EE6550 Machine Learning

Lecture Thirteen – Deep Network II
Convolutional Networks, Recurrent and Recursive Networks, and Related Topics

Chung-Chin Lu

Department of Electrical Engineering

National Tsing Hua University

June 5, 2017

Convolutional Neural Networks

Convolution

- Continuous-parameter convolution:

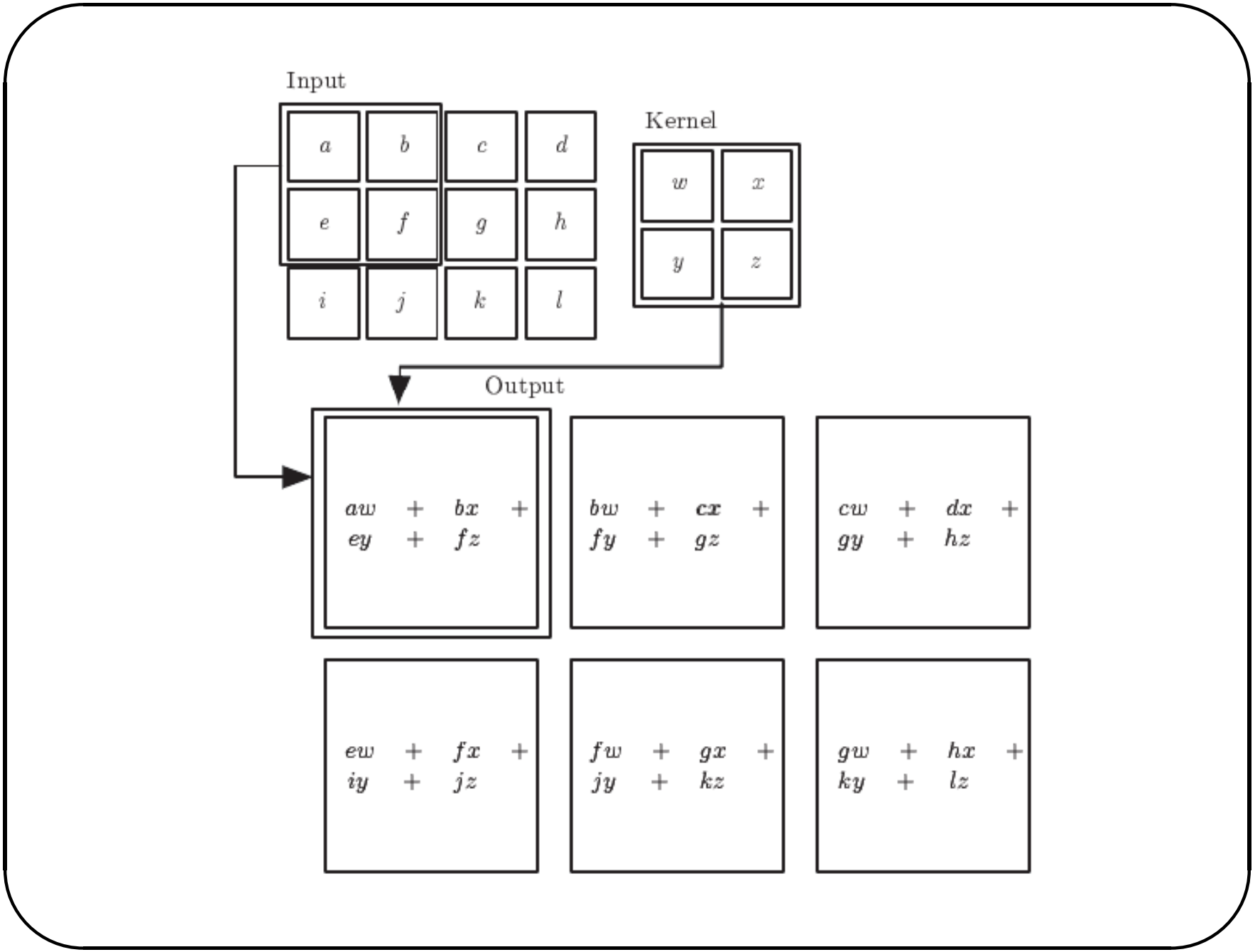
$$\begin{aligned} & s(t_1, t_2, \ldots, t_n) \\ = & \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} d\tau_1 d\tau_2 \cdots d\tau_n w(\tau_1, \tau_2, \ldots, \tau_n) \\ & \quad x(t_1 - \tau_1, t_2 - \tau_2, \ldots, t_n - \tau_n) \\ = & (x * w)(t_1, t_2, \ldots, t_n). \end{aligned}$$

  - $x(t_1, t_2, \ldots, t_n)$: the input.
  - $w(t_1, t_2, \ldots, t_n)$: the kernel.
  - $s(t_1, t_2, \ldots, t_n)$: the feature.

- Discrete-parameter convolution:

$$\begin{aligned} s(i_1, i_2, \ldots, i_n) &= \sum_{j_1=-\infty}^{\infty} \sum_{j_2=-\infty}^{\infty} \cdots \sum_{j_n=-\infty}^{\infty} w(j_1, j_2, \ldots, j_n) \\ &\qquad x(i_1 - j_1, i_2 - j_2, \ldots, i_n - j_n) \\ &= (x * w)(i_1, i_2, \ldots, i_n) \\ &= \sum_{j_1=-\infty}^{\infty} \sum_{j_2=-\infty}^{\infty} \cdots \sum_{j_n=-\infty}^{\infty} w(-j_1, -j_2, \ldots, -j_n) \\ &\qquad x(i_1 + j_1, i_2 + j_2, \ldots, i_n + j_n). \end{aligned}$$

- $w(j_1, j_2, \ldots, j_n) = 0$ for all but finitely many array points $(j_1, j_2, \ldots, j_n)$: a finite-impulse-response (FIR) kernel.
- An example: only $w(0, 0), w(0, -1), w(-1, 0), w(-1, -1)$ nonzero.


Input
a
b
c
d
e
f
g
h
i
j
k
l
Kernel
w
x
y
z
Output
aw + bx +
ey + fz
bw + cx +
fy + gz
cw + dx +
gy + hz
ew + fx +
iy + jz
fw + gx +
jy + kz
gw + hx +
ky + lz

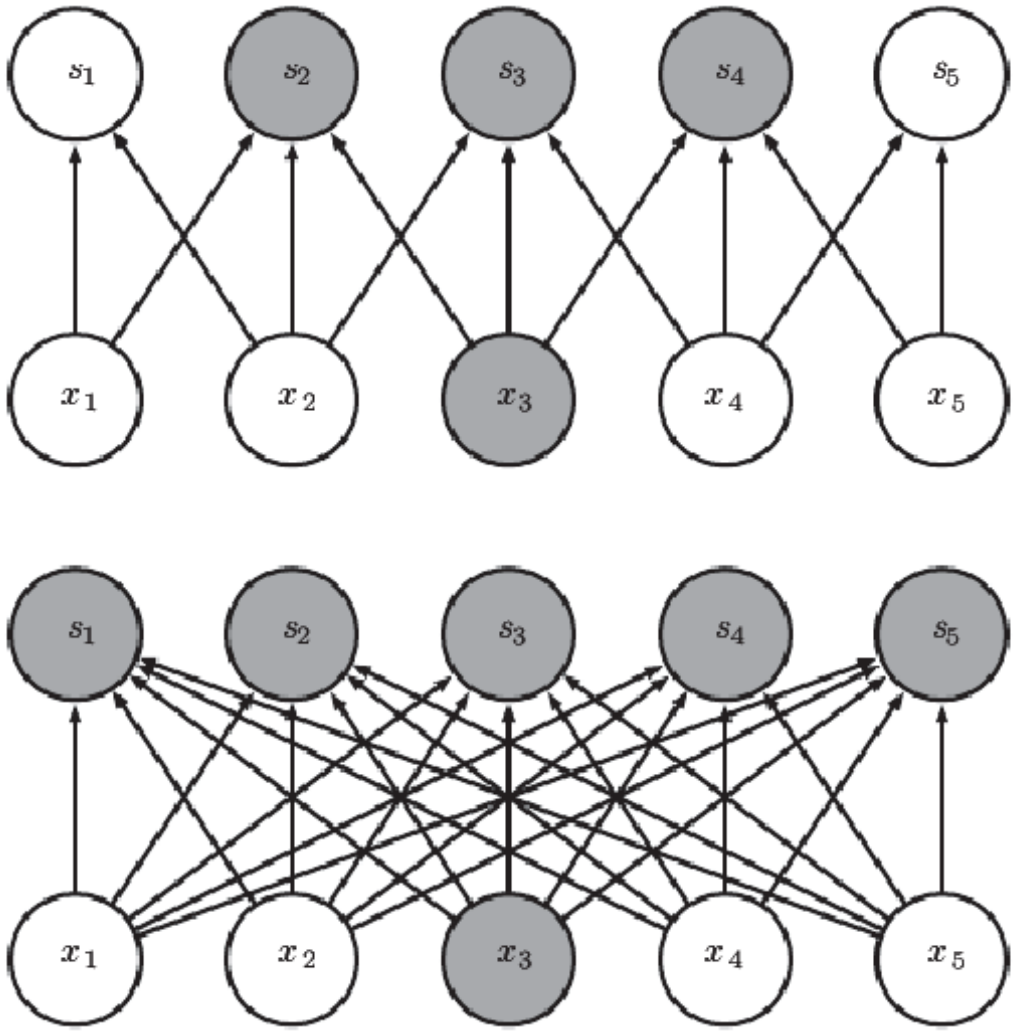
Sparse Interaction in Convolution

- Sparse interaction (sparse connectivity or sparse weights):

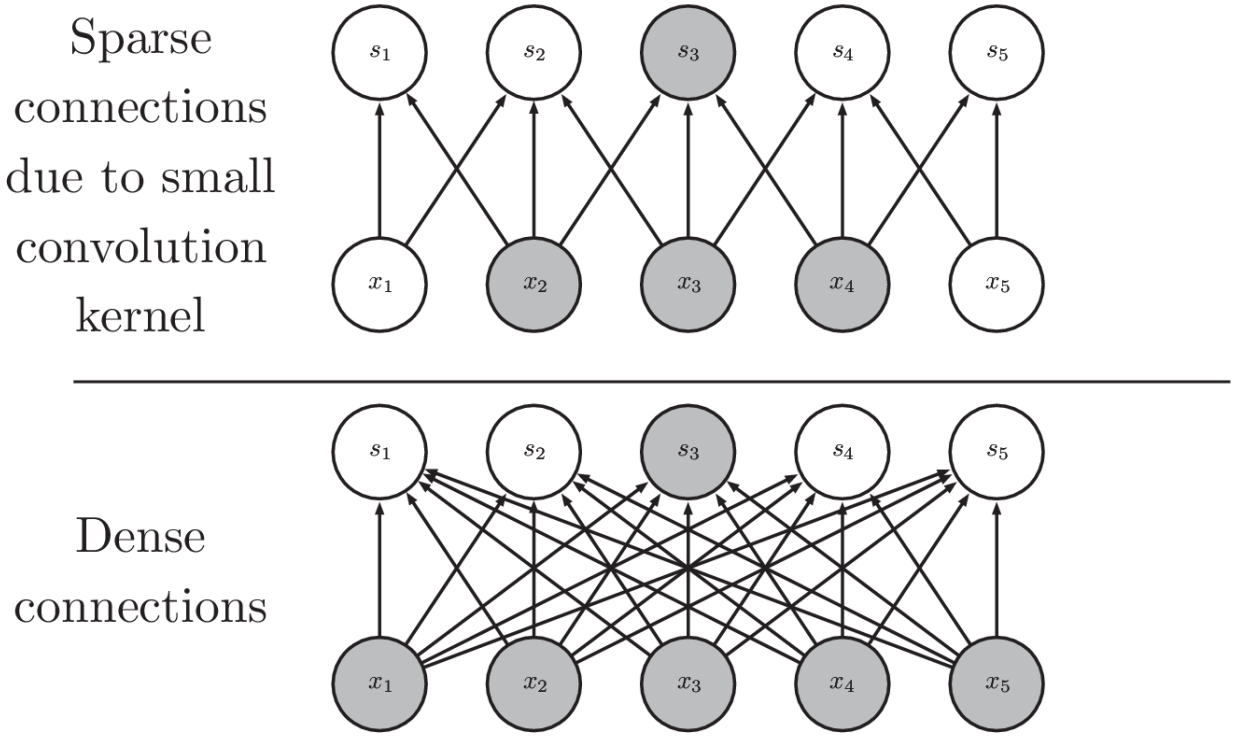
$$a_{t+1,i} = \sum_{j'=-l_t}^{m_t-1} o_{t,i-j'} w_{t,j'} = \sum_{j=1}^{k_t} o_{t,j} W_{t,i,j},$$

where $W_{t,i,j} = w_{t,i-j}$. In matrix form, we have

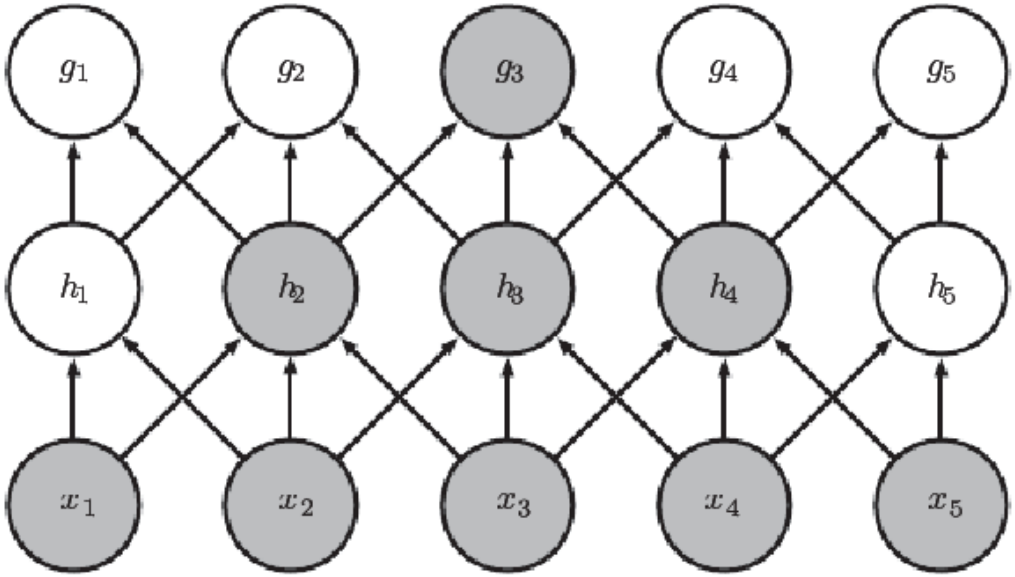
$$\begin{bmatrix} a_{t+1,1} \\ a_{t+1,2} \\ \vdots \\ a_{t+1,k_{t+1}} \end{bmatrix} = \begin{bmatrix} w_{t,0} & \cdots & w_{t,-l_t} & 0 & \cdots & 0 \\ w_{t,1} & \cdots & w_{t,-l_t+1} & w_{t,-l_t} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & w_{t,m_t-1} & \cdots & w_{t,k_{t+1}-k_t} \end{bmatrix} \begin{bmatrix} o_{t,1} \\ o_{t,2} \\ \vdots \\ o_{t,k_t} \end{bmatrix}$$



Sparse connectivity, viewed from below.



Sparse connectivity, viewed from above. The input units x_2, x_3, x_4 that affect the output unit s_3 consists of the receptive field of s_3.

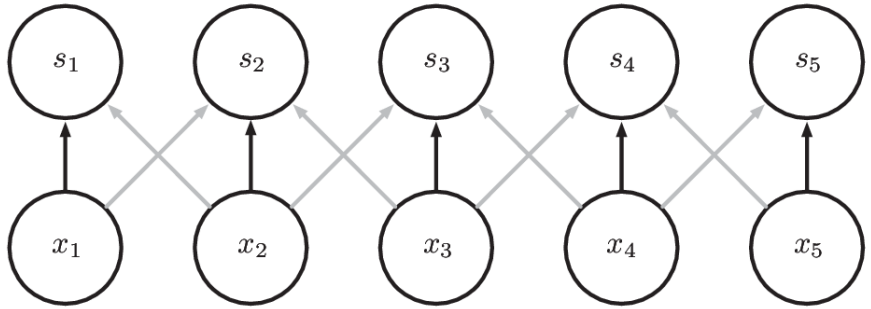


The receptive fields of an output unit in the deeper layers of a convolutional network is larger than the receptive fields in the shallow layers.

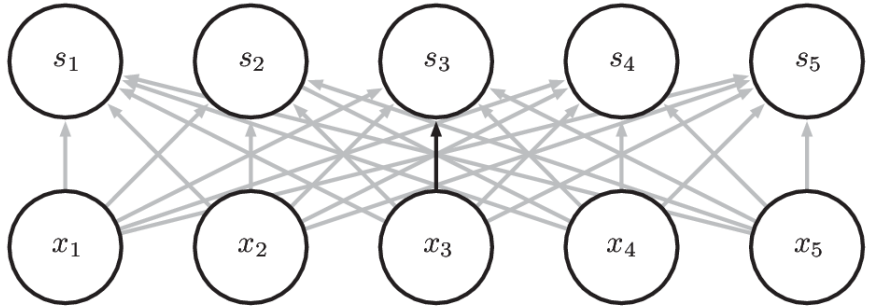
Parameter Sharing in Convolution

- Parameter sharing (i.e., having tied weights): $W_{t,i,j} = w_{t,i-j}$.

Convolution shares the same parameters across all spatial locations



Traditional matrix multiplication does not share any parameters



Equivariance to Translation in Convolution

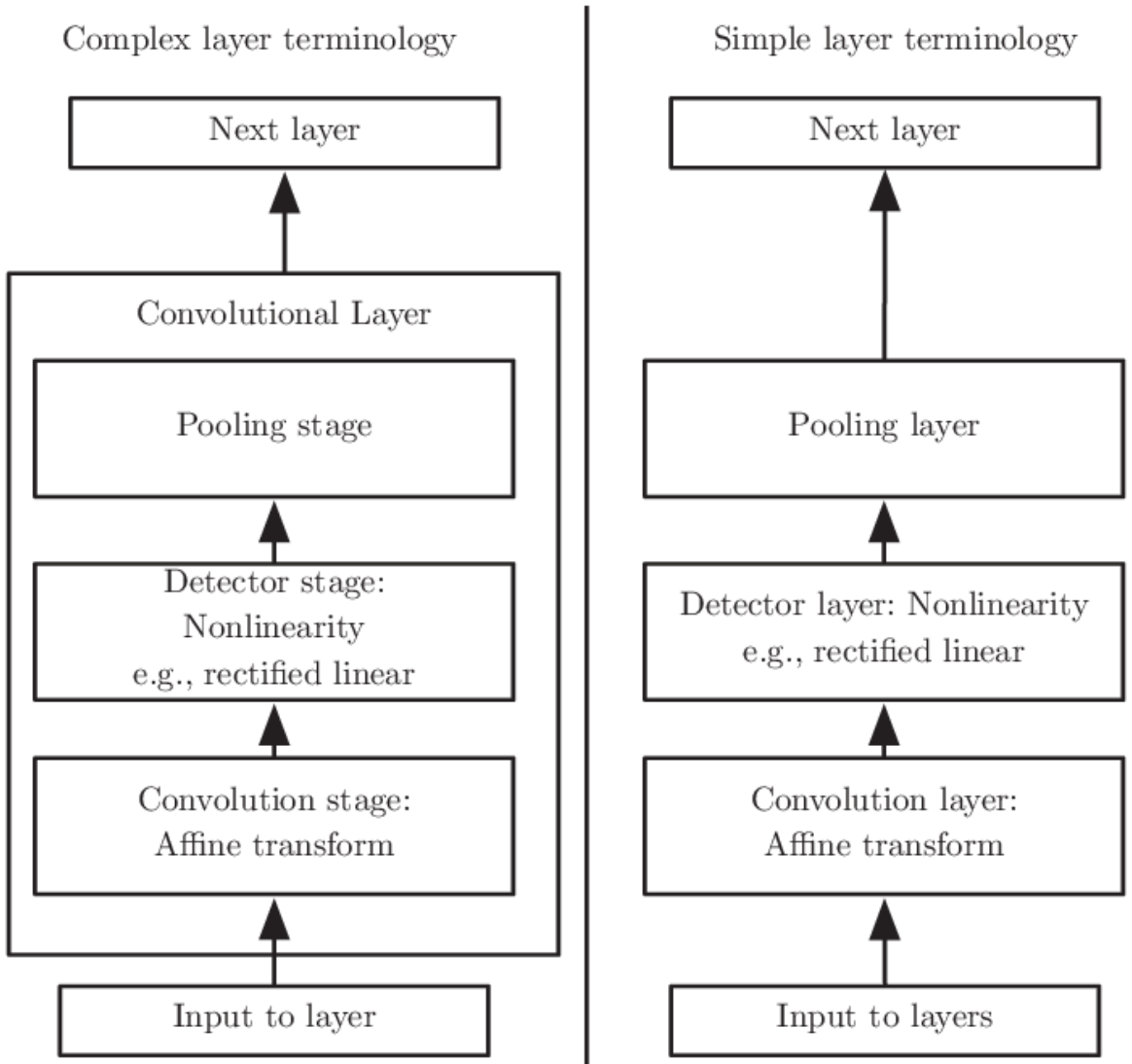
- Equivariance to translation (shift-invariance): if $\{x(i)\}$ and $\{s(i)\}$ are an input and output pair, then $\{x(i-\tau)\}$ and $\{s(i-\tau)\}$ is also an input and output pair for any τ. In convolutional neural network, we have

$$a_{t+1,i-\tau} = \sum_{j'=-l_t}^{m_t-1} o_{t,i-\tau-j'} w_{t,j'}.$$

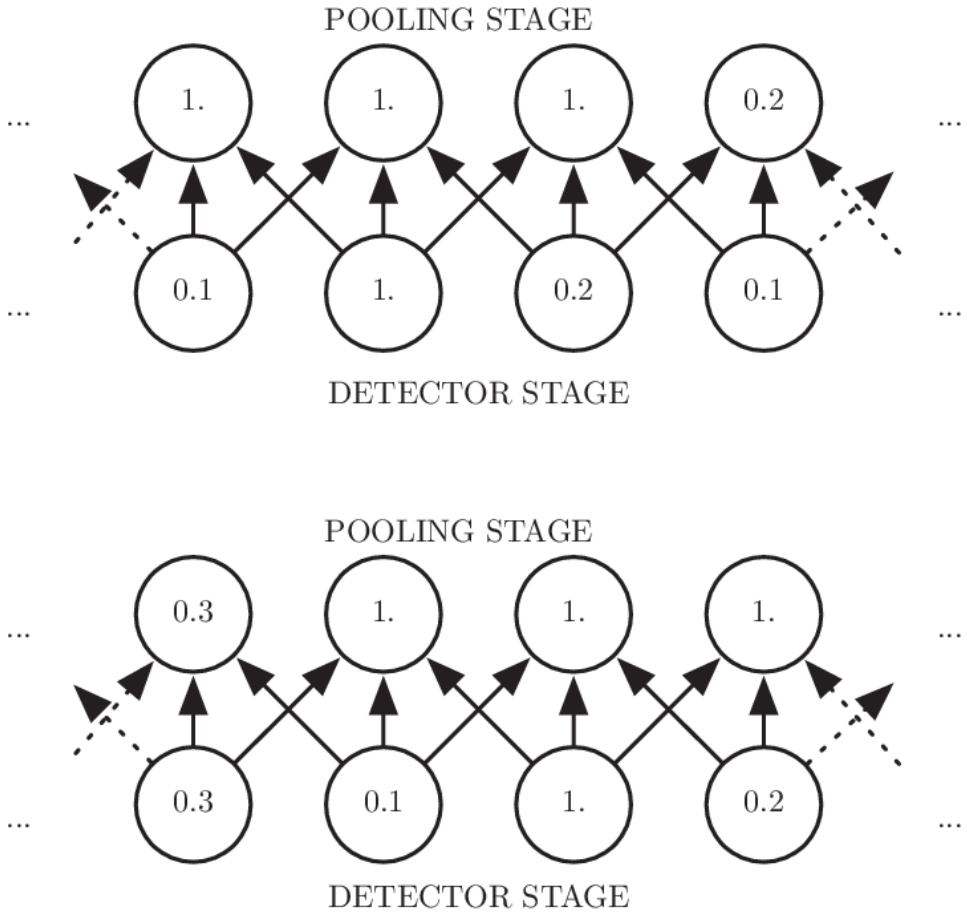
  - If we move the input object x in the time coordinate or in the space coordinates, its output representation s will move the same amount in the time coordinate or in the space coordinates.
  - Equivariance to translation is a consequence of parameter sharing.

Three Stages in a Typical Layer of a CNN

- Convolutional stage: $a_{t+1,i} = \sum_{j'=-l_t}^{m_t-1} o_{t,i-j'} w_{t,j'}$.
- Detector stage: $o'_{t+1,i} = \sigma(a_{t+1,i})$ by applying the activation function σ to the input $a_{t+1,i}$ at neuron $v_{t+1,i}$.
- pooling stage: applying a pooling function to the detected values $\{o'_{t+1,i}\}_{i=1}^{k_{t+1}}$ in the $(t+1)$th CNN layer such that at a certain neuron the pooling function gives a summary statistic of the detected values at the nearby neurons.
  - The max pooling: $o_{t+1,i} = \max_{i-r \le i' \le i+r} o'_{t+1,i'}$.
  - Other popular pooling functions include the average of a rectangular neighborhood, the L_2 norm of a rectangular neighborhood, or a weighted average based on the distance from the central neuron.

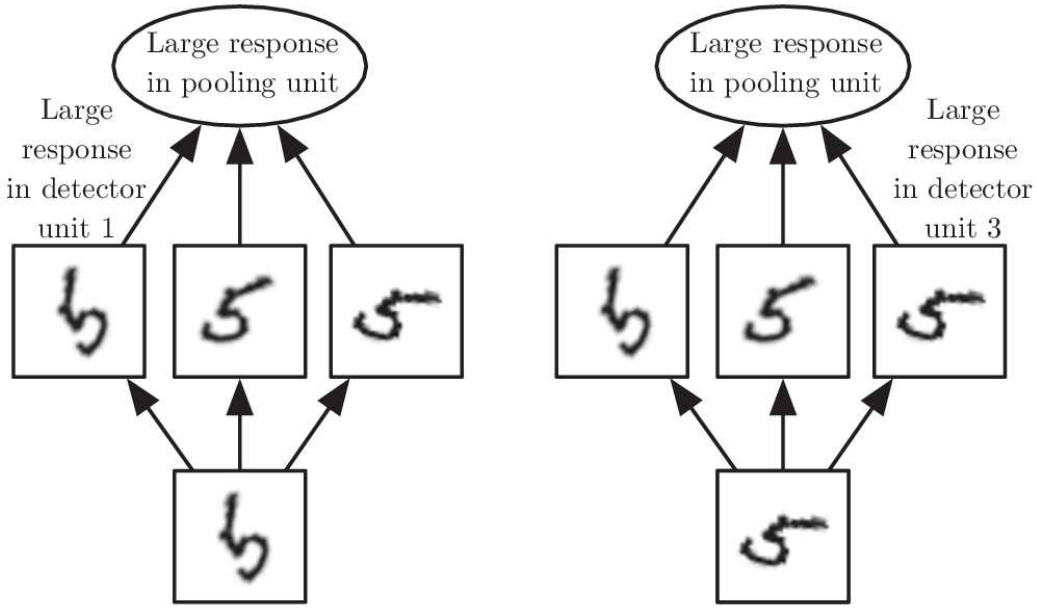

Complex layer terminology
Next layer
Convolutional Layer
Pooling stage
Detector stage:
Nonlinearity
e.g., rectified linear
Convolution stage:
Affine transform
Input to layer
Simple layer terminology
Next layer
Pooling layer
Detector layer: Nonlinearity
e.g., rectified linear
Convolution layer:
Affine transform
Input to layers

Pooling: Invariance to Local Translation



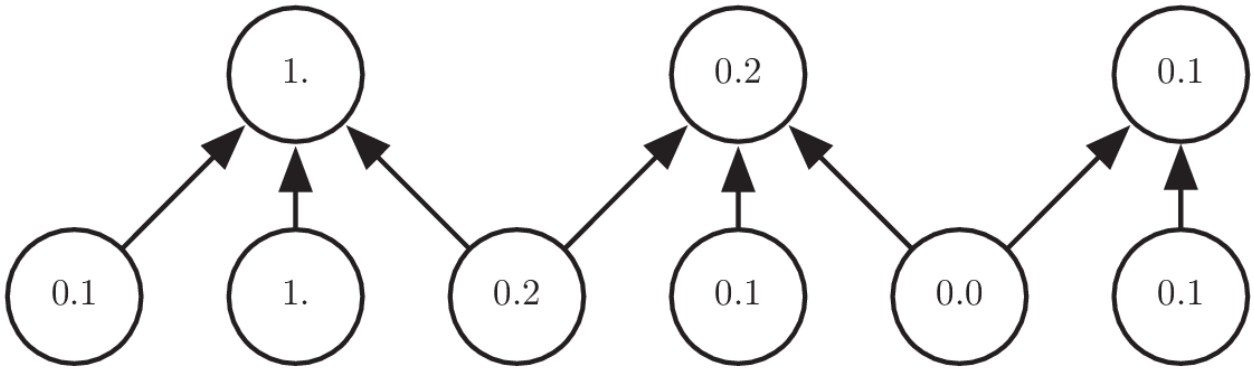
Max pooling introduces invariance.

Multi-channel Pooling: Invariance to Rotation



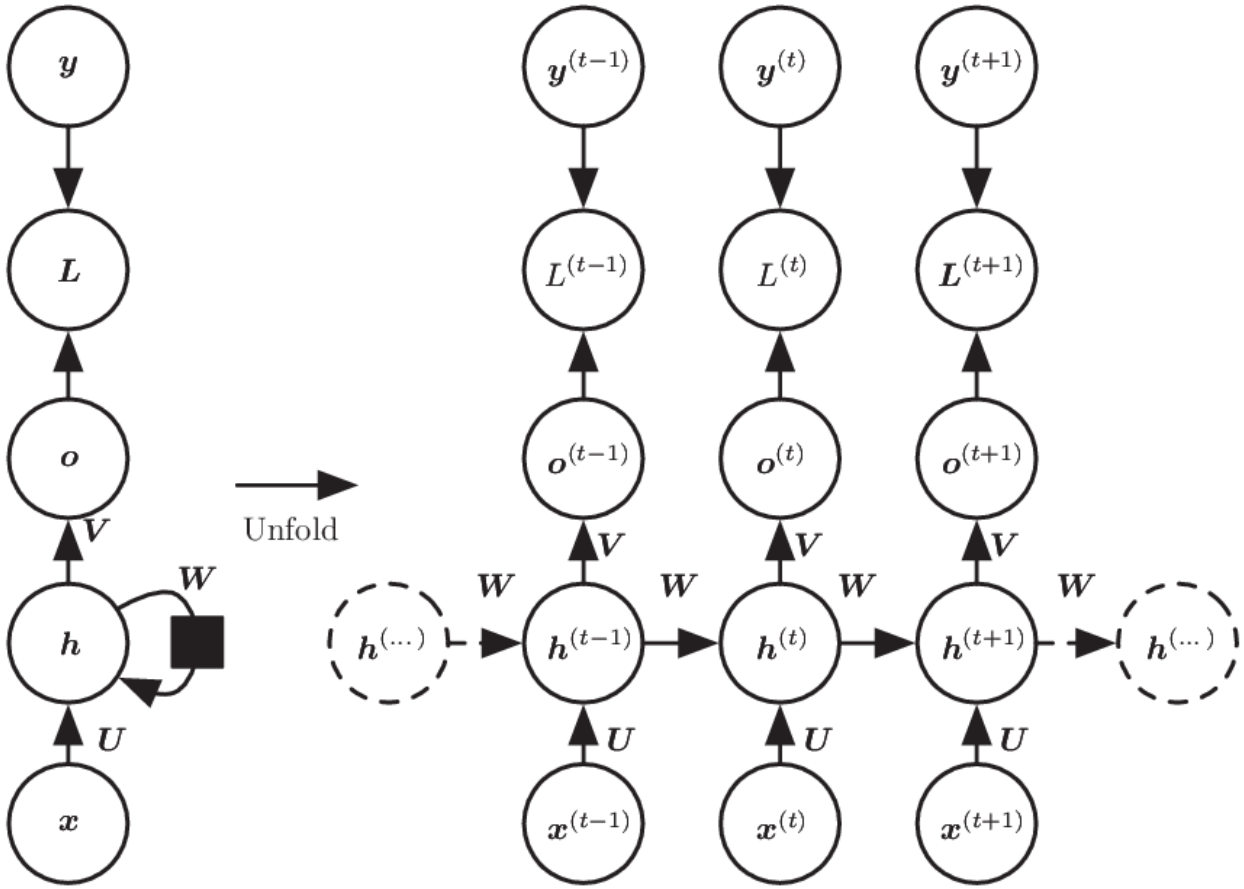
A set of three learned filters and a max pooling unit can learn to become invariant to rotation.

Pooling: Downsampling



Max-pooling with a pool width of three and a stride between pools of two.

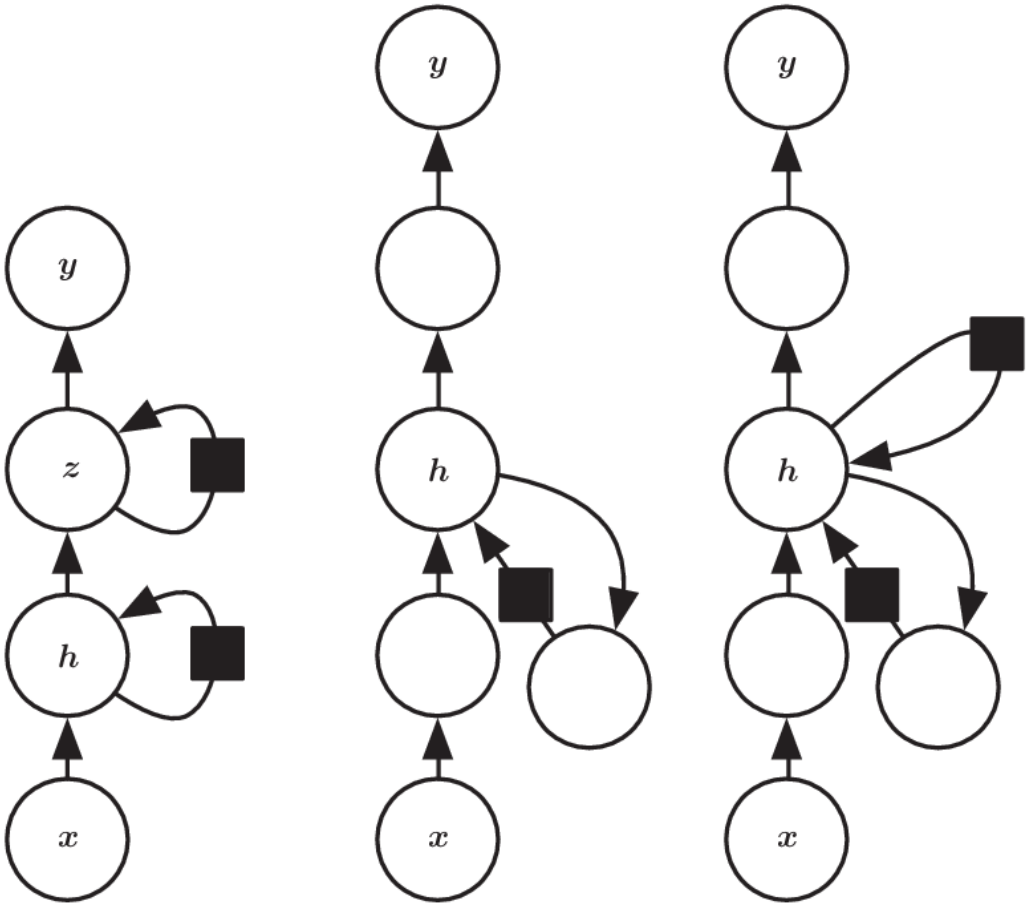
Recurrent Neural Networks



$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \boldsymbol{\sigma}(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \boldsymbol{\sigma}'(\boldsymbol{o}^{(t)}).
\end{aligned}
$$

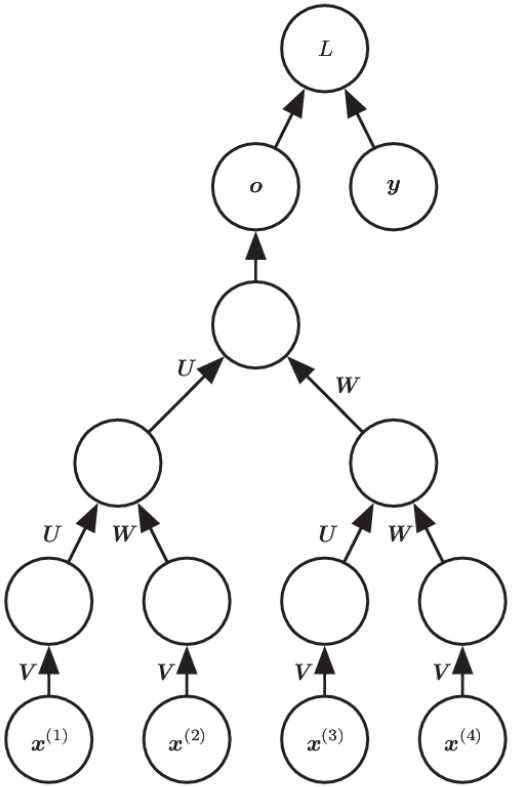
- This is just a finite-state machine.
- This recurrent neural network is universal in the sense that any function computable by a Turing machine can be computed by such a recurrent network of a finite size.

Deep Recurrent Neural Networks



A recurrent neural network can be made deep in many ways

Recursive Neural Networks



A recursive network has a computational graph that generalizes that of the recurrent network from a chain to a tree.

- A variable-size sequence $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$ can be mapped to a fixed-size representation (the output $\boldsymbol{o}$), with a fixed set of parameters (the weight matrices $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$).
- Recursive networks have been successfully applied to processing data structures as input to neural networks, in natural language processing as well as in computer vision.

A Detour to Clustering Algorithms

Clustering Problem

- $\mathscr{I}$: the input space of all possible items, associated with a probability space $(\mathscr{I}, \mathcal{F}, P)$.
- $S = \{\omega_1, \omega_2, \ldots, \omega_m\}$: a sample of size m drawn i.i.d. from the input space $\mathscr{I}$ with the distribution P.
- The sampled data (items in the sample S) may reveal a certain meaningful organization of the general data (items in the input space $\mathscr{I}$).
  - In multi-class classification in the supervised learning, the natural partition of the sample S by the class labels of the items in S reveals a partition of the input space $\mathscr{I}$.

- If the distribution P on $\mathscr{I}$ is a mixture of k unimodal distributions P_i on $\mathscr{I}$, $1 \leq i \leq k$, i.e., $P(E) = \sum_{i=1}^{k} p_i P_i(E)\ \forall\ E \in \mathcal{F}$ with $\sum_{i=1}^{k} p_i = 1$, then the sample S may reveal k modes.
  - A discrete or absolutely continuous distribution P is called unimodal if its pmf or pdf has at most one local maximum.
  - Since any distribution can be well approximated by a discrete distribution, it can be classified as a unimodal distribution or not.
- The general data have a hierarchical structure which may be revealed by the sampled data as a clustering tree.
  - A clustering tree for the sampled data has the sample S as the root, the parts of a partition of S as leaves, and a parent node is just the union of its child nodes.

- A metric d on the input space $\mathscr{I}$ may help to organize the general data (the items in the input space $\mathscr{I}$).
  - A metric $d : \mathscr{I} \times \mathscr{I} \to \mathbb{R}^+$ on the input space $\mathscr{I}$ is a nonnegative function of ordered pairs over $\mathscr{I}$ such that for all $\omega, \omega', \omega'' \in \mathscr{I}$,
    * $d(\omega, \omega') = 0$ if and only if $\omega = \omega'$;
    * symmetry: $d(\omega, \omega') = d(\omega', \omega)$;
    * triangular inequality: $d(\omega, \omega') \leq d(\omega, \omega'') + d(\omega', \omega'')$.
  - The modes established from sampling a mixture of unimodal distributions may be revealed by considering the distance between sampled items in S measured by the metric d.
  - The proximity of items in the sample S determined by the distance measure d can be exploited to establish a clustering tree for S which will exhibit a hierarchical structure of the general data.

- Problem: Find a partition $\mathbb{P}' = \{C'_1, C'_2, \ldots, C'_k\}$ of the input space $\mathscr{I}$ which represents a certain meaningful organization of the general data revealed by the sampled data.

Comments on the Input of a Clustering Algorithm

- Intuitively, clustering is the task of grouping sampled data such that close-by or similar items end up in the same group and far away or dissimilar items are separated into different groups.
- The metric d helps to organize the sampled data (and then the general data) in some meaningful way.
- Instead, a similarity measure $s : \mathscr{I} \times \mathscr{I} \to [0, 1]$ on $\mathscr{I}$ can be used to help organizing the data in some meaningful way, where s is symmetric and $s(\omega, \omega) = 1$ for all $\omega \in \mathscr{I}$.
  - A normalized PDF kernel K on $\mathscr{I}$ can serve as a similarity measure on $\mathscr{I}$.

- A metric d on $\mathscr{I}$ may be constructed by a feature mapping $\Phi : \mathscr{I} \to \mathbb{R}^N$ from the input space $\mathscr{I}$ to the N-dimensional Euclidean fracture space $\mathbb{R}^N$ such that

$$d(\omega, \omega') = \|\Phi(\omega) - \Phi(\omega')\|,$$

  where $\|\cdot\|$ is a norm on $\mathbb{R}^N$.

- k: number of clusters needed, a parameter up to learner's choice.

Comments on the Output of a Clustering Algorithm

- Hard partition: a partition $\mathbb{P} = \{C_1, C_2, \ldots, C_k\}$ of the sample S. Equivalently, a hard membership indicator $\boldsymbol{h}_j$ is assigned to the jth item ω_j in the sample S so that $\boldsymbol{h}_j = \boldsymbol{e}_i$ iff $\omega_j \in C_i$, where $\boldsymbol{e}_i, i = 1, 2, \ldots, k$ are standard unit vectors in $\mathbb{R}^k$.
- Soft partition: a soft membership indicator $\boldsymbol{h}_j = [p_1(\omega_j), p_2(\omega_j), \ldots, p_k(\omega_j)]^T$ is assigned to the jth item ω_i in the sample S so that $p_i(\omega_j)$ is the probability that the jth item ω_j is a member of cluster C_i and $\sum_{i=1}^{k} p_i(\omega_j) = 1$.

Linkage-Based Clustering

- $S = \{\omega_1, \omega_2, \ldots, \omega_m\}$: a given sample of size m.
- Initial partition: $\mathbb{P} = \{\{\omega_1\}, \{\omega_2\}, \ldots, \{\omega_m\}\}$.
- Agglomerative iteration: merging two clusters A and B in $\mathbb{P}$ to form a new cluster $A \cup B$ in $\mathbb{P}$ if the distance $D(A; B)$ of A and B is the smallest among all pairs of clusters in $\mathbb{P}$ until a stopping criterion is reached.
  - Without employing a stopping criterion, a clustering dendrogram, i.e. a clustering tree, will be created with the sample S as the root and all singletons of S as leaves.

- $D(A, B)$: distance between two clusters A and B.
  - Single linkage clustering: $D(A; B) = \min_{x \in A, y \in B} d(x, y)$.
  - Average linkage clustering:
    $D(A; B) = \frac{1}{|A||B|} \sum_{x \in A, y \in B} d(x, y)$.
  - Max linkage clustering: $D(A; B) = \max_{x \in A, y \in B} d(x, y)$.
- Stopping criterion.
  - A fixed number k of clusters is reached.
  - An upper bound r of cluster distance is broken:
    $r < D(A, B)$ for all distinct A, B in $\mathbb{P}$.
    - Scaled distance upper bound:
      $r = \alpha \max\{d(\omega_i, \omega_j) \mid \omega_i, \omega_j \in S\}$ for some $0 < \alpha < 1$.

Cost Minimization Clustering

Agnostic Cost Minimization Clustering

- $\mathscr{I}$: the input space of all possible items, associated with a probability space $(\mathscr{I}, \mathcal{F}, P)$.
- $d : \mathscr{I} \times \mathscr{I} \to \mathbb{R}^+$: a metric on the input space $\mathscr{I}$.
- k: the number of clusters, a parameter up to the learner's selection.
- $\mathbb{P}' = \{C'_1, C'_2, \ldots, C'_k\}$: a partition of the input space $\mathscr{I}$, where each cluster C'_i is represented by its f-centroid ϖ_i.
  - $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a monotone increasing function.
  - The f-centroid $\varpi_i(C'_i)$ of a cluster C'_i is defined as

$$\varpi_i(C'_i) \triangleq \arg \min_{\varpi \in C'_i} E[f(d(\omega, \varpi)) \mid \omega \in C'_i].$$

- Problem: Find a partition $\mathbb{P}'$ such that the average deviation of an item in each cluster from the f-centroid of the cluster

$$G(\mathbb{P}') = \sum_{i=1}^{k} P[C_i'] E[f(d(\omega, \varpi_i(C_i'))) \mid \omega \in C_i'].$$

  is minimized.

- When $f(x) = x^2$, the average deviation

$$G_{k\text{-means}}(\mathbb{P}') = \sum_{i=1}^{k} P[C_i'] E[d^2(\omega, \varpi_i(C_i')) \mid \omega \in C_i']$$

  is called the k-means cost function of the partition $\mathbb{P}'$, where the squared-centroid $\varpi_i(C_i')$ of cluster C_i' is

$$\varpi_i(C_i') \triangleq \arg \min_{\varpi \in C_i'} E[d^2(\omega, \varpi) \mid \omega \in C_i'].$$

Empirical Cost Minimization Clustering

- $\mathscr{I}$: the input space of all possible items, associated with a probability space $(\mathscr{I}, \mathcal{F}, P)$.
- $d : \mathscr{I} \times \mathscr{I} \to \mathbb{R}^+$: a metric on the input space $\mathscr{I}$.
- $S = \{\omega_1, \omega_2, \ldots, \omega_m\}$: a given sample of size m.
- k: the number of clusters, a parameter up to the learner's selection.
- $\mathbb{P}' = \{C'_1, C'_2, \ldots, C'_k\}$: a partition of $\mathscr{I}$, where each cluster C'_i is characterized by a representative item $\varpi_i(C'_i)$ in C'_i.
- $\mathbb{P} = \{C_1, C_2, \ldots, C_k\}$: the empirical partition associated with the partition $\mathbb{P}'$ under the sample S such that $C_i = C'_i \cap S$ for all $1 \leq i \leq k$.
  - $\frac{|C_i|}{|S|} = \frac{|C_i|}{m}$ is an estimator of $P[C'_i]$.

- Types of representative $\varpi_i(C'_i)$:
  - Agnostic-empirical f-centroid of C'_i:

$$\varpi_i(C'_i) = \arg\min_{\varpi \in C'_i} \frac{1}{|C_i|} \sum_{\omega \in C_i} f(d(\omega, \varpi)) = \arg\min_{\varpi \in C'_i} \sum_{\omega \in C_i} f(d(\omega, \varpi)).$$

  - Empirical f-centroid of C'_i, i.e.,

$$\varpi_i(C'_i) = \arg\min_{\varpi \in C_i} \frac{1}{|C_i|} \sum_{\omega \in C_i} f(d(\omega, \varpi)) = \arg\min_{\varpi \in C_i} \sum_{\omega \in C_i} f(d(\omega, \varpi)).$$

- Problem: Find an empirical partition $\mathbb{P}$ of the sample S such that the empirical average deviation of a sample item in a cluster from the representative of the cluster

$$G(\mathbb{P}) = \sum_{i=1}^{k} \frac{|C_i|}{m} \frac{1}{|C_i|} \sum_{\omega \in C_i} f(d(\omega, \varpi_i)) = \frac{1}{m} \sum_{i=1}^{k} \sum_{\omega \in C_i} f(d(\omega, \varpi_i))$$

  is minimized.

- This empirical average deviation $G(\mathbb{P})$ is called the empirical cost function of the empirical partition $\mathbb{P}$.

- Empirical cost function:
  - Empirical k-means cost function: $f(x) = x^2$ and

$$G_{k\text{-means}}(\mathbb{P}) = \sum_{i=1}^{k} \sum_{\omega \in C_i} d^2(\omega, \varpi_i(C'_i))$$

with the agnostic-empirical squared-centroid of C'_i

$$\varpi_i(C'_i) = \arg \min_{\varpi \in C'_i} \sum_{\omega \in C_i} d^2(\omega, \varpi);$$

– Empirical k-medoids cost function: $f(x) = x^2$ and

$$G_{k\text{-medoids}}(\mathbb{P}) = \sum_{i=1}^{k} \sum_{\omega \in C_i} d^2(\omega, \varpi_i(C'_i))$$

with the empirical squared-centroid of C'_i

$$\varpi_i(C'_i) = \arg\min_{\varpi \in C_i} \sum_{\omega \in C_i} d^2(\omega, \varpi);$$

– Empirical k-medians cost function: $f(x) = x$ and

$$G_{k\text{-medians}}(\mathbb{P}) = \sum_{i=1}^{k} \sum_{\omega \in C_i} d(\omega, \varpi_i(C'_i))$$

with the empirical absolute-centroid of C'_i

$$\varpi_i(C'_i) = \arg\min_{\varpi \in C_i} \sum_{\omega \in C_i} d(\omega, \varpi).$$

k-Means Clustering Algorithm

- The agnostic-empirical squared-centroid $\varpi(C_i')$ of the cluster C_i' can only be estimated.
- If $\mathscr{I}$ is a subset of $\mathbb{R}^N$ and an item is a point $\omega = \boldsymbol{x}$, an estimate is

$$\varpi(C_i') = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}.$$

Feature Learning

The Problem of Feature Learning

- $\mathscr{I}$: the input space of all possible items, which is associated with a probability space $(\mathscr{I}, \mathcal{F}, P)$.
- $\mathbb{R}^d$: the feature space.
- $\Phi : \mathscr{I} \to \mathbb{R}^d$: a feature mapping from the input space $\mathscr{I}$ to the d-dimensional feature space $\mathbb{R}^d$.
  - The feature mapping Φ from the input space $\mathscr{I}$ to the RKHS of a PDS kernel K over the input space $\mathscr{I}$ is a useful feature mapping.
- Problem: to automate the process of finding a good feature mapping.
  - The No-Free-Lunch theorem tells us that we must incorporate some prior knowledge on the distribution P in order to build a good feature representation.

Dictionary Learning: The "Bag-of-Words" Example

- A: an alphabet of letters
- $D = \{w_1, w_2, \ldots, w_k\}$: a dictionary of words over the alphabet A.
  - A word is a finite string of letters in A.
- $\boldsymbol{p} = (p_1, \ldots, p_d)$: a document, where each p_i is a word over A in the document.
- $\boldsymbol{x} = \Phi(\boldsymbol{p})$ in $\mathbb{R}^k$: a "bag-of-words" vector associated with the document $(p_1, \ldots, p_d)$ where $x_i = 1$ if $w_i = p_j$ for some $j \in [1, d]$ and $x_i = 0$ otherwise.
- It was empirically observed in many text processing tasks that linear predictors are quite powerful when applied on this representation.

Comments

- In the "bag-of-words" example, words in the dictionary can be regarded as features of documents. The "bag-of-words" vector of a document indicates what features this document has.
- In object recognition, the item is an image and the goal is to recognize which object appears in the image.
  - What we would like to have is a mapping Φ that would take the pixel-based representation of the image and would output a bag of "visual words" representing the content of the image.
  - A "visual word" can be "there is an eye in the image".
- Problem: how can we learn a dictionary of "visual words" such that a "bag-of-words" representation of an image would be helpful for predicting which object appears in the image?

Dictionary Learning Using Auto-Encoders

- $\psi : \mathbb{R}^d \rightarrow \mathbb{R}^k$: an "encoder" function.
- $\phi : \mathbb{R}^k \rightarrow \mathbb{R}^d$: a "decoder" function.
- Problem: to learn a pair of encoder and decoder functions such that the reconstruction error,

$$\sum_i \|\boldsymbol{x}_i - \phi(\psi(\boldsymbol{x}_i))\|^2,$$

  is minimized.

PCA As an Autoencoder

- k : a fixed integer in $[1, d]$.
- $\mathcal{P}_k$: the family of all rank-k $d \times d$ orthogonal projection matrices.
- $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m]$: the data block associated with a sample S of size m drawn from the input space $\mathbb{R}^d$.
- Problem : to project the input data block $\mathbf{X}$ in the d-dimensional input space $\mathbb{R}^d$ onto the data in a k-dimensional linear subspace of $\mathbb{R}^k$ that minimizes the sum of the squared L_2-distances between the original data and the projected data,

$$\min_{\mathbf{P} \in \mathcal{P}_k} \|\mathbf{PX} - \mathbf{X}\|_F^2 = \min_{\mathbf{P} \in \mathcal{P}_k} \sum_{i=1}^{m} \|\mathbf{P}\boldsymbol{x}_i - \boldsymbol{x}_i\|^2.$$

- $\mathbf{C} \triangleq \frac{1}{m}\mathbf{X}\mathbf{X}^T$: the covariance matrix of the data matrix $\mathbf{X}$.
- $\mathbf{U}_k^* = [\mathbf{u}_1, \ldots, \mathbf{u}_k]$: an $d \times k$ matrix consisting of orthonormal eigenvectors corresponding to the k largest eigenvalues of the covariance matrix $\mathbf{C}$.
- $\psi : \mathbb{R}^d \to \mathbb{R}^k$: the encoder function, which is $\boldsymbol{y}_i = \psi(\boldsymbol{x}_i) = \mathbf{U}_k^{*t}\boldsymbol{x}_i$ for all $i \in [1, m]$.
- $\phi : \mathbb{R}^k \to \mathbb{R}^d$: the decoder function, which is $\boldsymbol{x}'_i = \psi(\boldsymbol{y}_i) = \mathbf{U}_k^*\boldsymbol{y}_i$ for all $i \in [1, m]$.
- $\boldsymbol{x}'_i = \phi(\psi(\boldsymbol{x}_i)) = \mathbf{U}_k^*\mathbf{U}_k^{*T}\boldsymbol{x}_i = \mathbf{P}^*\boldsymbol{x}_i$: $\mathbf{P}^*$ is the optimal orthogonal projection in $\mathcal{P}_k$.

Clustering As an Autoencoder

- $c : \mathbb{R}^d \to \{1, 2, \ldots, k\}$: a learned clustering function such that $c(\mathbf{x})$ is the cluster to which the instance $\mathbf{x}$ belongs.
  - We may regard the clusters as "words" and instances as "documents".
- $\psi : \mathbb{R}^d \to \mathbb{R}^k$: the "encoder" function, where the feature vector $\psi(\boldsymbol{x})$ is the standard unit vector $\boldsymbol{e}_i$ in $\mathbb{R}^k$ if and only if $\boldsymbol{x}$ belongs to the ith cluster.
- $\phi : \mathbb{R}^k \to \mathbb{R}^d$: the "decoder" function, where $\phi(\boldsymbol{e}_i) = \varpi(C'_i)$ in $\mathbb{R}^d$.

A Formulation of Autoencoder

- s: a small positive integer.
- $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$: k vectors in the input space $\mathbb{R}^d$.
- $\psi : \mathbb{R}^d \to \mathbb{R}^k$: an "encoder" function such that $\|\psi(\boldsymbol{x})\|_0 \leq s$, where $\|\psi(\boldsymbol{x})\|_0$ is the number of nonzero components in $\psi(\boldsymbol{x})$.
- $\phi : \mathbb{R}^k \to \mathbb{R}^d$: a "decoder" function such that

$$\phi(\boldsymbol{v}) = \sum_{i=1}^{k} v_i \boldsymbol{\mu}_i.$$

- Since our goal is to have a small reconstruction error, and therefore we can define

$$\psi(\boldsymbol{x}) = \arg \min_{\boldsymbol{v} \in \mathbb{R}^k, \|\boldsymbol{v}\|_0 \leq s} \|\boldsymbol{x} - \phi(\boldsymbol{v})\|^2.$$

  - When $s = 1$ and we further restrict $\|\boldsymbol{v}\|_1 = 1$, then we

obtain the k-means encoding function.

- A more practical definition is

$$\psi(\boldsymbol{x}) = \arg\min_{\boldsymbol{v}\in\mathbb{R}^k} \left[\|\boldsymbol{x} - \phi(\boldsymbol{v})\|^2 + \lambda\|\boldsymbol{v}\|_1\right].$$

  - $\lambda > 0$: a regularization parameter.

- Problem: to learn a k-set $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k\}$ such that the reconstruction error,

$$\sum_i \|\boldsymbol{x}_i - \phi(\psi(\boldsymbol{x}_i))\|^2,$$

  is minimized.

- This is a computationally hard problem (similar to the k-means problem).

- Several heuristic search algorithms may give reasonably good solutions.

Neural Networks for Autoencoders

- Autoencoders may be thought of as being a special case of feedforward networks, with hidden layers to extract essential features of the input vectors.
- The size of each hidden layer is intentionally smaller than the dimension of the input data.