Representation Learning

Transfer Learning and Domain Adaptation

The scenario

- *Transfer Learning* and *Domain Adaptation* refer to the situation where what has been learned in one setting (i.e., distribution P_1) is exploited to improve generalization in another (say, Distribution P_2)
- It generalizes the idea previously seen in greedy unsupervised pretraining
  - Where we transfer representations between an unsupervised learning task and a supervised learning task

Transfer Learning

- In transfer learning the learner must perform two or more different tasks
- But we assume that the factors that explain the variations in P_1 are relevant to the variations to be captured for learning P_2

Example of transfer learning

- Supervised learning context
  - Input is the same, but target is different
  - There is more data in distribution P_1 and very few in distribution P_2
- Visual classification
  - Significantly more data in distribution sampled from $\mathrm{cats\ and\ dogs}$
  - Then learn to quickly generalize to $\mathrm{ants\ and\ wasps}$
  - Visual categories share low-level notions of edges and visual shapes, geometric changes, lighting

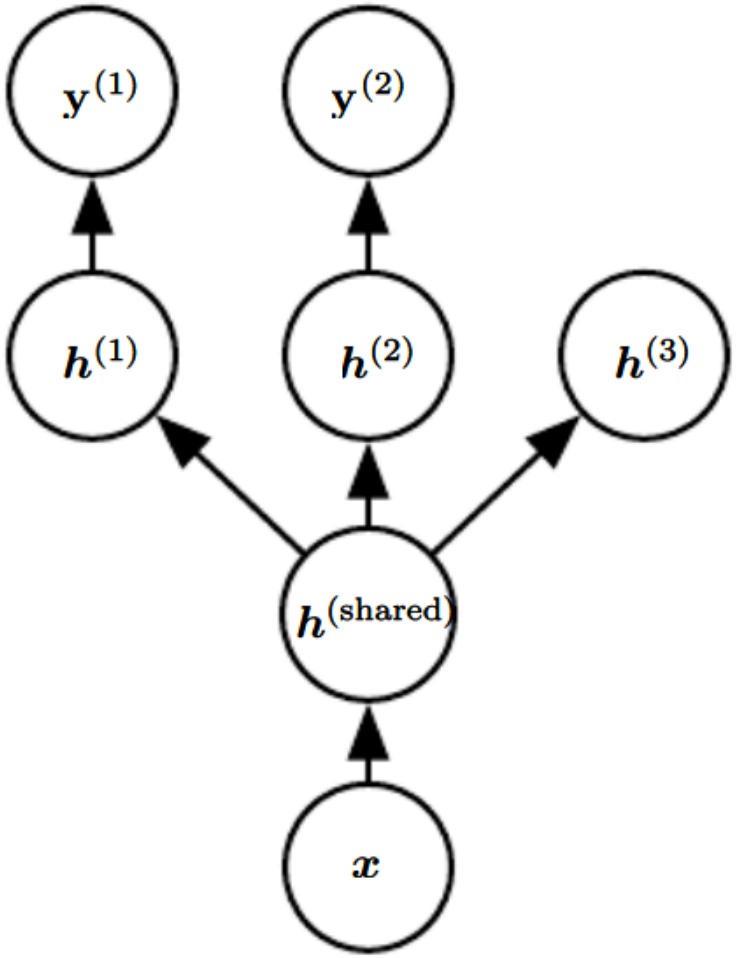
Shared Semantics of Input

- Transfer learning, multi-task learning and domain adaptation are achieved via representation learning
  - Where there exist features that are useful for different tasks or settings
  - This is illustrated next, with shared lower layers and task dependent upper layers

Multi-task learning (shared input)

– Tasks share a common input but involve different target random variables

- Task specific parameters (weights into and from $\boldsymbol{h}^{(1)}$ and $\boldsymbol{h}^{(2)}$ can be learned on top of those yielding $\boldsymbol{h}^{(\text{shared})}$
- In the unsupervised context some top level factors $\boldsymbol{h}^{(3)}$ are associated with none of the tasks

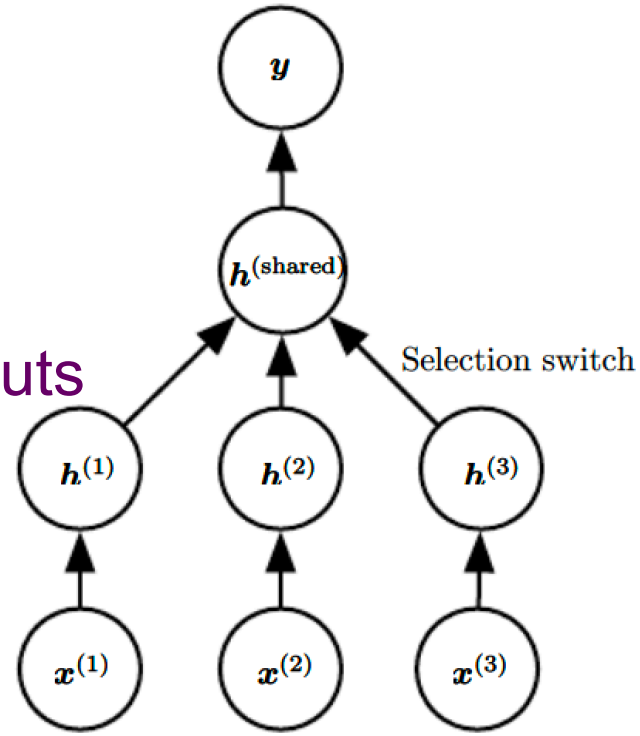


Shared semantics of Output

- What is shared among different tasks is not the semantics of the input but the semantics of the output
- Ex: speech recognition
  - Need to produce valid sentences at the output
  - Earlier layers near input need to recognize very different versions of input phonemes depending on person speaking

Transfer Learning (shared output)

- When output variable y has the same semantics for all classes
  - $\boldsymbol{x}$ has different meaning
  - dimension for each task
    - Three tasks $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$ and $\boldsymbol{x}^{(3)}$ are inputs



- Lower levels upto selection switch are task-specific
  - – Upper levels are shared
    - Semantics of output are shared, not semantics of input

      as in speech recognition where vocalizations are based on different speakers

Success of Transfer Learning

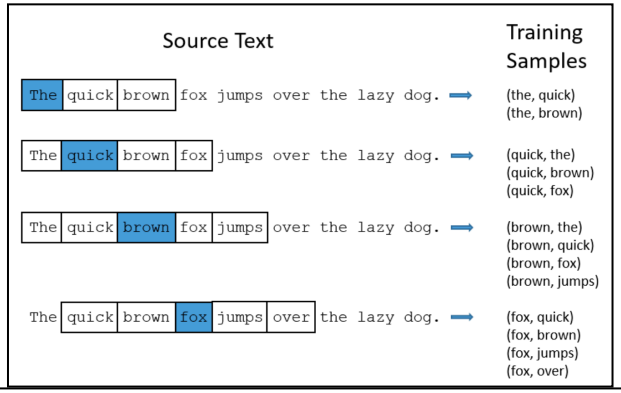
- Unsupervised deep learning for transfer learning has found success in ML competitions
  - Each participant is given data from distribution P_1 illustrating some set of categories
  - Participants learn a feature space
    - Mapping raw input to a representation space
  - This transformation is applied to samples from P_2
  - A linear classifier is trained from very few samples
- As deeper representations used (learned purely unsupervised from P_1) performance improves
  - For deeper representations fewer samples needed

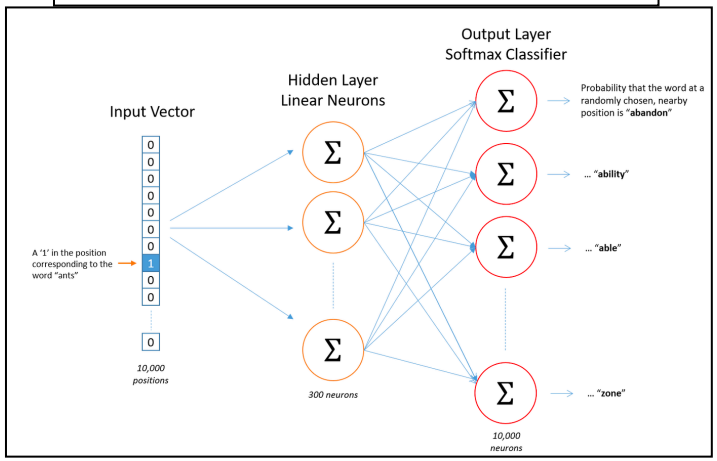
Domain Adaptation

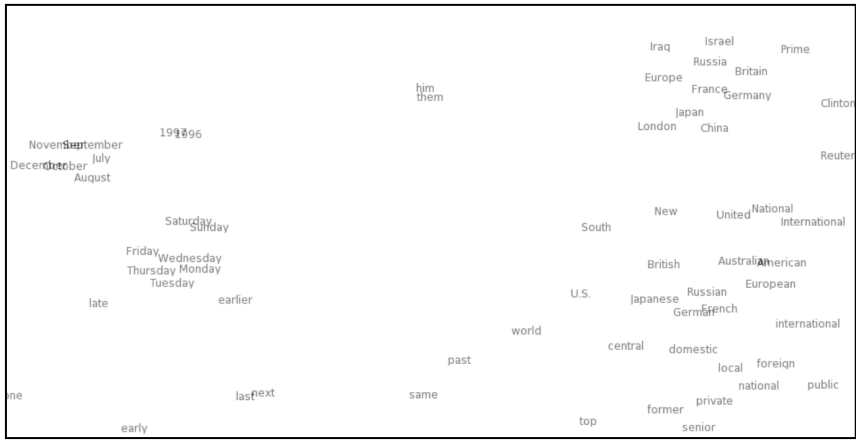
- Related to transfer learning
- Optimal input-to-output mapping remains the same between each setting
- But the input distribution is slightly different
- Ex: In *sentiment analysis*, moving from domain of media (books/music/videos) to domain of consumer electronics (TV/smartphones)

NLP: Word to Vec

- Training Data
- Word-to-vec
  - One-hot vector of words
    - With 30,000 elements
  - mapped to $\boldsymbol{h}$
    - vector of 300
- Word embedding
  - Similar words are close together







Examples of Words-to-vecs

Represent noun by co-occurrences with 25 verbs*

Semantic feature values:
"**celery**"
0.8368, eat
0.3461, taste
0.3153, fill
0.2430, see
0.1145, clean
0.0600, open
0.0586, smell
0.0286, touch
…
…
0.0000, drive
0.0000, wear
0.0000, lift
0.0000, break
0.0000, ride

Semantic feature values:
"**airplane**"
0.8673, ride
0.2891, see
0.2851, say
0.1689, near
0.1228, open
0.0883, hear
0.0771, run
0.0749, lift
…
…
0.0049, smell
0.0010, wear
0.0000, taste
0.0000, rub
0.0000, manipulate

* in a trillion word text collection

Example of Domain Adaptation

- Sentiment Analysis Task
- Determine if comment is positive/negative
  - Sentiment predictor is trained on customer reviews of media content such as books, videos and music
  - Later used to analyze comments about consumer electronics such as televisions and smartphones
  - Vocabulary and style may vary from one domain to other
  - Simple unsupervised pretraining (with denoising autoencoders) found useful with domain adaptation

Concept Drift

- Related to Domain Adaptation is Concept Drift
  - A form of transfer learning where there are gradual changes in data distribution over time
- Both concept drift and transfer learning can be regarded as different forms of multi-task learning
  - Multitask Learning typically refers to supervised learning
  - Transfer Learning is applicable to unsupervised learning and reinforcement learning as well

Success in transfer learning

- Unsupervised deep learning for transfer learning: successful in ML competitions
  - Participant given data set from first setting (from distribution P_1) illustrating examples from some set of categories to learn a good feature space
  - Learned transformation applied to inputs from transfer setting (distribution P_2), a linear classifier trained to generalize well from few labeled samples
- With deeper representations in learning from P_1, learning curve on P_2 becomes much better

Two extreme forms of transfer learning

- One-shot learning
  - Only one labeled sample example of the transfer task is given
- Zero-shot learning
  - No labeled samples are given for the transfer task

One-shot learning

- Only one labeled example of the transfer task
  - Possible because the representation learns to cleanly separate underlying classes during Stage 1
  - During transfer learning, only one labeled example is needed to infer the label of many possible test examples that cluster around the same point in representation space
- Works to the extent that factors of variation corresponding to these invariances have been cleanly separated from the other factors in the learned representation space

Zero-shot learning

- No labeled examples
- Ex: A learner reads a large collection of text and then solves object recognition problems
  - Having read that a cat has four legs and pointed ears, learner guesses that an image is a cat without having seen a cat before

Zero-data learning explained

- Possible because additional data exploited
- Zero-data learning scenario includes three random variables
  1. Traditional inputs $\boldsymbol{x}$
     - Unlabeled text data containing sentences such as “cats have four legs”, “cats have pointy ears’)
  2. Traditional outputs y ($y{=}1$ indicating yes, $y{=}0$ for no)
  3. Description of task T (represents questions to be answered)
     - Is there a cat in this image?
- Model trained to determine conditional $p(y|\boldsymbol{x}, T)$

Type of Representation of T

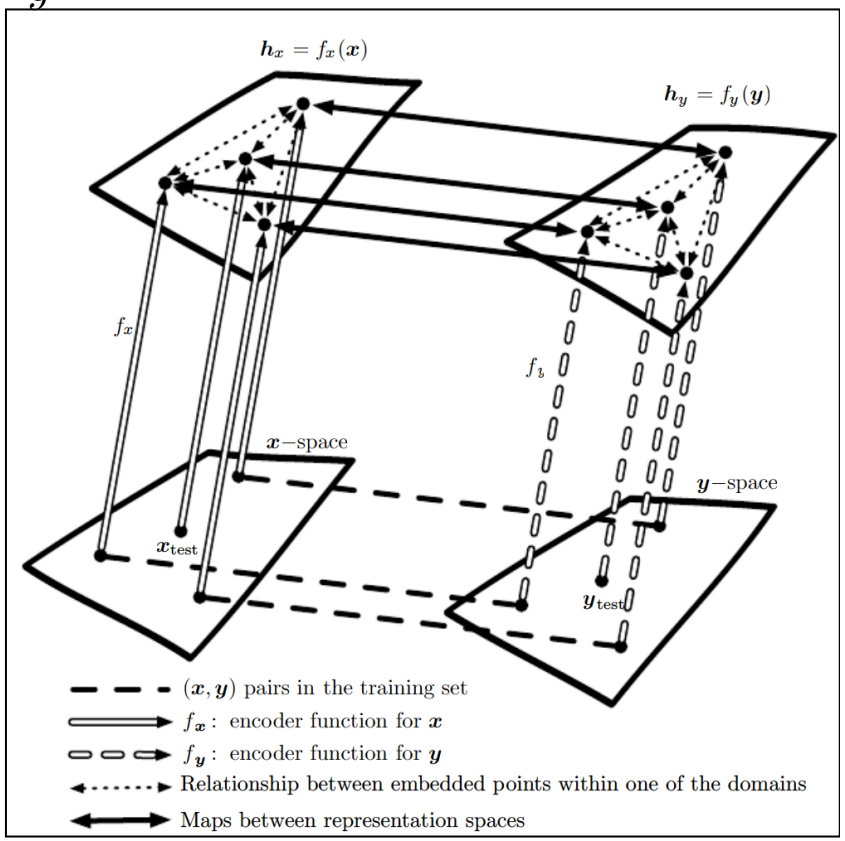
- Zero-shot learning requires T to be represented in a way that allows some sort of generalization
  - T cannot be just a one-hot code indicating an object category
  - Instead a distributed representation of object categories by using a learned word embedding for the word associated with each category

Similar phenomenon in Machine Translation

- We have words in one language
  - Word relationships learned from a unilingual corpus
- We have translated sentences that relate words in one language with words in the other
- No labeled word translations available
  - i.e., word A in language $\boldsymbol{X}$ to word B in language $\boldsymbol{Y}$
- Can guess a translation for word A because
  - We have learned distributed representations for words in $\boldsymbol{X}$ and for words in $\boldsymbol{Y}$ then created a link relating the two spaces via training examples of matched pairs of sentences
    - Works best when two representations and relations are learned jointly

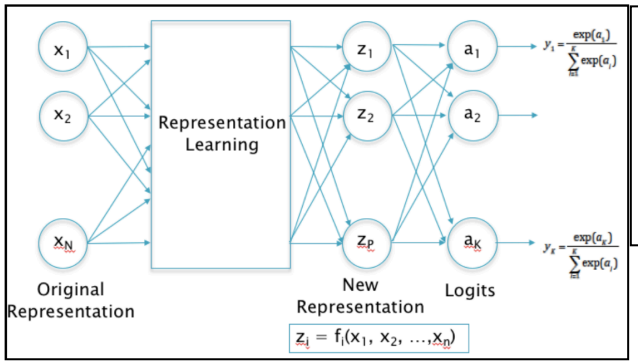
Transfer learning enables zero-shot

- Labeled or unlabeled examples of $\boldsymbol{x}$ allow:
  - Learning a representation function f_x and similarly with examples of $\boldsymbol{y}$ to learn f_y
    - Each application of f_x and f_y appears as an upward arrow
    - Distances in h_x and h_y space provide a similarity metric
    - Image $\boldsymbol{x}_{\text{test}}$ is associated with word $\boldsymbol{y}_{\text{test}}$ even if no image of that word was ever presented

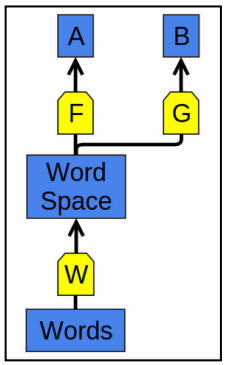


Semi-Supervised Disentangling of Causal Factors

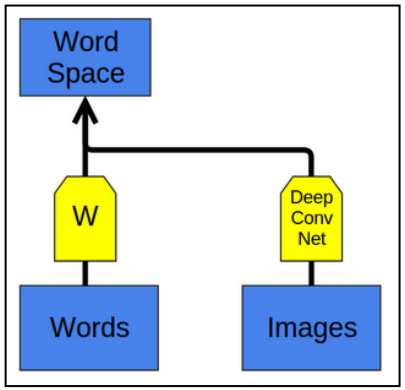
Representations using Deep Learning



Feedforward network learns a representation

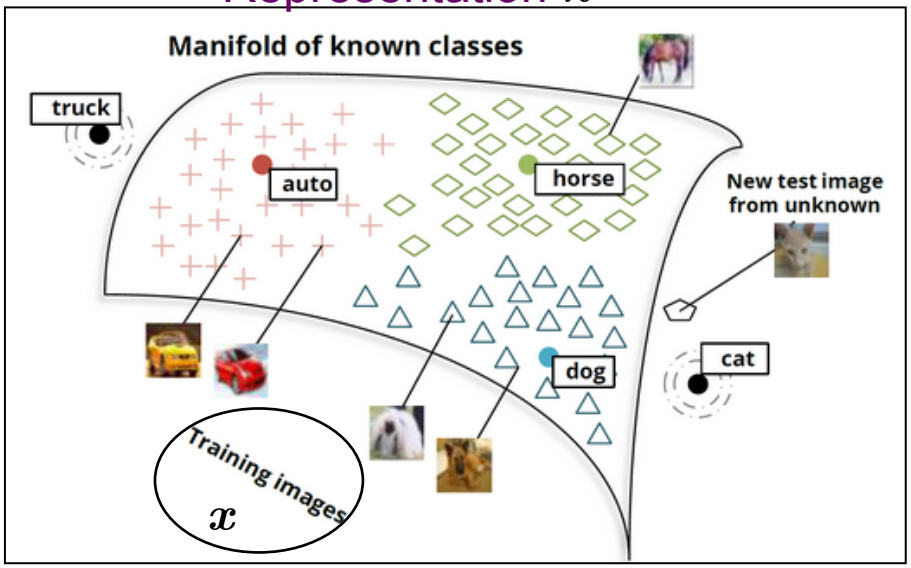


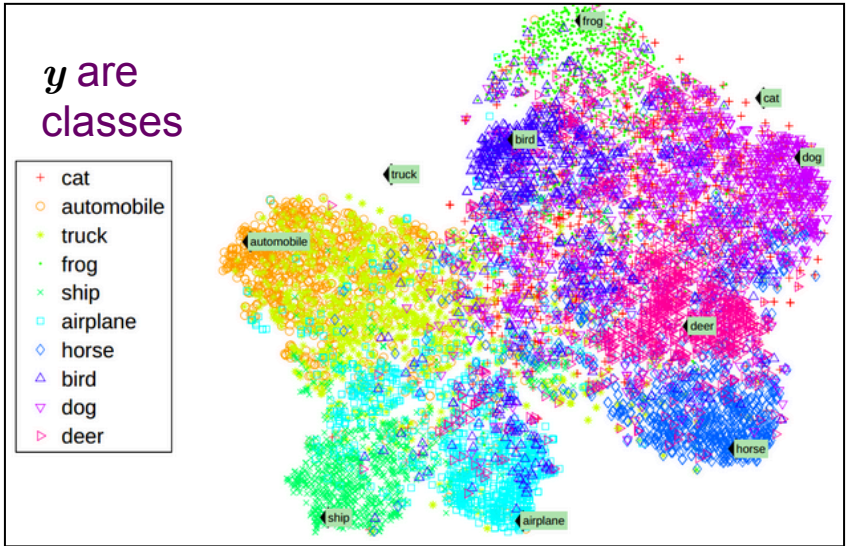
Shared Representation: W and F are used to learn to perform task A Later, G can learn to perform task B based on the representation of W



Embedding words and images in a single representation

Representation h





What makes one representation better than an other?

- Ideal representation $\boldsymbol{h}$ is one where features correspond to the underlying causes of the observed $\boldsymbol{x}$
  - With features h_i correspond to different causes
    - Thus representation disentangles causes from one another
- This motivates approaches in which we seek a good representation for $p(\boldsymbol{x})$
  - Which may also be good for representing $p(\boldsymbol{y}|\boldsymbol{x})$ if $\boldsymbol{y}$ is among the most salient causes of $\boldsymbol{x}$

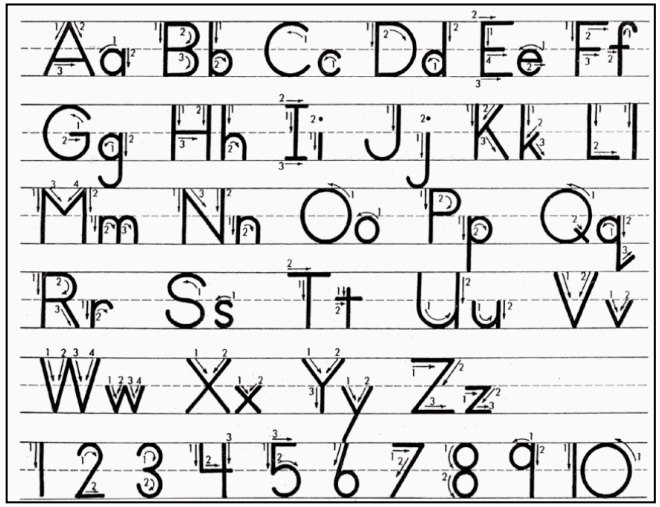
Two goals of representation learning

1. A representation that is easy to model
   – E.g., independence, sparsity
2. Representation that separates causal factors
   – May not be easy to model

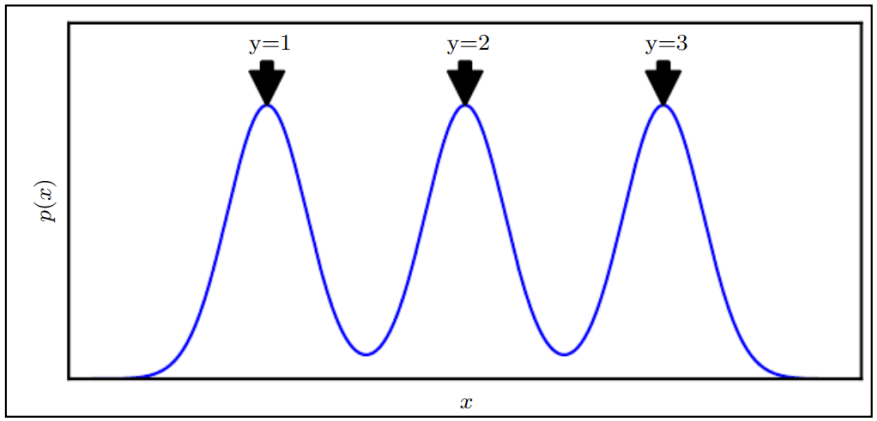
- For many tasks the two coincide
- If a representation $\boldsymbol{h}$ represents many of the underlying causes of the observed $\boldsymbol{x}$, and the outputs $\boldsymbol{y}$ are among the most salient causes, then it is easy to predict $\boldsymbol{y}$ from $\boldsymbol{h}$

How semi-supervised can succeed

- Ex: density over $\boldsymbol{x}$ is a mixture over three components, one per value of $\mathbf{y}$
- If components well-separated:
  - modeling $p(\mathbf{x})$ reveals where each component is
    - A single labeled example per class enough to learn $p(\mathbf{y}|\mathbf{x})$



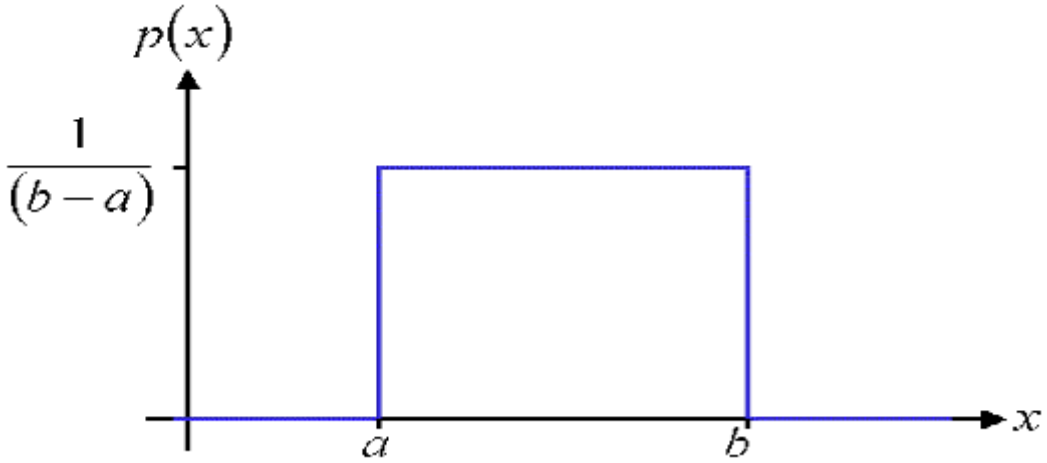
$\boldsymbol{x}$ = no. of black pixels



In this case $p(\mathbf{y}|\mathbf{x})$ is a univariate Gaussian for $\mathbf{y}=1,2,3$

How semi-supervised learning can fail

- When is $p(\mathbf{x})$ if of no help to learning $p(\mathbf{y}|\mathbf{x})$?
- Consider where $p(\mathbf{x})$ is uniformly distributed and we want to learn $f(\boldsymbol{x})=\mathrm{E}[\mathbf{y}|\boldsymbol{x}]$
- Clearly observing the training set of $\boldsymbol{x}$ values alone gives us no information about $p(\mathbf{y}|\mathbf{x})$



Causal factor associated with y

- What could tie $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{x})$ together?
  - If $\mathbf{y}$ is closely associated with one of the causal factors of $\mathbf{x}$, then $p(\mathbf{x})$ and $p(\mathbf{y}|\mathbf{x})$ will be strongly tied
- Unsupervised learning that tries to disentangle the underlying factors of variation is likely to be useful as a semi-supervised learning strategy

Formalizing best possible model

- Assume $\mathbf{y}$ is one of the causal factors of $\mathbf{x}$
- Let $\mathbf{h}$ represent all those factors
- The true generative process can be conceived as structured according to this directed model with $\mathbf{h}$ as the parent of $\mathbf{x}$: $p(\mathbf{h},\mathbf{x})=p(\mathbf{x})p(\mathbf{x}|\mathbf{h})$
  - Thus data has marginal probability $p(\boldsymbol{x})=\mathrm{E}_{\mathbf{h}}\, p(\boldsymbol{x}|\boldsymbol{h})$
- Thus we conclude that the best possible model of $\mathbf{x}$ is one that uncovers the above true structure with $\mathbf{h}$ as a latent variable that explains the observed variations in $\boldsymbol{x}$

Ideal representation learning

- It should recover the latent factors
- If $\mathbf{y}$ is one of these then it will be easy to predict $\mathbf{y}$ from such a representation
- We also see from Bayes rule: $p(\mathbf{y} \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathbf{y})p(\mathbf{y})}{p(\mathbf{x})}$
- Thus the marginal $p(\mathbf{x})$ is intimately tied to the conditional $p(\mathbf{y}|\mathbf{x})$
  - Knowledge of the structure of $p(\mathbf{x})$ should help learn $p(\mathbf{y}|\mathbf{x})$
  - Therefore in situations respecting these assumptions, semi-supervised learning should improve performance

Brute force for large no of causes

- Most observations are formed by an extremely large no of causes
- Suppose $\mathbf{y}=\mathrm{h}_i$, but the unsupervised learner does not know which h_i
- *Brute-force solution*
  - Unsupervised learning: a representation that captures *all* the reasonably salient generative factors h_j
  - Disentagle them from each other thus making it easy to predict $\mathbf{y}$ from $\mathbf{h}$ regardless of which h_i is associated with $\mathbf{y}$

Brute force is infeasible

- It is not possible to capture all factors of variation that influence the observation
  - Ex: in a visual scene, should the representation encode all the smallest objects in the background?
    - Humans fail to perceive changes in environment not relevant to task they are performing
- Research frontier in semi-supervised learning: *What* to encode in each situation

Distributed Representation

Distributed Representation of Concepts

- Distributed representations of concepts are representations composed of many elements that can be set separately from each other
- They are one of the most important tools for representation learning
- They are powerful because they can use n features with k values to describe k^n different concepts

Neural nets use distributed rep.

- Neural networks with multiple hidden units and probabilistic models with multiple latent variables both make use of the strategy of distributed representation
- Many deep models are motivated by hidden units can learn to represent causal factors that explain the data
  - Each direction in representation space can correspond to the value of a different underlying configuration variable

Ex. of Distributed representation

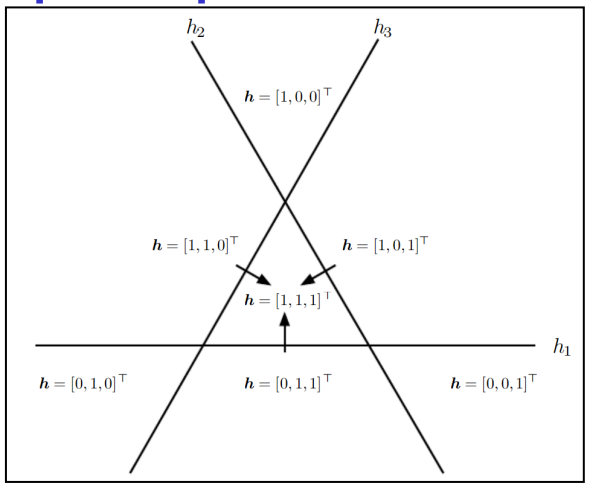
- A vector of n binary features that can take 2^n configurations
- Each potentially corresponding to a different region of input space
- Can be compared to a symbolic representation where the input is associated with a single symbol or category

Distributed/Non-distributed Reps.

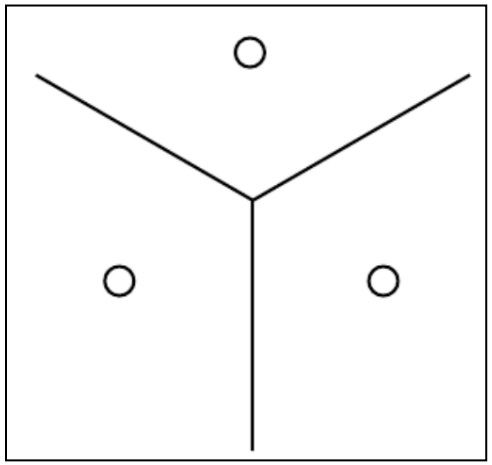
A learning algorithm based on a distributed representation breaks up the input space into regions

Three binary features h_1,h_2 and h_3, each defined by thresholding the output of a learned linear transformation

Each feature divides R^2 into two half planes
Let h_i^+ be the set of inputs for which h_i=1
Each line represents the decision boundary
For one h_i with the corresponding arrow pointing to the h_i^+ side of the boundary



Nearest-neighbor is a non-distributed representation.
There is a separate set of parameters for each region
Advantage: given enough parameters it can fit the training set well without solving a difficult optimization problem. Because it is straightforward to choose a different output independently for each region
Disadvantage: Generalizes only locally via the smoothness prior making it difficult to learn a complicated function with more peaks and troughs than the available no. of examples



Distributed vs Symbolic Represent.

- Important concept that distinguishes a distributed representation from a symbolic one:
  - Generalization arises due to shared attributes between different concepts
  - As pure symbols cat and dog are as far from each other as any two symbols
  - If one associates them with meaningful distributed representation then many things that can be said about cats can generalize to cats
    - Distributed representation may contain entries "$\mathrm{has\_fur}$" or "$\mathrm{number\_of\_legs}$" that have the same value

Statistical Value of Distributed Rep

- Q: When and why can there be a statistical advantage from using a distributed representation as part of a learning algorithm?
- A: When a complicated structure can be compactly represented by a small no. of parameters

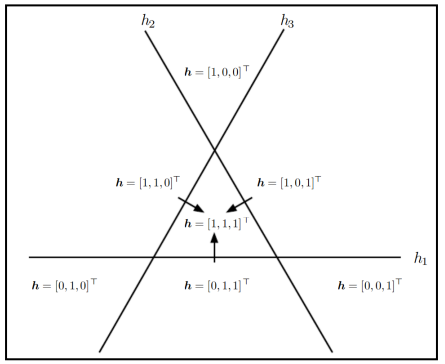
Generalization power of representations

1. Distributed representation
   - Ex: extract n binary features (hidden units) by thresholding linear functions of input
     - Each hidden unit divides input space R^d into a pair of half-spaces
       - With n units no. of regions distinguished:

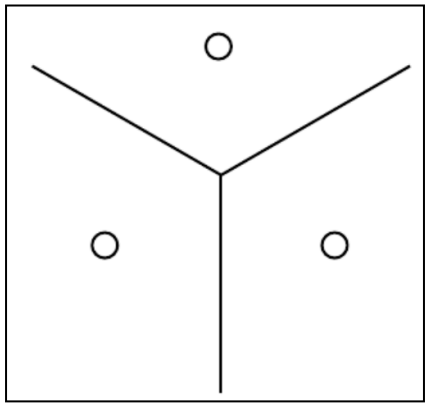
$$\sum_{j=0}^{d} \binom{n}{j} = n^d$$

     - With $O(nd)$ parameters we can represent $O(n^d)$ regions in input space

d=2, n=3



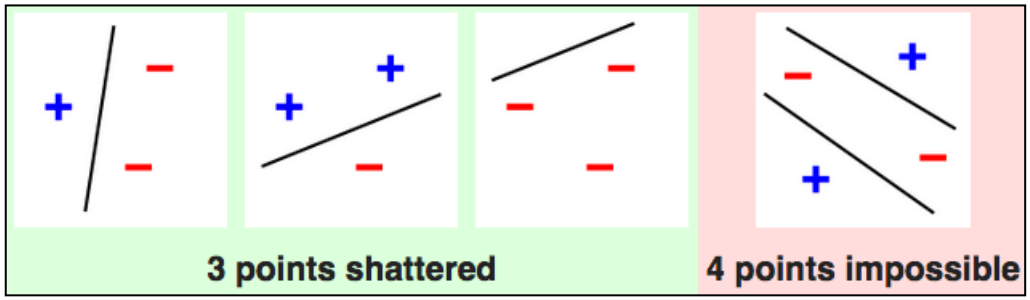
2. If we used a representation with a unique symbol for each region
   - Specifying $O(n^d)$ regions would require $O(n^d)$ examples



- Fewer parameters means fewer samples needed

VC Dimension of Distributed Rep

- VC dimension of a model f: max. no. of points that can be arranged so that f shatters them



  - VC Dimension of Linear Classifier in R^d is: $d+1$
  - VC Dimension of Distributed Rep is: $O(w \log w)$
    - where w is the no. of weights
- Distributed representation can generalize well
  - because capacity remains limited despite being able to encode so many regions

Justifying linear classification

- A distributed representation to learn $\boldsymbol{h}$ combined with a linear classifier to determine $\boldsymbol{y}$
  - Expresses belief that classes are linearly separable of the underlying causal factors captured by $\boldsymbol{h}$
- We typically want to learn concepts such as
  - set of all green objects or set of all cars
- But not categories that require XOR logic
  - set of all red cars and green trucks as one class and


  - set of all green cars and red trucks as another class


Hidden units learn interpretable features

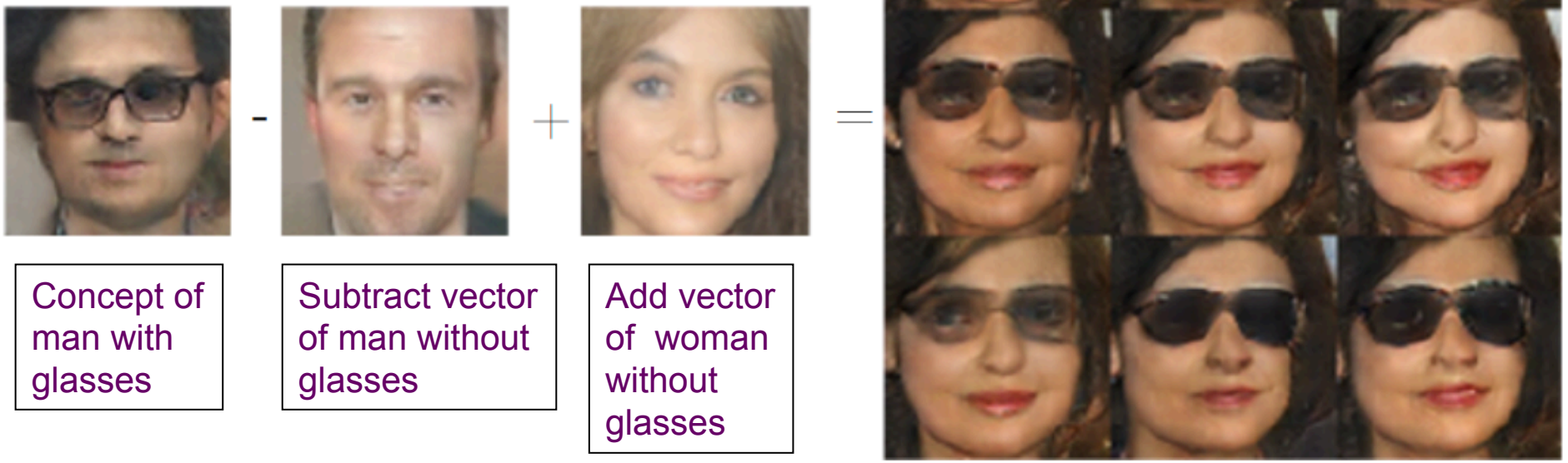
- Not always the case that hidden units learn something that has a linguistic name
- But they emerge near the top levels of deep networks
- They tend to be concepts we could learn about each without having to see configurations of all

Disentangling Concepts

- Generative model learns a distributed representation
- Images of faces with separate directions in representation space capturing different underlying factors of variation
  - Disentangles concept of gender from concept of glasses

Disentangling gender and glasses

- One direction in representation space is gender, another is whether wearing glasses
- Features discovered automatically and not discovered a priori
- No need to have labels for hidden units



Concept of man with glasses

Subtract vector of man without glasses

Add vector of woman without glasses

Concept of woman with glasses

Greedy Layer-wise Unsupervised Pretraining

Pre-training and fine tuning

- Using dataset A train model m
- Pre-training:
  - You have a dataset B
  - Before training the model, initialize some of the parameters of m with model trained on A
- Fine-tuning:
  - You train m on B
- This is one form of transfer learning

Unsupervised Learning

- Unsupervised learning played key historical role in revival of deep neural networks
  - Enabling training a deep supervised network without requiring architectural specializations such as convolution or recurrence
- We call this procedure *unsupervised pretraining*
- Or more precisely *greedy layer-wise unsupervised pretraining*

Greedy Layer-wise Unsupervised Pretraining

- A representation learned for one task
  - unsupervised learning, that captures the shape of the input distribution
- Is used for another task
  - supervised learning with the same input domain
- Greedy layer-wise pre-training relies on a single-layer representation learning

Single layer Pretraining

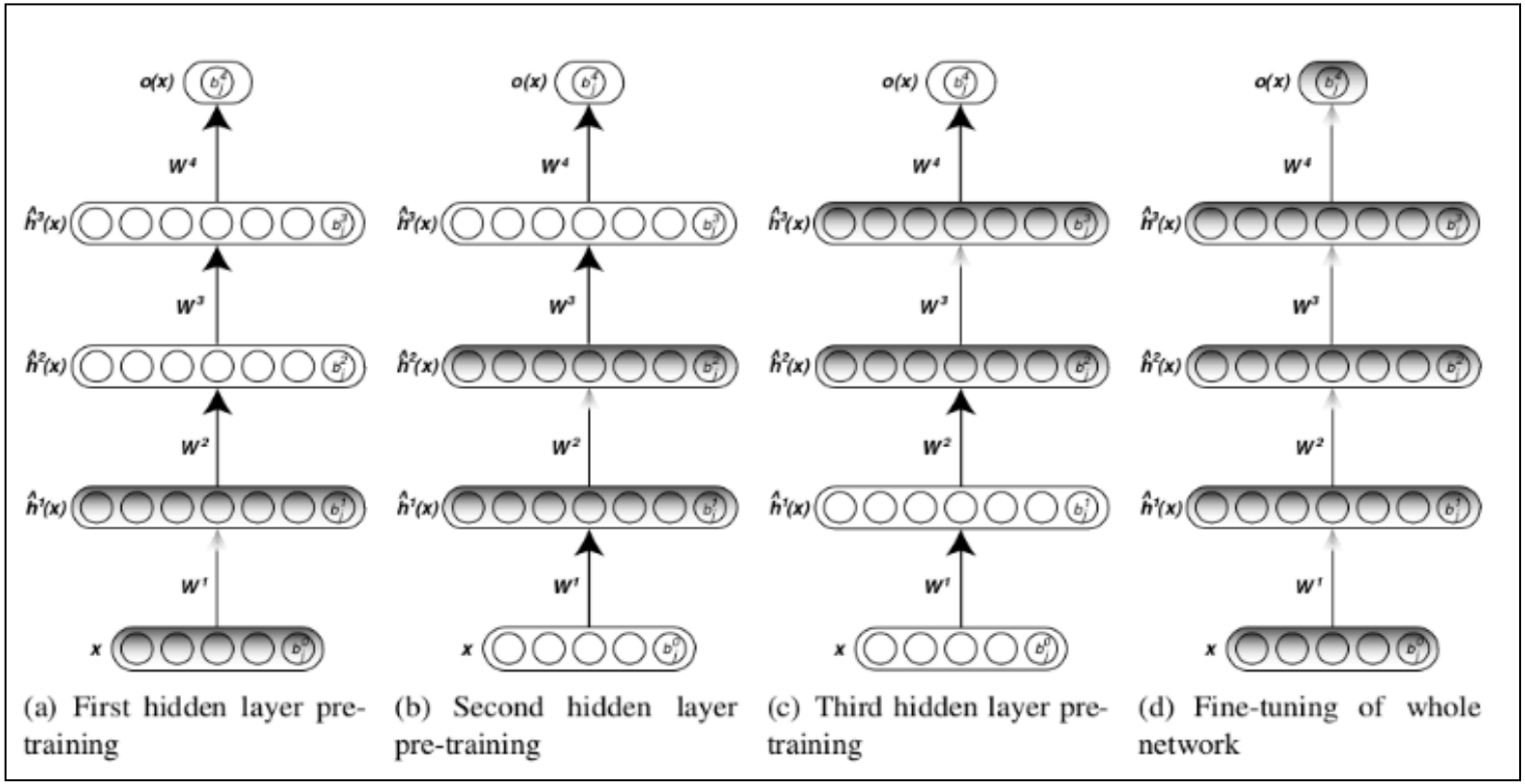
- Each layer pretrained using unsupervised learning
  - Taking the output of the previous layer and producing as output a new representation of data,
    - Whose distribution (or relation to categories) is simpler

Single-layer representation learning

- We need a single-layer representation learning algorithm, such as:
  - An *RBM* (a Markov network with restrictions)
  - A single-layer *autoencoder*
  - A sparse coding model
  - Or another model that learns latent representations

Training a 4-layer network

- Pairs of layers active in each stage



(a) First hidden layer pre-training (b) Second hidden layer pre-training (c) Third hidden layer pre-training (d) Fine-tuning of whole network

Formal Algorithm

Algorithm: Greedy Layer-wise Unsupervised Pretraining Protocol

- Given unsupervised feature learning algorithm $\mathcal{L}$
  - Which takes as input a training set of examples and returns an encoder or feature function f.
- Raw input data is $\boldsymbol{X}$, with one row per example, $f^{(1)}(\boldsymbol{X})$ is output of the first stage encoder on $\boldsymbol{X}$.
- In the case where fine tuning is performed we use
  a learner $\mathcal{T}$ which takes an initial function f, input examples $\boldsymbol{X}$ (and in the supervised fine-tuning case, associated targets $\boldsymbol{Y}$) and returns a tuned function.
  The no of stages is m.

$f \leftarrow$ Identity function
$\tilde{\boldsymbol{X}} = \boldsymbol{X}$
for $k = 1, \ldots, m$ **do**
  $f^{(k)} = \mathcal{L}(\tilde{\boldsymbol{X}})$
  $f \leftarrow f^{(k)} \circ f$
  $\tilde{\boldsymbol{X}} \leftarrow f^{(k)}(\tilde{\boldsymbol{X}})$
end for
if *fine-tuning* ***then***
  $f \leftarrow \mathcal{T}(f, \boldsymbol{X}, \boldsymbol{Y})$
end if
Return f

History: layer-wise unsupervised

- Unsupervised greedy layer-wise training
  - was used to sidestep difficulty of training layers of a deep neural net for a supervised task
  - Origins in Neocognitron (Fukushima, 1975)
  - Deep learning renaissance of 2006 began with
    - Greedy learning to find initialization for all layers
      - Useful for fully connected architectures
    - Earlier, only deep CNNs or depth resulting from recurrence were feasible to train
- Today greedy layer-wise pretraining is not required to train fully connected deep networks

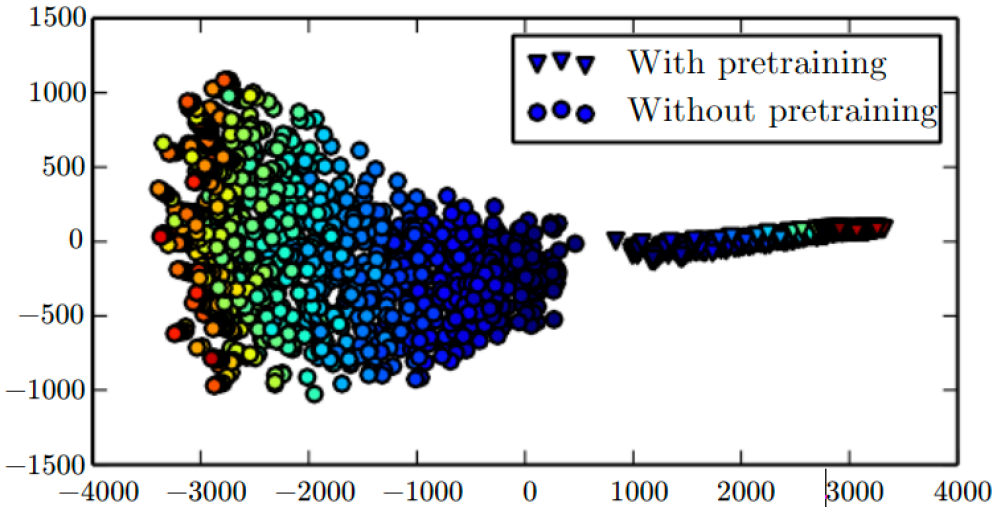
Greedy pretraining terminology

- Greedy layer-wise pretraining
  - Greedy because
    - It is a greedy algorithm that optimizes each piece of the solution independently
      - One piece at a time rather than jointly
  - Layer-wise because
    - Independent pieces are the layers of the network
    - Training proceeds one layer at a time
      - Training the k^{th} layer while previous ones are fixed
  - Pretraining because
    - It is only a first step before applying a joint training algorithm is applied to *fine-tune* all layers together

When/why does pretraining work?

- Greedy layer-wise unsupervised pretraining can yield substantial improvements for classification
  - However it is sometimes *harmful*
- Pretraining accesses new part of space:
  - With pretraining: halt in one region of function space
  - Without pretraining: another region

Visualization of functions projected into $2\mathrm{d}$ space.
(Each function is an infinite-dimensional vector that associates every input x with output y).
Color indicates time.
Area where pretrained networks arrive is smaller.



Exponential Gains from Depth

Efficiency of Mutilayer Perceptrons

- Multilayer perceptrons have these properties:
  1. They are universal approximators
     - i.e., can approximate most functions given enough hidden units upto any non-zero tolerance
  2. Functions are represented by smaller deep networks compared to shallow networks
     - No of linear regions carved out with d inputs, depth l and n units per hidden layer is exponential in l $O\left(\binom{n}{d}^{d(l-1)} n^d\right)$
     - Decrease in model size leads to statistical efficiency
- This section looks at how similar results apply to other distributed hidden representations

Expressive power of deep networks

- There are families of functions that can be:
  1. Efficiently represented by architecture of depth k
  2. But would require an exponential number of hidden units (wrt input size) with insufficient depth (depth 2 or depth k-1)

Minimum depth required

- Theoretical result based on using a Sum-Product Network (SPN)
  - Which use polynomial circuits
  - Models compute a probability distribution over a set of random variables
- Showed that there exist probability distributions for which a minimum depth of SPN is required to avoid needing an exponentially large model
  - Later showed that there are significant differences between every two finite depths of SPN
  - Some constraints used to make SPNs tractable may limit their representational power

Expressive power of CNNs

- Theoretical result based on families of deep circuits related to convolutional networks
- There is an exponential advantage for the deep circuit even when the shallow circuit is allowed to only approximate the function computed by the deep circuit

Providing Clues to Discover Underlying Causes

What is an ideal representation?

- What makes one representation better than another?
- One answer: ideal representation is one that disentangles the underlying *causal factors of variation* that generated the data, especially those that are relevant to our application
- Most strategies for representation learning are based on introducing clues that help the learning to find these underlying factors of variation

Clues to find factors of variation

- Supervised learning provides a strong clue
  - Label $\boldsymbol{y}$ presented with each $\boldsymbol{x}$ specifies one factor of variation directly
- To make use of unlabeled data, representation learning makes use of other less direct hints about underlying factors
  - Imposed by designers of learning algorithm to guide the learner
- Goal of deep learning is to find regularization strategies applicable to many AI tasks that people solve

Generic Regularization Strategies

- To encourage learning algorithms to discover features that correspond to factors of variation

1. Smoothness
2. Linearity
3. Multiple explanatory factors
4. Causal factors
5. Depth (hierarchical organization of explanatory factors)
6. Shared factors across tasks
7. Manifolds
8. Natural Clustering
9. Temporal/Spatial Coherence
10. Sparsity
11. Simplicity of factor dependencies

1. Smoothness

- This is the assumption that $f(\boldsymbol{x}+\varepsilon\boldsymbol{d})\approx f(\boldsymbol{x})$ for unit $\boldsymbol{d}$ and small ε
- Allows generalizing from training points to nearby points
- Many ML algorithms use this idea
- But it is insufficient to overcome curse of dimensionality

2. Linearity

- Assume that relationships between variables are linear
- Allows making predictions far from observed data
  - But sometimes leads to extreme predictions

3. Multiple Explanatory Factors

- Motivated by assumption that data is generated by multiple underlying explanatory factors
- Most tasks can be solved easily given the state of each of these factors
- Learning $p(\boldsymbol{x})$ requires some of the same features that are useful for modeling $p(\boldsymbol{y}|\boldsymbol{x})$ because both refer to the same underlying explanatory factors
  - This view motivates the use of distributed representations, with separate directions in representation space corresponding to separate factors of variation

4. Causal Factors

- The model is constructed in such a way that it treats the factors of variation described by the learned representation $\boldsymbol{h}$ as the causes of the observed data $\boldsymbol{x}$ and not vice versa
- This is advantageous for semi-supervised learning and the learned model more robust when the distribution over the underlying causes changes or when we use the model for a new task

5. Depth

- By using a hierarchical organization of explanatory factors, high-level abstract concepts can be defined in terms of simple concepts
- Another viewpoint: task should be accomplished via a multi-step program
  - With each step referring back to the output of the processing accomplished via previous steps

6. Shared factors across new tasks

- When we have many tasks corresponding to:
  - Different y_i variables sharing the same input $\mathbf{x}$, or
  - When each task is associated with a subset or a function $f^{(i)}(\mathbf{x})$ of a global input $\mathbf{x}$
- The assumption is that each each y_i is associated with a different subset from a common pool of relevant factors $\mathbf{h}$
- Because these subsets overlap, learning all the $P(\mathrm{y}_i|\mathbf{x})$ via a shared intermediate representation $P(\mathbf{h}|\mathbf{x})$ allows sharing of statistical strength between the tasks

7. Manifolds

- Probability mass concentrates
  - In locally connected regions
  - Occupy a tiny volume
- In continuous cases, regions characterized by low-dimensional manifolds
- Many ML algorithms behave sensibly only on this volume
- Many ML algorithms, especially autoencoders attempt to explicitly learn the structure of manifolds

8. Natural Clustering

- Many ML algorithms assume that each connected manifold in the input space may be assigned to a single class
  - But class data may lie on disconnected manifolds
- This motivates many learning algorithms
  - Tangent propagation, double backprop, manifold tangent classifier, adversarial training

9. Temporal and Spatial Coherence

- Most important explanatory factors are assumed to change slowly with time
- It is easier to predict the true underlying explanatory factors than to predict raw observations such as pixel values

10. Sparsity

- Most features should presumably not be relevant to describing most inputs
  - There is no need to use a feature that detects elephant trunks when representing an image of a cat
- It is therefore reasonable to impose a prior that any feature that can be interpreted as “present” or “absent” should absent most of the time

11. Simplicity of factor dependencies

- In good high-level representations, the factors are related to each other through simple dependencies
- Simplest possible is marginal independence $P(\boldsymbol{h}) = \Pi_i P(\mathrm{h}_i)$
- But linear dependencies are also reasonable assumptions
  - Many laws of physics
  - Assumed when plugging a linear predictor of factorized prior

Summary of representation learning

- Representation learning ties together all the many forms of deep learning
- Feedforward/recurrent networks, autoencoders and deep PGMs all learn and exploit representations
- Learning the best possible representation remains an exciting avenue of research

Summary of Representation Learning

- Representations should have four qualities:
  1. They should be as informative as the data
  2. They should be invariant, or unaffected by nuisance factors, or “noise”
  3. They should be disentangled
  4. Be as simple as possible and easy to work with
     - It should find the right tradeoff between accuracy and complexity.