Multitask Learning

Lei Tang

Arizona State University

Nov. 6th, 2006

Current Machine Learning

Typical Classification Setting

- Given some labeled data, use some learning algorithm (kNN, SVM, Naïve Bayes Classifier, decision tree) to build a model.
- widely used for face recognition, object detection, text categorization
- But most learning methods fail when number of training examples are rare!!
- Each task is single-purposed.
- Can we achieve better if we have multiple related tasks?

Current Machine Learning

Typical Classification Setting

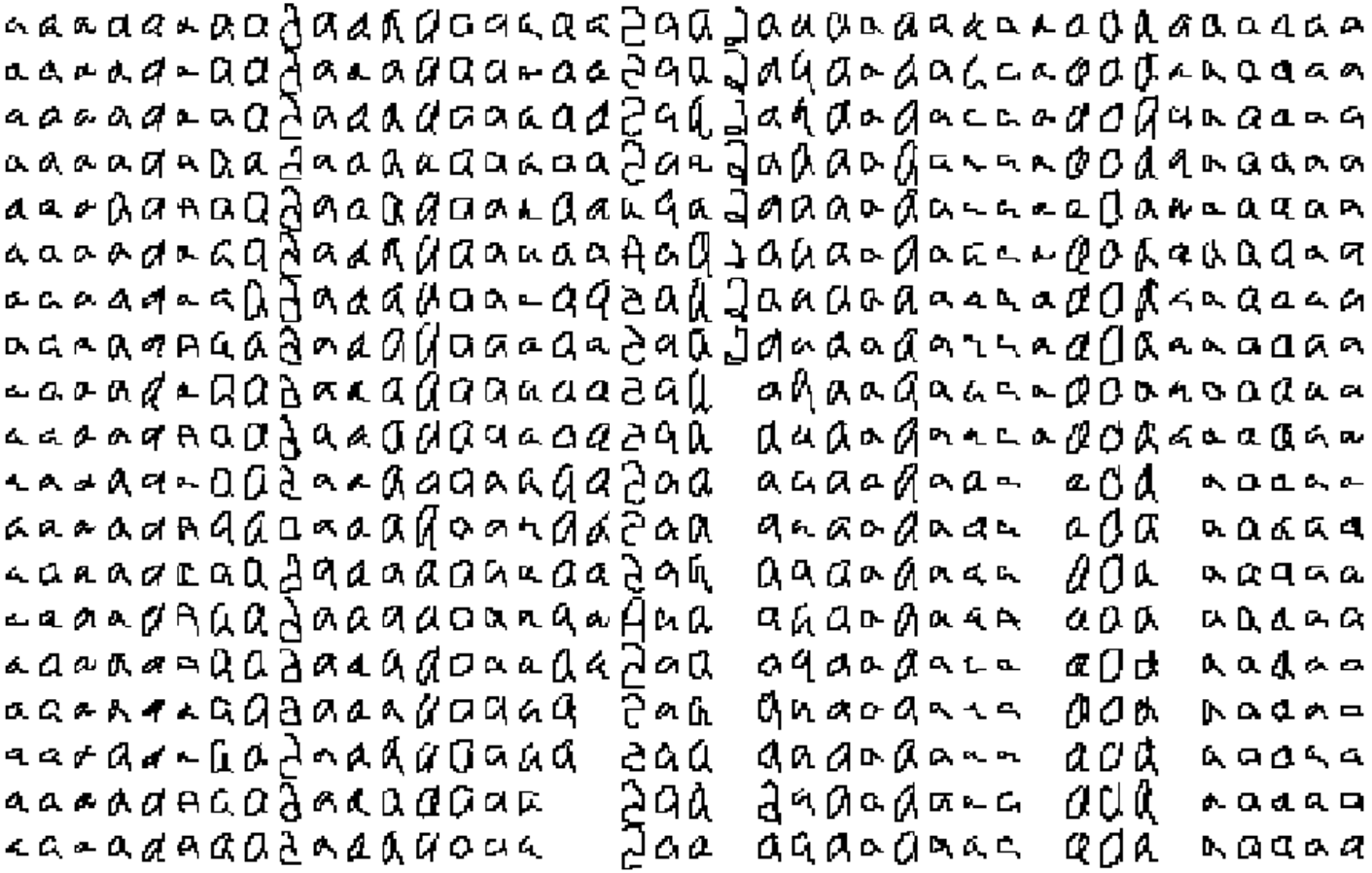
- Given some labeled data, use some learning algorithm (kNN, SVM, Naïve Bayes Classifier, decision tree) to build a model.
- widely used for face recognition, object detection, text categorization
- But most learning methods fail when number of training examples are rare!!
- Each task is single-purposed.
- Can we achieve better if we have multiple related tasks?

Current Machine Learning

Typical Classification Setting

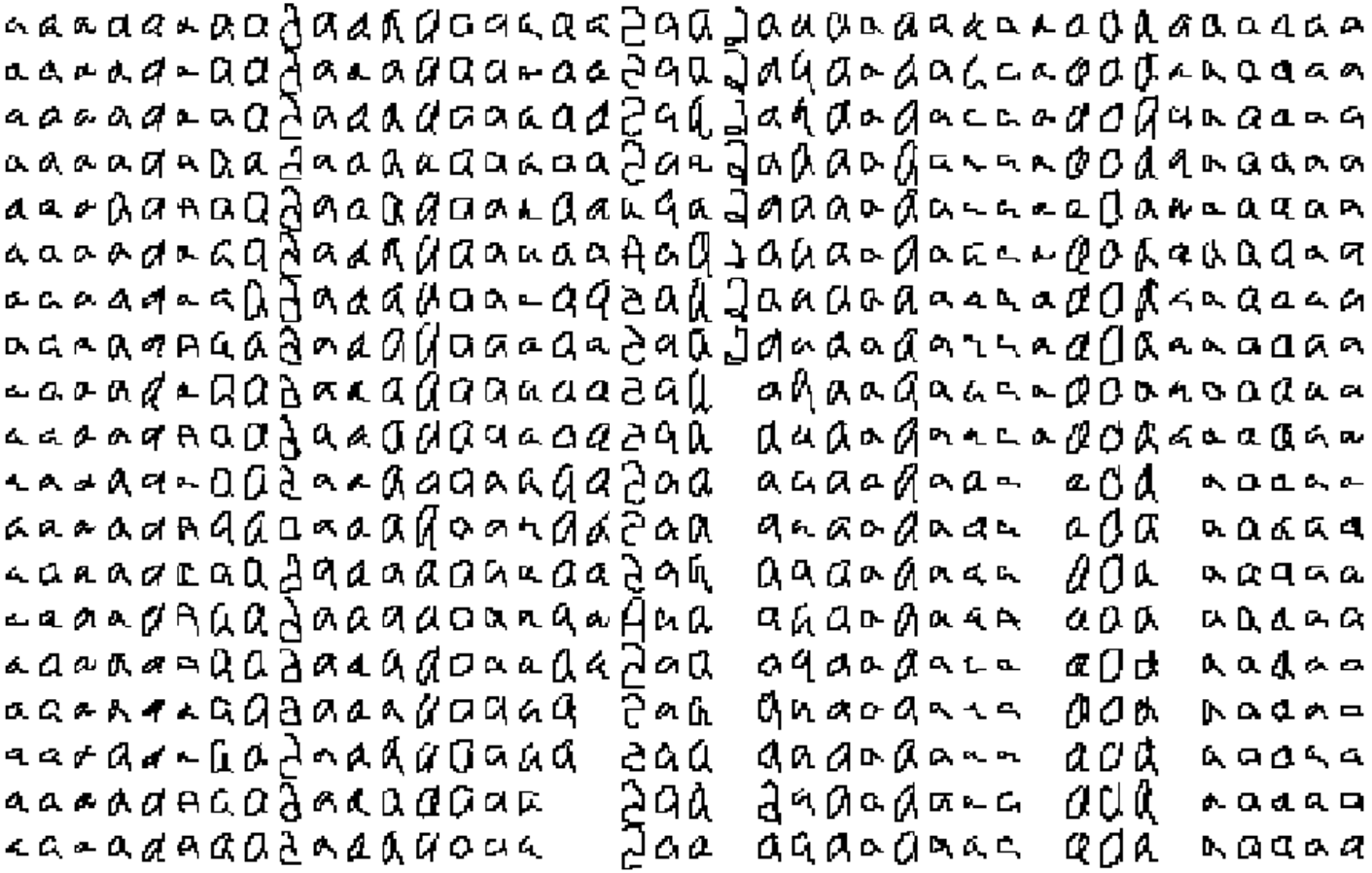
- Given some labeled data, use some learning algorithm (kNN, SVM, Naïve Bayes Classifier, decision tree) to build a model.
- widely used for face recognition, object detection, text categorization
- But most learning methods fail when number of training examples are rare!!
- Each task is single-purposed.
- Can we achieve better if we have multiple related tasks?

Letter *a* by 40 different writers



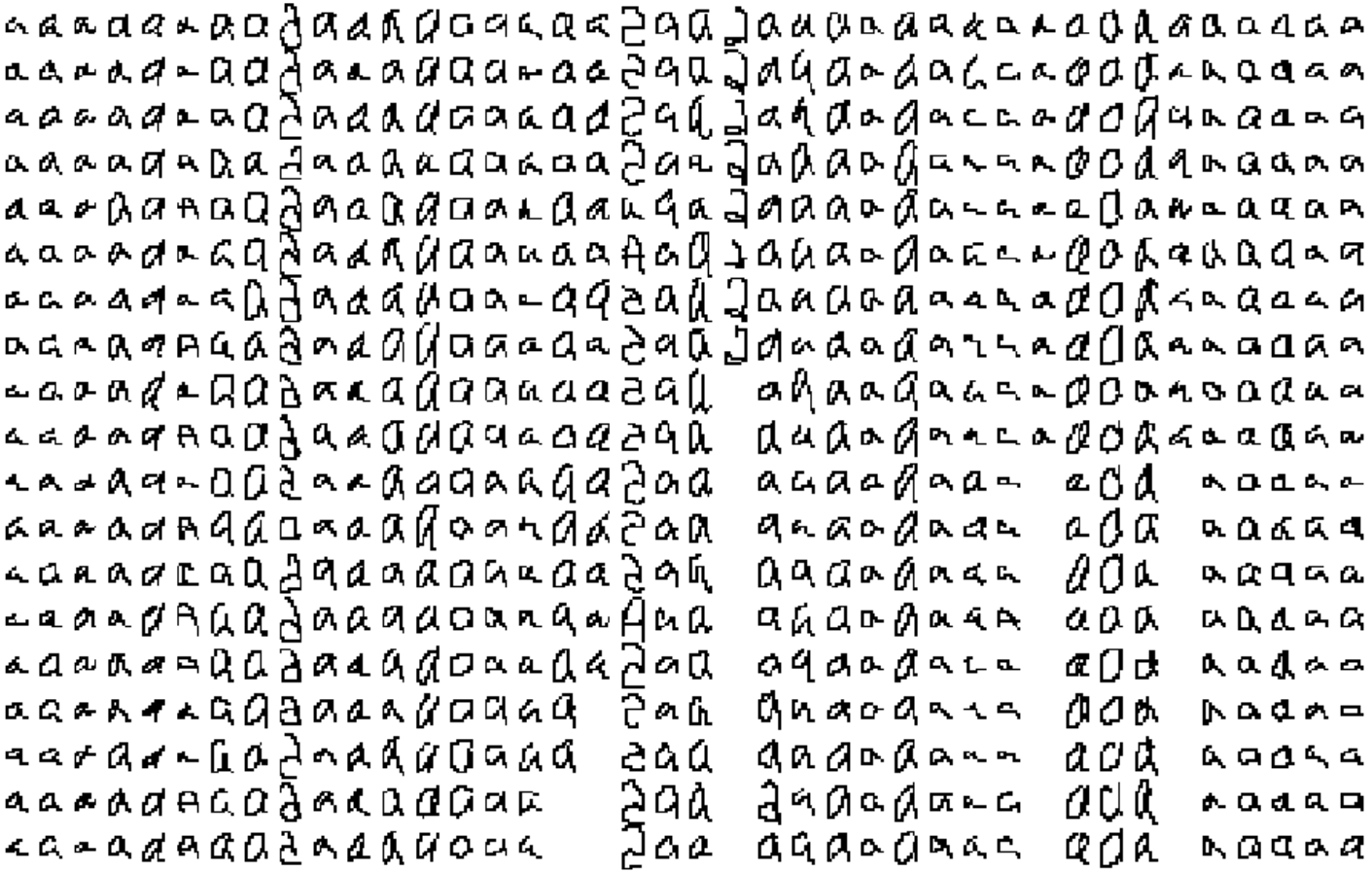
- Quite different writing style
- Very few examples per task (person)
- Is it possible to achieve better result by borrowing strength from each other?

Letter *a* by 40 different writers



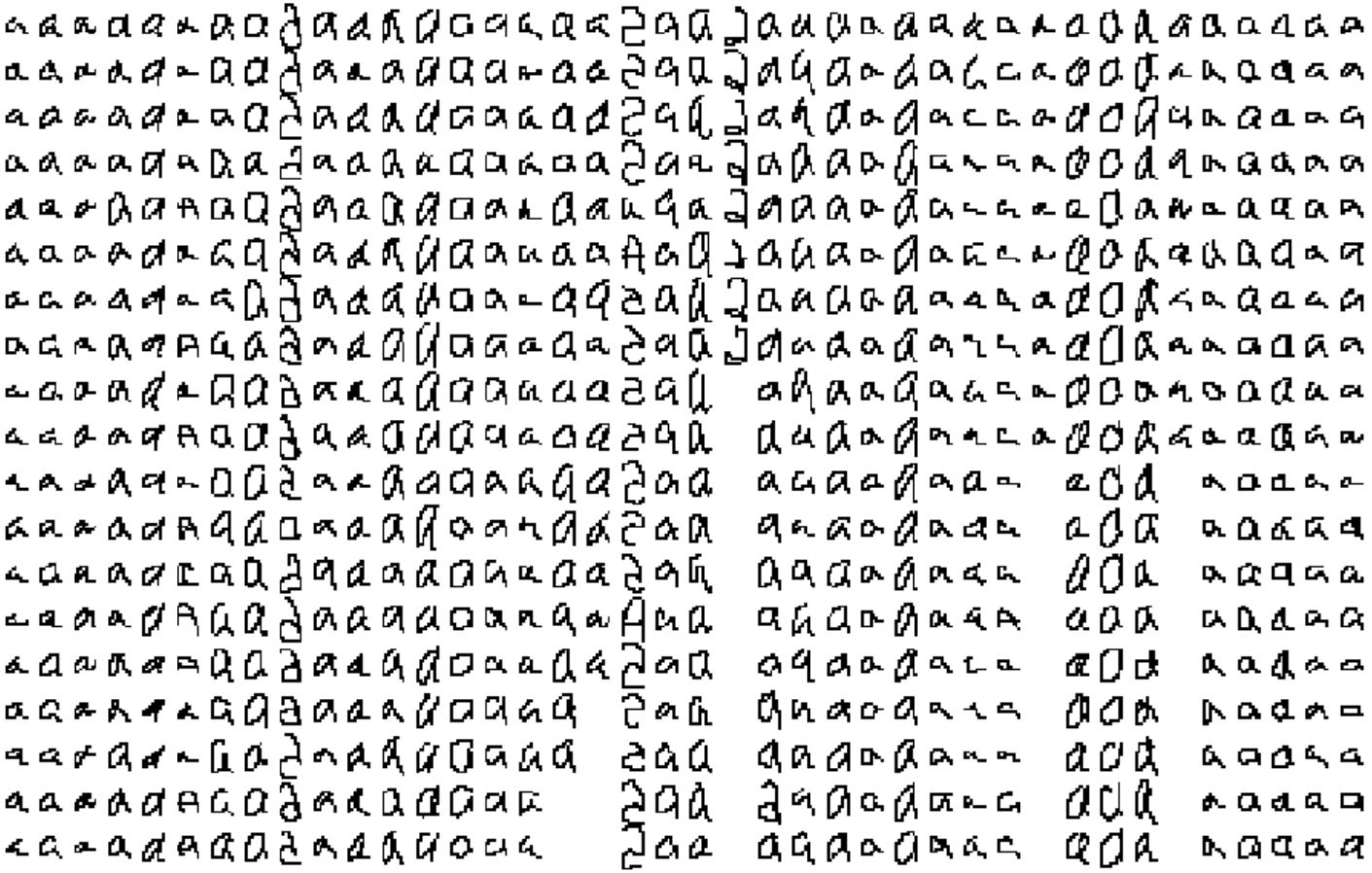
- Quite different writing style
- Very few examples per task (person)
- Is it possible to achieve better result by borrowing strength from each other?

Letter *a* by 40 different writers



- Quite different writing style
- Very few examples per task (person)
- Is it possible to achieve better result by borrowing strength from each other?

Letter *a* by 40 different writers



- Quite different writing style
- Very few examples per task (person)
- Is it possible to achieve better result by borrowing strength from each other?

Multiple Related Tasks

- Speech recognition for different speakers
- Character recognition for different writers
- Control a robot arm for different object grasping tasks
- Driving in different landscapes
- Text categorization of different corpus
- Natural Language Processing
- Computer Vision
- Concept Drift
- Collaborative Filtering
- Multi-class classification problem
- Spam filtering

Multitask Learning

- Multitask Learning: Given multiple **related tasks**, learn all tasks **simultaneously**.
- counterpart: single-task learning

Multitask Learning vs. Transfer Learning

- Similar concept: *transfer learning*(A.K.A *inductive bias transfer*, *learning to learn*, *life-long learning*)
- Transfer learning is incremental-oriented while multitask learning is batch-oriented.
- Transfer learning is more general than multitask learning (within-domain transfer, cross-domain transfer, lateral transfer, vertical transfer).
- Multitask Learning requires the same feature representation for all the tasks.

Multitask Learning

- Multitask Learning: Given multiple **related tasks**, learn all tasks **simultaneously**.
- counterpart: single-task learning

Multitask Learning vs. Transfer Learning

- Similar concept: *transfer learning*(A.K.A *inductive bias transfer*, *learning to learn*, *life-long learning*)
- Transfer learning is incremental-oriented while multitask learning is batch-oriented.
- Transfer learning is more general than multitask learning (within-domain transfer, cross-domain transfer, lateral transfer, vertical transfer).
- Multitask Learning requires the same feature representation for all the tasks.

Multitask Learning

- Multitask Learning: Given multiple **related tasks**, learn all tasks **simultaneously**.
- counterpart: single-task learning

Multitask Learning vs. Transfer Learning

- Similar concept: *transfer learning*(A.K.A *inductive bias transfer*, *learning to learn*, *life-long learning*)
- Transfer learning is incremental-oriented while multitask learning is batch-oriented.
- Transfer learning is more general than multitask learning (within-domain transfer, cross-domain transfer, lateral transfer, vertical transfer).
- Multitask Learning requires the same feature representation for all the tasks.

Multitask Learning

- Multitask Learning: Given multiple **related tasks**, learn all tasks **simultaneously**.
- counterpart: single-task learning

Multitask Learning vs. Transfer Learning

- Similar concept: *transfer learning*(A.K.A *inductive bias transfer*, *learning to learn*, *life-long learning*)
- Transfer learning is incremental-oriented while multitask learning is batch-oriented.
- Transfer learning is more general than multitask learning (within-domain transfer, cross-domain transfer, lateral transfer, vertical transfer).
- Multitask Learning requires the same feature representation for all the tasks.

Multitask Learning

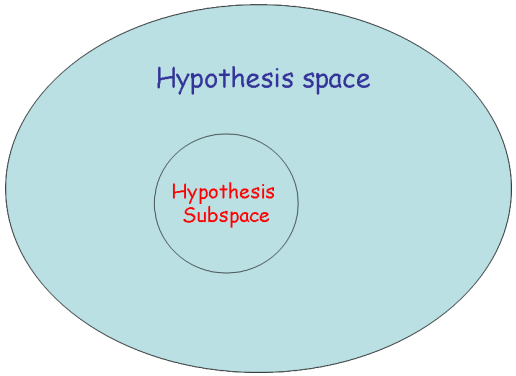
- Multitask Learning: Given multiple **related tasks**, learn all tasks **simultaneously**.
- counterpart: single-task learning

Multitask Learning vs. Transfer Learning

- Similar concept: *transfer learning*(A.K.A *inductive bias transfer*, *learning to learn*, *life-long learning*)
- Transfer learning is incremental-oriented while multitask learning is batch-oriented.
- Transfer learning is more general than multitask learning (within-domain transfer, cross-domain transfer, lateral transfer, vertical transfer).
- Multitask Learning requires the same feature representation for all the tasks.

Why multitask learning is better?

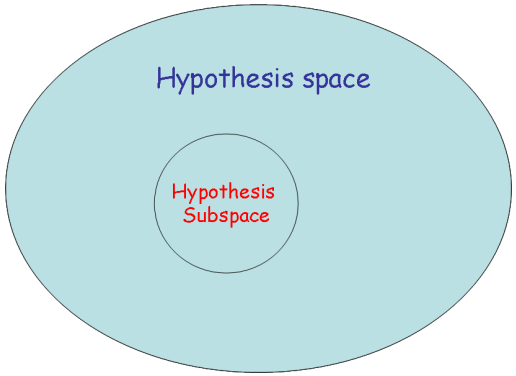
1. Typical machine Learning: bias is used to guide the search in the hypothesis space during learning.



2. Multitask learning can be considered as a bias learning procedure. (Find a proper hypothesis subspace applicable for all tasks).
3. Employ the data in all tasks, thus actually increasing the number of data

Why multitask learning is better?

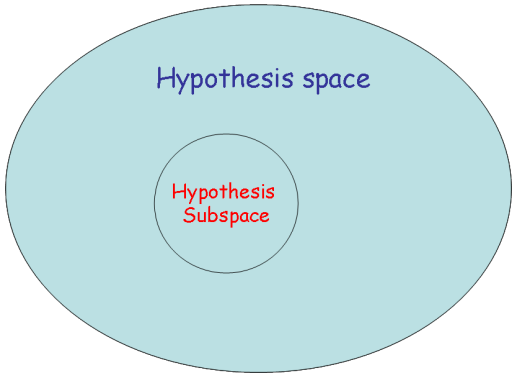
1. Typical machine Learning: bias is used to guide the search in the hypothesis space during learning.



2. Multitask learning can be considered as a bias learning procedure. (Find a proper hypothesis subspace applicable for all tasks).
3. Employ the data in all tasks, thus actually increasing the number of data

Why multitask learning is better?

1. Typical machine Learning: bias is used to guide the search in the hypothesis space during learning.



2. Multitask learning can be considered as a bias learning procedure. (Find a proper hypothesis subspace applicable for all tasks).
3. Employ the data in all tasks, thus actually increasing the number of data

Multitask Learning Approaches

1. MTL by sharing distance metric
2. MTL by sharing common feature set
3. MTL by sharing internal representation
4. MTL by sharing priors
5. MTL by sharing manifold in predictor space

A Toy Example

- Tasks: To recognize letter **a** written by three different people: Alice, Bob, and Caleb
- Each image provides three features:
  - O: whether there's a circle in the image
  - $\sim$: whether there's a tail
  - θ: whether the circle is cut into two parts.
- a decision function is adopted:

$$f(x)\begin{cases} > 0 & is \quad \mathbf{a} \\ < 0 & not \quad \mathbf{a} \end{cases}$$

MTL by sharing distance metric

- A distance metric is defined over all tasks.
- Objective goal: the data of the same class are close while those of different classes are far away.
- Map the original input space to another space and define a proper distance metric.
- For the toy example, we can define a distance as

$$dist(x, x') = ||g(x) - g(x')||_2$$

$$where \quad g(x) = w_1 O + w_2 \sim + w_3 \theta$$

- Typical distance metric learning methods can be used.
- A classifier which employs the distance directly (kNN, kernel classifiers) is used.

MTL by sharing distance metric

- A distance metric is defined over all tasks.
- Objective goal: the data of the same class are close while those of different classes are far away.
- Map the original input space to another space and define a proper distance metric.
- For the toy example, we can define a distance as

$$dist(x, x') = ||g(x) - g(x')||_2$$

$$where \quad g(x) = w_1 O + w_2 \sim + w_3 \theta$$

- Typical distance metric learning methods can be used.
- A classifier which employs the distance directly (kNN, kernel classifiers) is used.

MTL by sharing distance metric

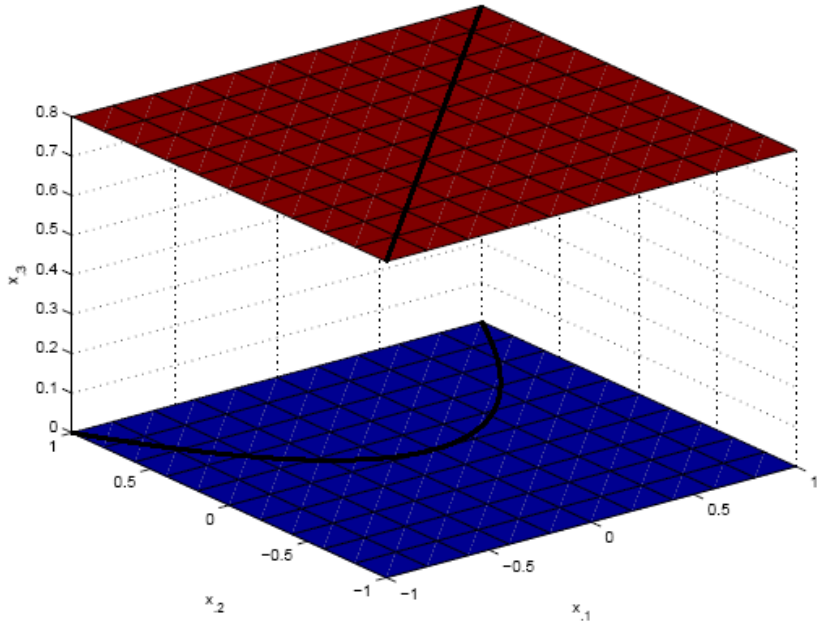
- A distance metric is defined over all tasks.
- Objective goal: the data of the same class are close while those of different classes are far away.
- Map the original input space to another space and define a proper distance metric.
- For the toy example, we can define a distance as

$$dist(x, x') = ||g(x) - g(x')||_2$$

$$where \quad g(x) = w_1 O + w_2 \sim + w_3 \theta$$

- Typical distance metric learning methods can be used.
- A classifier which employs the distance directly (kNN, kernel classifiers) is used.

MTL by sharing common feature set

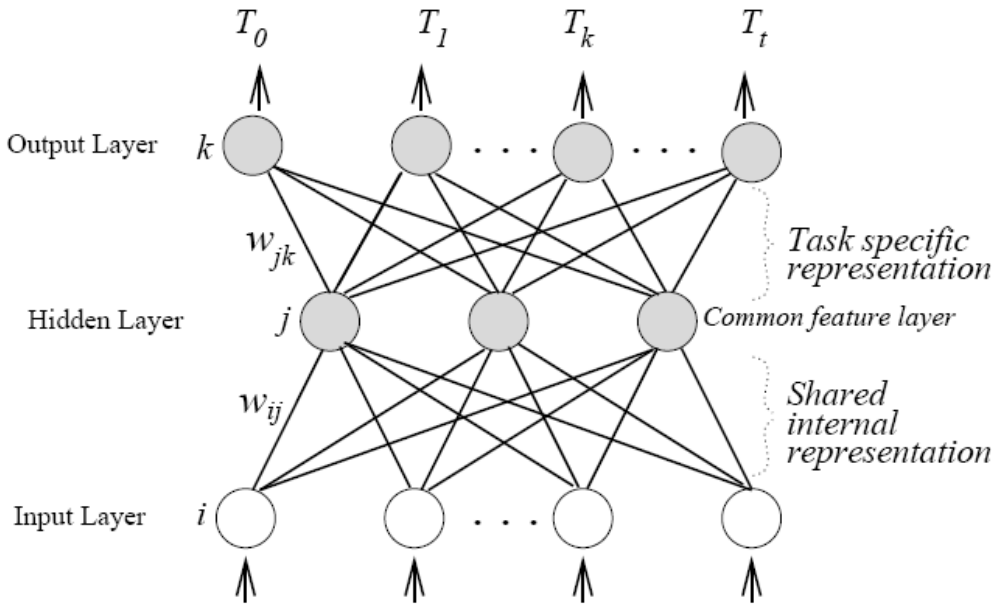


- Optical character recognition of different people. Focusing on some common feature sets; avoid selecting too many specific features.
- Methods:
  - boosting methods to do greedy search
  - different forms of norm
  - assign an indicator variable for each feature

MTL by sharing internal representation

Sharing the internal features after mapping.

$$
\begin{aligned}
\textit{Alice}: \quad & f(x) = w_{A1}(0.8O + 0.9 \sim) + w_{A2}(0.40O - 0.5\theta) \\
\textit{Bob}: \quad & f(x) = w_{B1}(0.8O + 0.9 \sim) + w_{B2}(0.40O - 0.5\theta) \\
\textit{Caleb}: \quad & f(x) = w_{C1} \underbrace{(0.8O + 0.9 \sim)}_{F_1} + w_{C2} \underbrace{(0.40O - 0.5\theta)}_{F_2}
\end{aligned}
$$



MTL by sharing priors

$$
\begin{aligned}
\textit{Alice}: &\quad 0.7 \cdot O + 0.6 \cdot \sim -0.2 \cdot \theta \\
\textit{Bob}: &\quad 0.6 \cdot O + 0.8 \cdot \sim -0.4 \cdot \theta \\
\textit{Caleb}: &\quad 0.6 \cdot O + 0.7 \cdot \sim -0.3 \cdot \theta
\end{aligned}
$$

Method

Model the parameter of all tasks by some prior distribution(eg. $N(\mu, \Sigma)$), then μ, Σ are hyper-parameters shared across all tasks. To estimate hyper-parameters:

1. Calculate the average and variance directly based on multiple tasks.
2. Formulate the likelihood of data in all tasks; maximize it or (penalized likelihood) with respect to hyper-parameters. *penalized likelihood = likelihood × prior(hyper-parameters)*

MTL by sharing priors

$$
\begin{aligned}
\textit{Alice}: &\quad 0.7 \cdot O + 0.6 \cdot \sim -0.2 \cdot \theta \\
\textit{Bob}: &\quad 0.6 \cdot O + 0.8 \cdot \sim -0.4 \cdot \theta \\
\textit{Caleb}: &\quad 0.6 \cdot O + 0.7 \cdot \sim -0.3 \cdot \theta
\end{aligned}
$$

Method

Model the parameter of all tasks by some prior distribution(eg. $N(\mu, \Sigma)$), then μ, Σ are hyper-parameters shared across all tasks.
To estimate hyper-parameters:

1. Calculate the average and variance directly based on multiple tasks.
2. Formulate the likelihood of data in all tasks; maximize it or (penalized likelihood) with respect to hyper-parameters.
   penalized likelihood = likelihood × prior(hyper-parameters)

MTL by sharing priors

$$
\begin{aligned}
\textit{Alice}: &\quad 0.7 \cdot O + 0.6 \cdot \sim -0.2 \cdot \theta \\
\textit{Bob}: &\quad 0.6 \cdot O + 0.8 \cdot \sim -0.4 \cdot \theta \\
\textit{Caleb}: &\quad 0.6 \cdot O + 0.7 \cdot \sim -0.3 \cdot \theta
\end{aligned}
$$

Method

Model the parameter of all tasks by some prior distribution(eg. $N(\mu, \Sigma)$), then μ, Σ are hyper-parameters shared across all tasks.
To estimate hyper-parameters:

1. Calculate the average and variance directly based on multiple tasks.
2. Formulate the likelihood of data in all tasks; maximize it or (penalized likelihood) with respect to hyper-parameters.
   penalized likelihood $=$ *likelihood* $\times$ *prior(hyper-parameters)*

MTL by constructing manifold in predictor space

$$
\begin{aligned}
\textit{Alice}: &\quad 0.7 \cdot O + 0.6 \cdot \sim -0.2 \cdot \theta \\
\textit{Bob}: &\quad 0.6 \cdot O + 0.8 \cdot \sim -0.3 \cdot \theta \\
\textit{Caleb}: &\quad 0.6 \cdot O + 0.7 \cdot \sim -0.2 \cdot \theta
\end{aligned}
$$

One commonality shared by all the tasks is that

$$w_1 + w_2 - w_3 = 1.1$$

- Manifold actually represents a pattern in the predictor space.
- It can be a line, a curve, a hyperplane, a complicated manifold etc.
- Usually formulated as an optimization problem.
- Perform SVD in the predictor space; specify a required manifold etc.

MTL by constructing manifold in predictor space

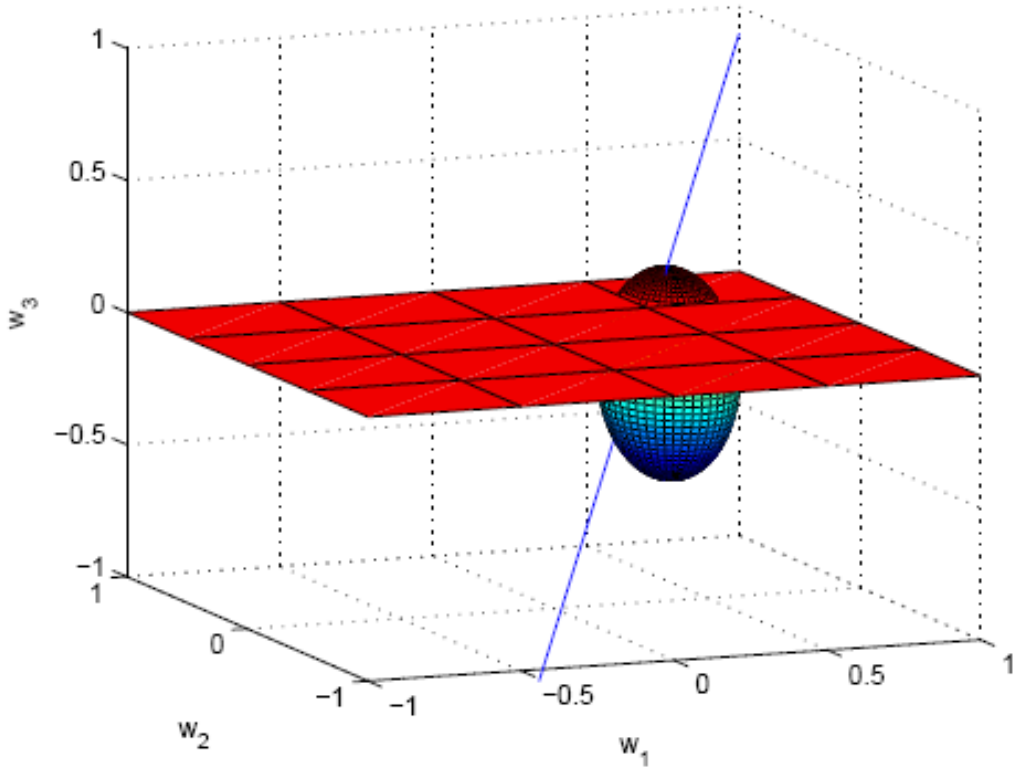
$$
\begin{aligned}
\textit{Alice}: &\quad 0.7 \cdot O + 0.6 \cdot \sim -0.2 \cdot \theta \\
\textit{Bob}: &\quad 0.6 \cdot O + 0.8 \cdot \sim -0.3 \cdot \theta \\
\textit{Caleb}: &\quad 0.6 \cdot O + 0.7 \cdot \sim -0.2 \cdot \theta
\end{aligned}
$$

One commonality shared by all the tasks is that

$$w_1 + w_2 - w_3 = 1.1$$

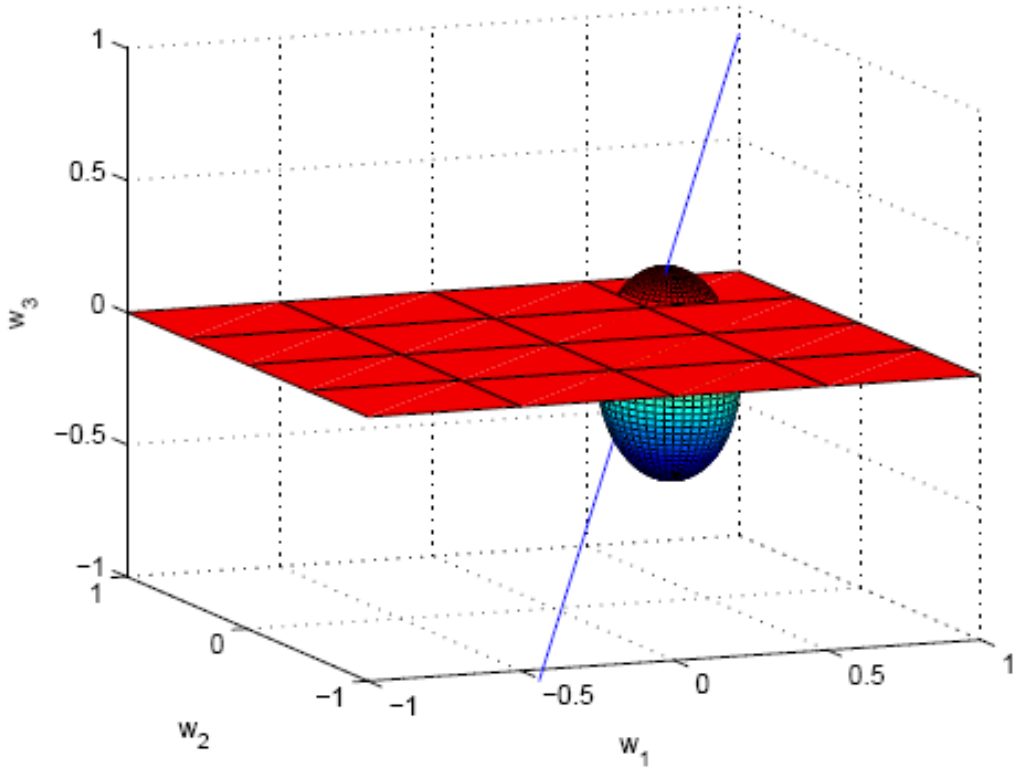
- Manifold actually represents a pattern in the predictor space.
- It can be a line, a curve, a hyperplane, a complicated manifold etc.
- Usually formulated as an optimization problem.
- Perform SVD in the predictor space; specify a required manifold etc.

A Unified View



$$\min_{\theta} \sum_{t=1}^{N} L(D^t, \theta) + \gamma C_S(H_\theta)$$

A Unified View



$$\min_{\theta} \sum_{t=1}^{N} L(D^t, \theta) + \gamma C_{\mathcal{S}}(H_\theta)$$

Pros and Cons of MTL

Pros

1. Improve classification accuracy (or some other similar measure) as a more reliable.
2. Improve learning speed.

Cons

- No general conclusion about approaches of MTL.
- Assumption: all the tasks are related. What if there are some unrelated tasks? unfortunately, dissimilar tasks might hurt the performance, the same as introducing noise.
- Existing methods treat all tasks equivalently, what if some tasks are more reliable? Need uneven task weighting.
- What if no related tasks are in hand? Generate automatically?
- Existing methods show that MTL improve performance mostly when data are scarce (Some papers just use 1-10 examples for training). What if I have 100, 200, 500, 1000, 1000000 data?

Pros and Cons of MTL

Pros

1. Improve classification accuracy (or some other similar measure) as a more reliable.
2. Improve learning speed.

Cons

- No general conclusion about approaches of MTL.
- Assumption: all the tasks are related. What if there are some unrelated tasks? unfortunately, dissimilar tasks might hurt the performance, the same as introducing noise.
- Existing methods treat all tasks equivalently, what if some tasks are more reliable? Need uneven task weighting.
- What if no related tasks are in hand? Generate automatically?
- Existing methods show that MTL improve performance mostly when data are scarce (Some papers just use 1-10 examples for training). What if I have 100, 200, 500, 1000, 1000000 data?

Pros and Cons of MTL

Pros

1. Improve classification accuracy (or some other similar measure) as a more reliable.
2. Improve learning speed.

Cons

- No general conclusion about approaches of MTL.
- Assumption: all the tasks are related. What if there are some unrelated tasks? unfortunately, dissimilar tasks might hurt the performance, the same as introducing noise.
- Existing methods treat all tasks equivalently, what if some tasks are more reliable? Need uneven task weighting.
- What if no related tasks are in hand? Generate automatically?
- Existing methods show that MTL improve performance mostly when data are scarce (Some papers just use 1-10 examples for training). What if I have 100, 200, 500, 1000, 1000000 data?

Pros and Cons of MTL

Pros

1. Improve classification accuracy (or some other similar measure) as a more reliable.
2. Improve learning speed.

Cons

- No general conclusion about approaches of MTL.
- Assumption: all the tasks are related. What if there are some unrelated tasks? unfortunately, dissimilar tasks might hurt the performance, the same as introducing noise.
- Existing methods treat all tasks equivalently, what if some tasks are more reliable? Need uneven task weighting.
- What if no related tasks are in hand? Generate automatically?
- Existing methods show that MTL improve performance mostly when data are scarce (Some papers just use 1-10 examples for training). What if I have 100, 200, 500, 1000, 1000000 data?

Pros and Cons of MTL

Pros

1. Improve classification accuracy (or some other similar measure) as a more reliable.
2. Improve learning speed.

Cons

- No general conclusion about approaches of MTL.
- Assumption: all the tasks are related. What if there are some unrelated tasks? unfortunately, dissimilar tasks might hurt the performance, the same as introducing noise.
- Existing methods treat all tasks equivalently, what if some tasks are more reliable? Need uneven task weighting.
- What if no related tasks are in hand? Generate automatically?
- Existing methods show that MTL improve performance mostly when data are scarce (Some papers just use 1-10 examples for training). What if I have 100, 200, 500, 1000, 1000000 data?

Pros and Cons of MTL

Pros

1. Improve classification accuracy (or some other similar measure) as a more reliable.
2. Improve learning speed.

Cons

- No general conclusion about approaches of MTL.
- Assumption: all the tasks are related. What if there are some unrelated tasks? unfortunately, dissimilar tasks might hurt the performance, the same as introducing noise.
- Existing methods treat all tasks equivalently, what if some tasks are more reliable? Need uneven task weighting.
- What if no related tasks are in hand? Generate automatically?
- Existing methods show that MTL improve performance mostly when data are scarce (Some papers just use 1-10 examples for training). What if I have 100, 200, 500, 1000, 1000000 data?

Pros and Cons of MTL

Pros

1. Improve classification accuracy (or some other similar measure) as a more reliable.
2. Improve learning speed.

Cons

- No general conclusion about approaches of MTL.
- Assumption: all the tasks are related. What if there are some unrelated tasks? unfortunately, dissimilar tasks might hurt the performance, the same as introducing noise.
- Existing methods treat all tasks equivalently, what if some tasks are more reliable? Need uneven task weighting.
- What if no related tasks are in hand? Generate automatically?
- Existing methods show that MTL improve performance mostly when data are scarce (Some papers just use 1-10 examples for training). What if I have 100, 200, 500, 1000, 1000000 data?

Pros and Cons of MTL

Pros

1. Improve classification accuracy (or some other similar measure) as a more reliable.
2. Improve learning speed.

Cons

- No general conclusion about approaches of MTL.
- Assumption: all the tasks are related. What if there are some unrelated tasks? unfortunately, dissimilar tasks might hurt the performance, the same as introducing noise.
- Existing methods treat all tasks equivalently, what if some tasks are more reliable? Need uneven task weighting.
- What if no related tasks are in hand? Generate automatically?
- Existing methods show that MTL improve performance mostly when data are scarce (Some papers just use 1-10 examples for training). What if I have 100, 200, 500, 1000, 1000000 data?

Let's rock!!

Multitask learning is not new. Its original idea dated back to 1980s. But still lots of open problems.