

INT301 W11

INTRODUCTION TO PRINCIPAL COMPONENT ANALYSIS (PCA)

Eigenvalues and Eigenvectors

如果 \mathbf{v} 是一个非零向量, λ 是一个数字, 并且:

$$A\mathbf{v} = \lambda\mathbf{v}$$

那么 \mathbf{v} 是 A 的 eigenvector (特征向量), λ 是 A 的 eigenvalue (特征值)。

The diagram shows the equation $\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 3 \times \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Red arrows point from labels to parts of the equation: 'A' points to the matrix, 'V (eigenvectors)' points to the vector $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and ' λ (eigenvalues)' points to the scalar 3.

最多可以有多少特征值:

$$\mathbf{S}\mathbf{v} = \lambda\mathbf{v} \iff (\mathbf{S} - \lambda\mathbf{I})\mathbf{v} = 0$$

仅在 $|\mathbf{S} - \lambda\mathbf{I}| = 0$ 时具有一个非零解, 并最多具有 m 个不同的 λ 值 (m 是 \mathbf{v} 的维度)。

注: \mathbf{I} 表示单位矩阵, 即在主对角线上元素均为 1, 而其他元素都是 0 的方阵。

对于对称矩阵 (symmetric matrices, 指以主对角线为对称轴, 各元素对应相等的矩阵), 特征值不同的特征向量是正交的:

$$\mathbf{S}\mathbf{v}_{\{1,2\}} = \lambda_{\{1,2\}}\mathbf{v}_{\{1,2\}}, \text{ and } \lambda_1 \neq \lambda_2 \Rightarrow \mathbf{v}_1 \bullet \mathbf{v}_2 = 0$$

实对称矩阵 (real symmetric matrix, 矩阵的元素都为实数) 的所有特征值都是实数。

正半定矩阵 (positive semidefinite matrix, 设 A 为实对称矩阵, 若对于每个非零实向量 \mathbf{X} , 都有 $\mathbf{X}^T \mathbf{A} \mathbf{X} \geq 0$, 则称 A 为半正定矩阵) 的所有特征值都是非负的:

$$\forall \mathbf{w} \in \mathfrak{R}^n, \mathbf{w}^T \mathbf{S} \mathbf{w} \geq 0, \text{ then if } \mathbf{S}\mathbf{v} = \lambda\mathbf{v} \Rightarrow \lambda \geq 0$$

Example

求 $\mathbf{S} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ 的特征值和特征向量。

由 $\mathbf{S}\mathbf{v} = \lambda\mathbf{v} \iff (\mathbf{S} - \lambda\mathbf{I})\mathbf{v} = 0$ 可以得到:

$$\begin{aligned}
 (S - \lambda \mathbf{I})\mathbf{v} &= 0 \\
 \left(\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \mathbf{v} &= 0 \\
 \left(\begin{bmatrix} 2-\lambda & 1-0 \\ 1-0 & 2-\lambda \end{bmatrix} \right) \mathbf{v} &= 0 \\
 [(2-\lambda)^2 - 1]\mathbf{v} &= 0
 \end{aligned}$$

因为 \mathbf{v} 是非零向量，所以 $(2-\lambda)^2 - 1 = 0$ ，因此 $\lambda = 1$ 或 3 。

由于 S 是对称矩阵，因此其中特征值不同的特征向量是正交的：

$$\mathbf{v} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \text{ 或 } \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Diagonal Decomposition

设平方矩阵 $\mathbf{S} \in \mathbb{R}^{m \times m}$ 具有 m 个线性独立特征向量 (square matrix)。

Theorem: Exists an eigen decomposition (对角线分解)

$$\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$$

\mathbf{U} 的列是 S 的特征向量， $\mathbf{\Lambda}$ 的对角元素是 S 的特征值

$$\mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_m), \quad \lambda_i \geq \lambda_{i+1}$$

Write \mathbf{U} with the eigenvectors as columns: $\mathbf{U} = \begin{bmatrix} \mathbf{v}_1 & \dots & \mathbf{v}_m \end{bmatrix}$

Then, \mathbf{SU} can be written

$$\mathbf{SU} = S \begin{bmatrix} \mathbf{v}_1 & \dots & \mathbf{v}_m \end{bmatrix} = \begin{bmatrix} \lambda_1 \mathbf{v}_1 & \dots & \lambda_m \mathbf{v}_m \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 & \dots & \mathbf{v}_m \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \dots & \\ & & \lambda_m \end{bmatrix}$$

Thus $\mathbf{SU} = \mathbf{U}\mathbf{\Lambda}$, or $\mathbf{U}^{-1}\mathbf{SU} = \mathbf{\Lambda}$

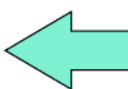
Therefore $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$.

Example

Recall $S = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}; \lambda_1 = 1, \lambda_2 = 3.$

The eigenvectors $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ form $U = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$

Inverting U , we have

$$U^{-1} = \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{bmatrix}$$


Recall
 $UU^{-1} = I.$

$$\text{Then, } S = U\Lambda U^{-1} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{bmatrix}$$

Let's divide U (and multiply U^{-1}) by $\sqrt{2}$

$$\text{Then, } S = \underbrace{\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}}_Q \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}}_\Lambda \underbrace{\begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}}_{(Q^{-1} = Q^T)}$$

Symmetric Diagonal Decomposition

- **Theorem:** If $S \in \mathbb{R}^{m \times m}$ is a **symmetric** matrix, there exists an **eigen decomposition**, where Q is **orthogonal**:

$$S = Q\Lambda Q^T$$

- $Q^{-1} = Q^T$
- Columns of Q are normalized eigenvectors
- Columns are orthogonal.
- (everything is real)

Singular Value Decomposition (SVD)

对一个秩 (rank) 为 k 的 $m \times n$ 的矩阵 A , 存在一个因数分解 (factorization, 奇异值分解 = SVD):

$$A = U\Sigma V^T$$

$m \times m$

$m \times n, \text{ nonnegative real}$

$n \times n$

U 的列是 AA^T 的正交特征向量 (orthogonal eigenvectors);

V 的列是 $A^T A$ 的正交特征向量 (orthogonal eigenvectors);

AA^T 的特征值 $\lambda_1 \dots \lambda_r$ 也是 $A^T A$ 的特征值:

$$\sigma_i = \sqrt{\lambda_i}$$

$$\Sigma = \text{diag}(\sigma_1 \dots \sigma_r), \text{ 注: 这个是奇异值 (Singular values)}$$

■ Illustration of SVD dimensions and

$$\begin{aligned}
 \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_A &= \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_U \underbrace{\begin{bmatrix} \bullet & & \\ & \bullet & \\ & & \bullet \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{V^T} \\
 \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_A &= \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_U \underbrace{\begin{bmatrix} \bullet & & & & \\ & \bullet & & & \\ & & \bullet & & \\ & & & \bullet & \\ & & & & \bullet \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{V^T}
 \end{aligned}$$

Example

$$\text{Let } A = \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Thus $m=3$, $n=2$. Its SVD is

$$\begin{bmatrix} 0 & 2/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & 1/\sqrt{6} & -1/\sqrt{3} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{3} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

Dimensionality Reduction

处理高维数据的一种方法是降低其维数，使用线性或非线性变换将高维数据投影到低维子空间。

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix} \xrightarrow{\text{Reduce dimensionality}} \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_k \end{bmatrix} \quad (K \ll N)$$

线性变换易于计算：

$$\begin{array}{ccccc}
 & Y & = & U & X & (b_i = u_i^t a_i) \\
 & \nearrow & & \nearrow & \nwarrow & \\
 k \times 1 & & & k \times d & d \times 1 & (k \ll d)
 \end{array}$$

Process

简单来说，就是丢失一些信息，从而把数据从高维转到低维。

• Find a basis in a low dimensional sub-space:

- Approximate vectors by projecting them in a low dimensional sub-space:

(1) Original space representation:

$$x = a_1 v_1 + a_2 v_2 + \dots + a_N v_N$$

where v_1, v_2, \dots, v_n is a base in the original N-dimensional space

$$\begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_N \end{bmatrix}$$

(2) Lower-dimensional sub-space representation:

$$\hat{x} = b_1 u_1 + b_2 u_2 + \dots + b_K u_K$$

where u_1, u_2, \dots, u_K is a base in the K-dimensional sub-space ($K < N$)

$$\begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_K \end{bmatrix}$$

1'

- If $K=N$, then $\hat{x} = x$
- Example ($K=N$):

$$v_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, v_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, v_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (\text{standard basis})$$

$$x_v = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} = 3v_1 + 3v_2 + 3v_3$$

$$u_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, u_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, u_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (\text{some other basis})$$

$$x_u = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} = 0u_1 + 0u_2 + 3u_3$$

thus, $x_v = x_u$

Principal Component Analysis (PCA)

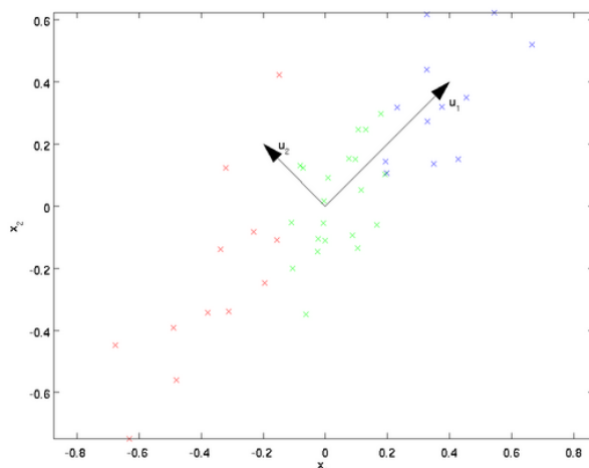
每一种降维技术都通过满足一定的标准(例如，信息丢失、数据识别等)来找到适当的转换。

PCA 的目标是降低数据的维数，同时尽可能多地保留数据集中存在的变化。

Principal Component

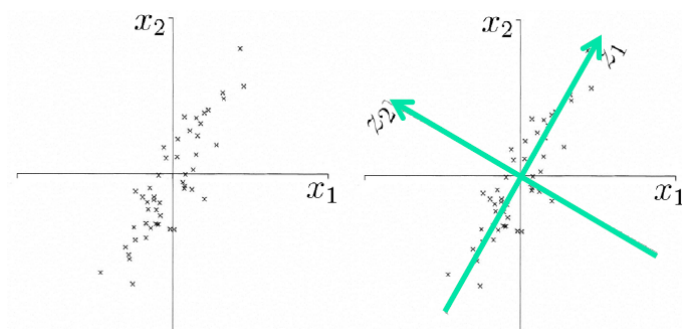
PCA 顾名思义，就是找出数据里最主要的方面，用数据里最主要的方面来代替原始数据。

主成分分析要先找到主成分。我们先看看最简单的情况，也就是 $n=2$ ， $n'=1$ ，也就是将数据从二维降维到一维。数据如下图。我们希望找到某一个维度方向，它可以代表这两个维度的数据。图中列了两个向量方向， u_1 和 u_2 ，其中 u_1 比 u_2 好。



为什么 u_1 比 u_2 好呢？可以有两种解释，第一种解释是样本点到这个直线的距离足够近，第二种解释是样本点在这个直线上的投影能尽可能的分开。

为了方便计算，我们选择有最大方差 (**maximum variance**) 的方向，作为 (第一) 主成分 (在多维中，主成分可能有很多个)，而其余的成分 (maximum residual variance) 的方向垂直于主成分。



Method

要进行主成分分析，先要找到主成分。根据最大方差来寻找。

方差的公式如下 (E 是 X 的期望):

$$\sigma^2 = E(X^2) - E^2(X)$$

我们假设:

$$E[\mathbf{x}] = \mathbf{0}, \quad a = \mathbf{x}^T \mathbf{q} = \mathbf{q}^T \mathbf{x}, \quad \|\mathbf{q}\| = (\mathbf{q}^T \mathbf{q})^{1/2} = 1$$

其中， \mathbf{q} 就是我们要找的主成分。

因此：

$$\begin{aligned} \sigma^2 &= E[a^2] - E[a]^2 = E[a^2] \quad (\text{因为 } E[\mathbf{x}] = \mathbf{0}) \\ &= E[(\mathbf{q}^T \mathbf{x})(\mathbf{x}^T \mathbf{q})] = \mathbf{q}^T E[\mathbf{x}\mathbf{x}^T] \mathbf{q} = \mathbf{q}^T \mathbf{R} \mathbf{q} \end{aligned}$$

上面的 \mathbf{R} 就是我们的原始数据。因此，当 \mathbf{q} 是 \mathbf{R} 的主分量时，方差最大化。现在我们要找到最大的 \mathbf{q} 。

主成分 \mathbf{q} 可以通过特征向量分解 (eigenvector decomposition) 得到：

$$\mathbf{R} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^T, \\ \mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_j, \dots, \mathbf{q}_m], \quad \mathbf{\Lambda} = \text{diag}[\lambda_1, \lambda_2, \dots, \lambda_j, \dots, \lambda_m]$$

其中， \mathbf{Q} 中的 \mathbf{q}_1 就是第一主成分， \mathbf{q}_2 就是第二主成分 ...

$$\Leftrightarrow \mathbf{R} \mathbf{q}_j = \lambda_j \mathbf{q}_j \quad j = 1, 2, \dots, m \quad \Longrightarrow \mathbf{R} \mathbf{q} = \lambda \mathbf{q}$$

最后可以通过上式获得 \mathbf{q} 。

Advantage and Limitation

Advantage

- 减少原始数据的维度：减少训练过程中的时间消耗，提高效率
- 丢弃原始数据的某些信息：如果此信息是噪声

Limitation

- 丢弃原始数据的一些信息：如果丢弃的信息很重要，则不宜应用 PCA
- 主成分的含义：可能有较差的可解释性
- PCA的线性模型：不适用于非线性问题
- 假设第一主成分具有更高的重要性

UNSUPERVISED LEARNING: HEBBIAN LEARNING & AE

无监督学习通过局部操作的一般规则发现输入数据中的重要特征或模式。

无监督学习网络通常由前馈连接和元素组成，以促进 ‘local’ learning。

Hebbian Learning

当一个神经元反复激发另一个神经元时，后一个神经元的阈值降低，或者神经元之间的突触权重增加，实际上增加了第二个神经元激发的可能性。

Hebbian learning rule: $\Delta w_{ji} = \eta y_j x_i$, η 是学习率。

Hebb 规则中不需要期望或目标信号，因此它是无监督学习。

Weight update

考虑单个权重 w 的更新 (x 和 y 是突触前和突触后的活动，即输入和输出)：

$$w(n+1) = w(n) + \eta x(n)y(n)$$

对于线性激活函数，假如 $y = w^T x = x^T w$ ：

$$w(n+1) = w(n)[1 + \eta x(n)x^T(n)]$$

权重无限制地增加。如果初始权重为负，则它将在负数中增加；如果为正数，则将在正范围内增加。

Similarity measure

对于 y (输出)，点积可以写为：

$$y = |w||x| \cos(\alpha)$$

α = 向量 x 和 w 之间的角度

- 如果 α 近似值 0 (x 和 w 很接近)，则 y 很大
- 如果 α 近似值 90 (x 和 w 很远)，则 y 是 0

使用 Hebbian learning 训练的网络根据权重中包含的信息在其输入空间中创建相似性度量 (内积)。权重在训练期间捕获 (或记忆) 数据中的信息。

在操作期间，当权重固定时，较大的输出 y 表示当前输入与在训练期间创建权重的输入 x "相似"。

Oja's Rule

简单的 Hebbian rule 会导致权重无限制地增加 (或减少)。

权重需要 normalized 为 1:

$$w_{ji}(n+1) = \frac{w_{ji}(n) + \eta x_i(n)y_j(n)}{\sqrt{\sum_i [w_{ji}(n) + \eta x_i(n)y_j(n)]^2}}$$

Oja 证明了，对于很小的 $\eta \ll 1$ ，上述归一化可以近似为：

$$w_{ji}(n+1) = w_{ji}(n) + \eta y_j(n) [x_i(n) - y_j(n)w_{ji}(n)]$$

这就是 Oja's rule，或 normalized Hebbian rule。它涉及一个"遗忘项 (forgetting term)"，防止权重无限制地增长。其中 w_{ji} 是第一个主成分。

Oja's rule 渐近收敛，不像 Hebbian rule 是不稳定的。

Oja's rule 在输入空间中创建一个主成分，作为应用于单个神经元时的权重向量。Oja's rule 可以被扩展以提取多个主成分。

Deflation method

Oja's rule 创建了主成分后，我们可以使用 deflation method 来找到其他成分（垂直于主成分）。

Deflation method 的原理是从输入中减去主成分。

采用 deflation method 计算其他特征向量：

- 假设已经获得第一个成分 (w_1)，计算第一个特征向量在输入上的投影：

$$y = w_1^T x$$

注： w_1 是一个向量，而 x 是一个方向，这两个相乘，就是 w_1 在 x 方向上的投影。

- 将修改后的输入生成为：

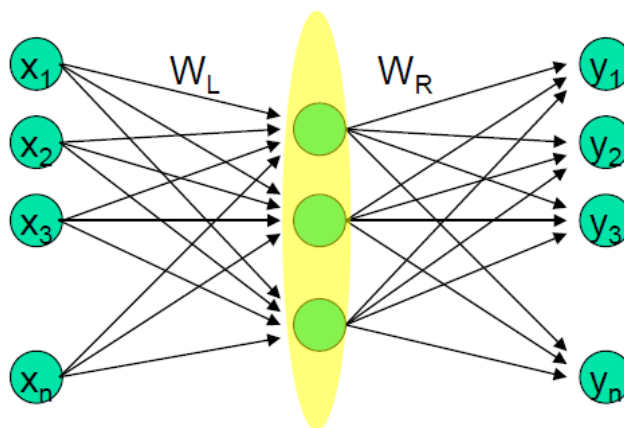
$$\hat{x} = x - w_1 y = x - w_1 w_1^T x$$

注：这是 deflation method 的核心步骤，就是从输入中减去主成分。 $w_1 y$ 是主成分，而 \hat{x} 是剩余的成分，即 residual component。

- 对修改后的数据重复 Oja's rule。

PCA in Neural Networks

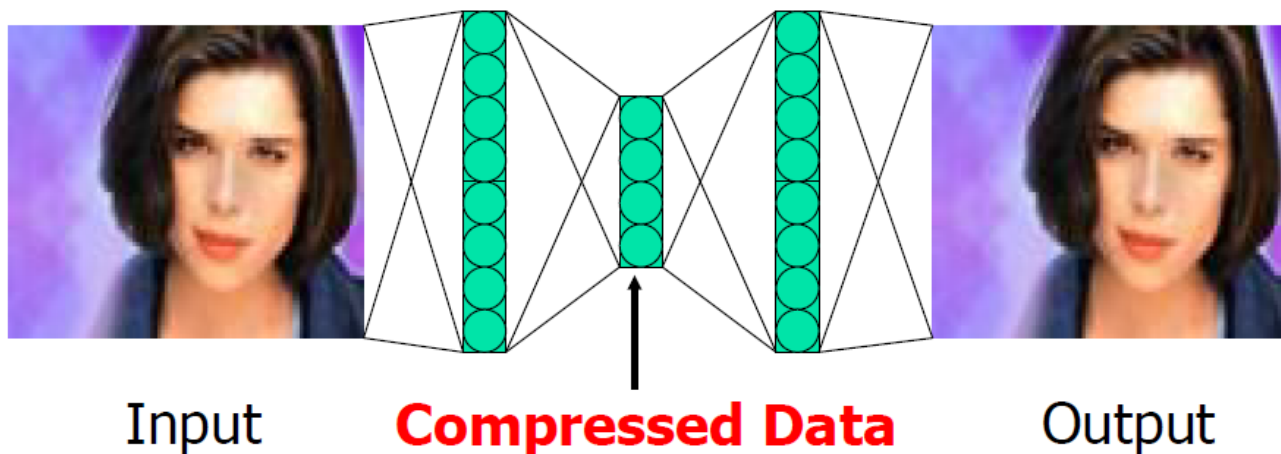
Multi-layer networks with bottleneck layer:



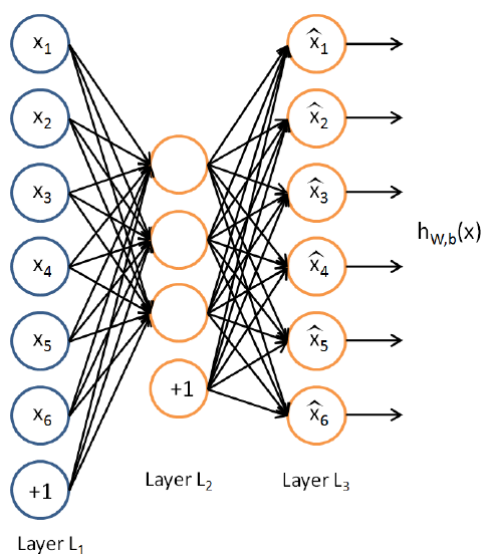
这里的 bottleneck layer 就是主成分，一共有 m 个，因此 W_L 跨过第一个 m 个主特征向量的子空间。

Train using auto-associative output: $e = x - y$, e 就是 error。使用反向传播进行无监督学习。

这里的训练是使 input 和 output 一致，以发现表征输入模式的重要特征。如果学习到最后，发现输入和输出一致，说明网络找的主成分都很重要。这可以通过学习 identity mapping 来实现，通过瓶颈传递数据：自动编码器 (auto-encoders)。



Auto-encoders



Auto-encoders 是一种前馈神经网络，它学习预测输出中的输入本身（在输出层重现输入）。

$$y^{(i)} = x^{(i)}$$

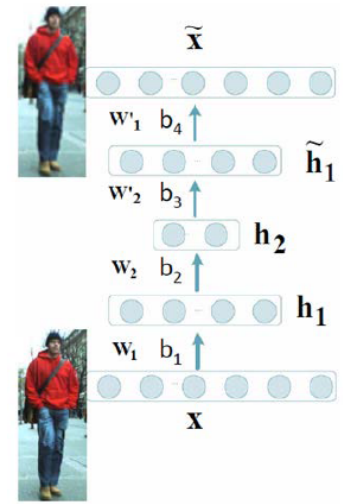
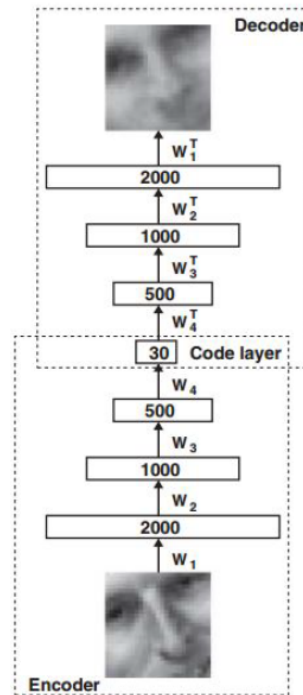
输入到隐藏的部分对应于 encoder，隐藏到输出的部分对应于 decoder。

自动编码器倾向于找到类似于 PCA 的数据描述；而瓶颈层中的少量神经元充当 information compressor (code)。

Deep Auto-encoder

就是把 encoder 和 decoder 扩展为多层。

- A deep auto-encoder is constructed by extending the encoder and decoder of autoencoder with **multiple hidden layers**.

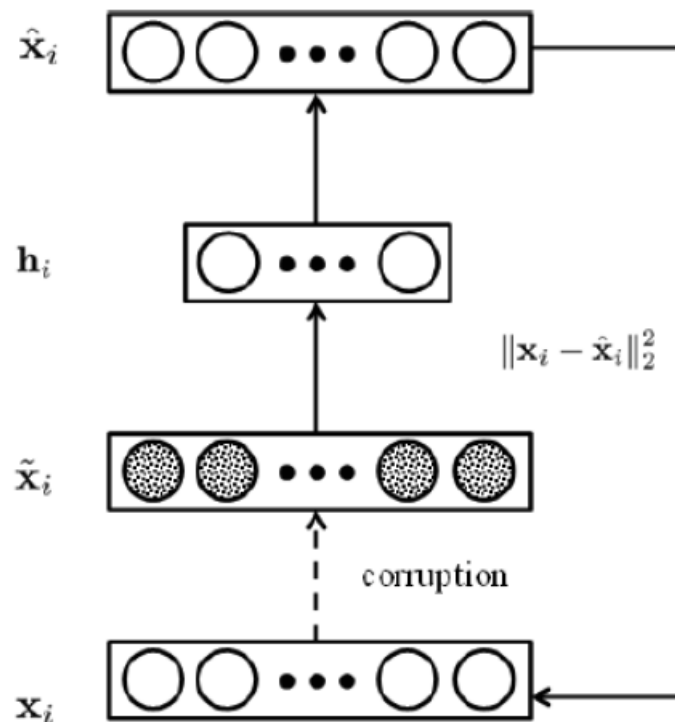


Encoding: $h_1 = \sigma(W_1 x + b_1)$
 $h_2 = \sigma(W_2 h_1 + b_2)$

Decoding: $\tilde{h}_1 = \sigma(W'_2 h_2 + b_3)$
 $\tilde{x} = \sigma(W'_1 \tilde{h}_1 + b_4)$

Denoising Auto-encoder

通过添加随机噪声，它可以迫使自动编码器学习更强大的功能。



注: x_i 到 \tilde{x}_i 添加了噪声, \tilde{x}_i 到 h_i 进行了降维, 最后 \hat{x}_i 是输出。 $\|x_i - \hat{x}_i\|_2^2$ 是 error。

The loss function:

$$\mathbf{h}^{(\ell)} = \sigma(\mathbf{W}_1 \tilde{\mathbf{x}}^{(\ell)} + \mathbf{b}_1)$$

$$\hat{\mathbf{x}}^{(\ell)} = \sigma(\mathbf{W}_2 \mathbf{h}^{(\ell)} + \mathbf{b}_2)$$

$$\min_{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2} \sum_{\ell} \|\mathbf{x}^{(\ell)} - \hat{\mathbf{x}}^{(\ell)}\|_2^2 + \lambda (\|\mathbf{W}_1\|_F^2 + \|\mathbf{W}_2\|_F^2)$$

Auto-encoders Network

网络尝试在输出中重现输入，在隐藏层中诱导 short encoding。

此编码在较小的维度空间中保留有关输入的最大信息量，以便可以重建输入。

自动编码器网络可用于降维、压缩等。