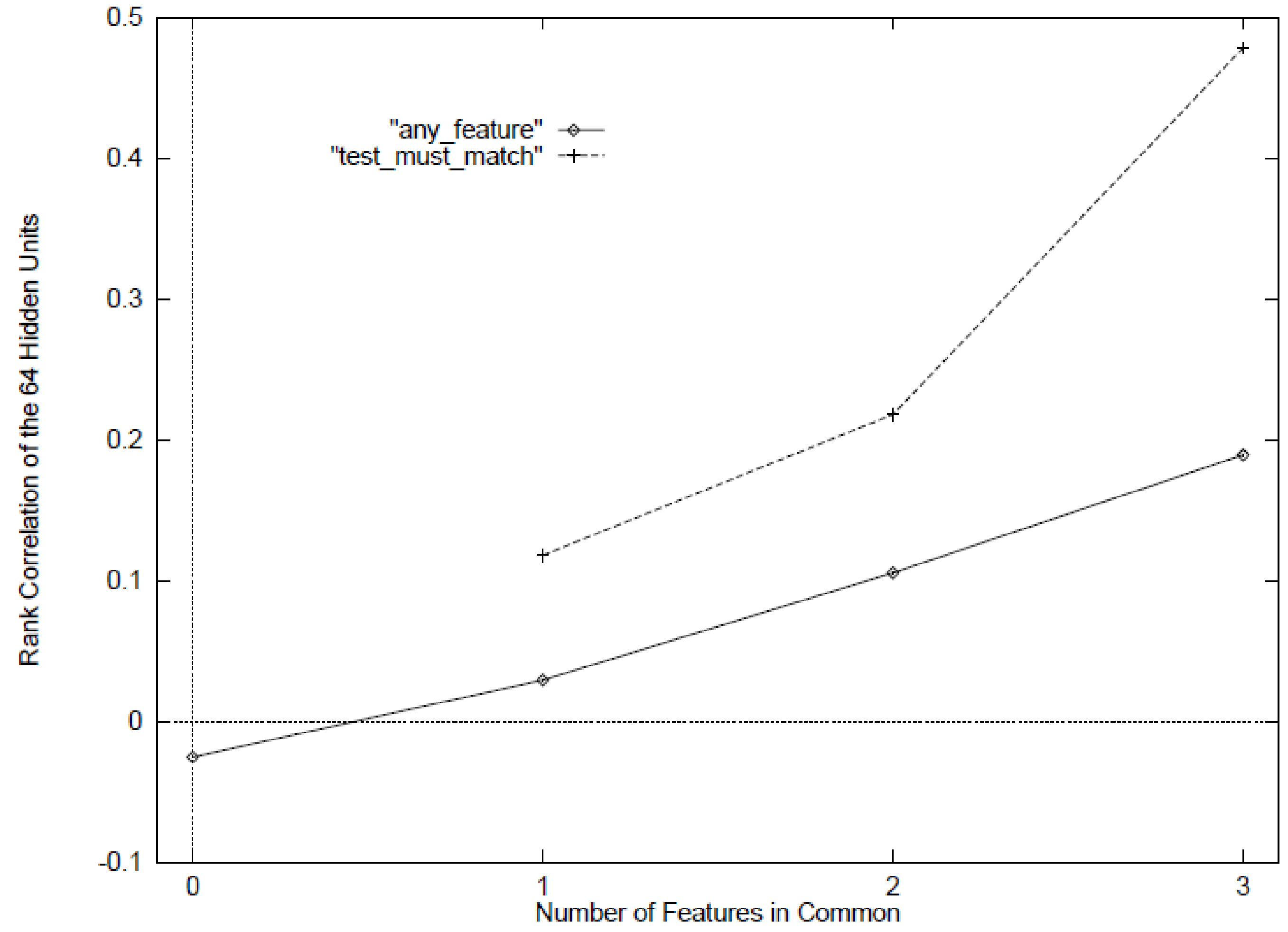
- Data amplification: the effective amount of data for learning each task increases
- Attribute selection: attributes unimportant for all tasks are learned to be left out
- “Eavesdropping”: suppose there is an intermediate feature easy to learn for task A but hard to learn for task B (because its involvement in task B is more complicated or because task B is “noisier”). If the feature gets learned for the benefit of task A, then task B can start to make use of it.
- Representation bias: tasks prefer hidden layer representations that other tasks prefer

Images from R. Caruana, Multitask Learning, Machine Learning, 1997

Basic multitask learning with neural networks, mechanisms

Amount of sharing of the intermediate features can indicate how related two tasks are

(Two lines in the picture: two different measures of task similarity)



Images from R. Caruana, Multitask Learning, Machine Learning, 1997

Basic multitask learning with neural networks, some scenarios

- Multiple simultaneous tasks may arise naturally from the problem domain. An early example (1986) was in modeling pronounciation of speech: phonemes and stress were learned at the same time
- The pneumonia case is an example of using **future measurements** (the extra lab measurements: features that could have been used as inputs if we were “in the future”) to help train predictions based on present measurements
- In time series prediction, often the same type of prediction can be made at multiple times (e.g. given past stock prices, predict stock price next day, next week, in 2 weeks, 3 weeks, ...) ----> the predictions for far-future times can be used as targets to help predict the nearest future.

Images from R. Caruana, Multitask Learning, Machine Learning, 1997

Basic multitask learning with neural networks, some scenarios

- Sometimes extra targets are things that could be computed or measured at the current time, but they are computationally hard or otherwise expensive to acquire fast enough in an online running system (e.g. they might need human expertise) ---> we can acquire them for an offline training set, and use them as multitask targets to train a predictor based on easily-acquired features
- Single-task learning might end up using only “easy” strongly active features of the data; multitask learning could be designed to make the model pay more attention to features that it might otherwise ignore, by designing additional tasks that rely on those features
- Sometimes we have models/predictors for tasks learned earlier, but don't have the earlier data anymore. Then we can use the stored predictors to generate additional targets for our current data

Images from R. Caruana, Multitask Learning, Machine Learning, 1997

Basic multitask learning with neural networks, some scenarios

- Sometimes the targets to learn are quantized (e.g. exam score); it may be useful to use a non-quantized or less quantized variable (e.g. exam points) as additional targets
  - For example in the pneumonia case, length of stay in the hospital may be a useful extra variable when predicting the risk rank of patients; it is not available for new patients yet but may be available for the training data set.

Images from R. Caruana, Multitask Learning, Machine Learning, 1997

Other model families with similar multitask learning approaches

- K-nearest neighbor regression: predict output for a new example, as a function of the outputs of the nearest training examples (e.g. average)
  - Which inputs are nearest depends on the metric (e.g. Mahalanobis metric or feature weighting)
  - The metric can be learned for several simultaneous tasks
- Kernel regression: similar to K-nearest neighbor regression, kernel function may include parameters that can be learned for several simultaneous tasks
- Decision trees: each branch of the tree split the data space into two or more parts--->in total the tree partitions the data space into ever-smaller areas; the splits can be optimized to serve several tasks simultaneously

References

- Caruana, R. Multitask learning. Machine Learning, 28, 41-75, 1997.
- Silver, D.L., Poirier, R., and Currie, D. Inductive transfer with context-sensitive neural networks. Machine learning, 73: 313-336, 2008.
- R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. ICML 2008.