AutoEncoders

Mtro. Víctor Mijangos

Facultad de Ingeniería

13-17 de enero, 2020

Curse of dimensionality

Uno de los problemas que suelen presentar los datos es la llamada **curse of dimensionality** (o maldición de la dimensionalidad) [1].

En muchas tareas de aprenndizaje, se han observado **complicaciones** cuando los datos viven en un **espacio de alta dimensionalidad**.

Por ejemplo, en **optimización** con GD, cuando hay muchas dimensiones se vuelve **más costoso el proceso**, pues se realizan un mayor número de combinaciones.

Reducción de la dimensionalidad

Se buscan métodos que reduzcan la dimensionalidad **sin perder mucha información**. Se quiere que:

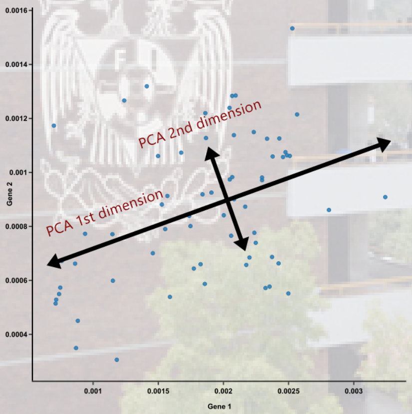
- No se modifique la topología de los datos
- Puedan visualizarse los datos
- Puedan observarse las clases naturales de los datos

Así, se han propuesto diferentes métodos de reducción de dimensionalidad. Por ejemplo:

- Lapalcian Eigenmaps
- Stochastic Neighbor Embeddings (SNE y t-SNE)
- Principal Component Analysis (PCA)

PCA

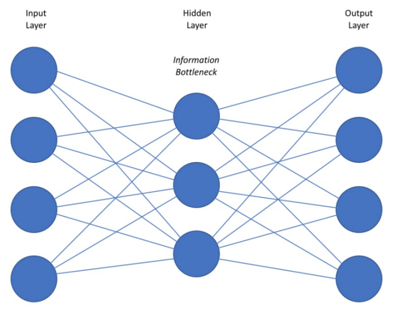
El método de **PCA** ha sido uno de los más utilizados por su simplicidad y la obtención de buenos resultados.



NNs y reducción de dimensionalidad

Las **redes neuronales** ofrecen una opción para reducir la dimensión de los datos.

Si tomamos una red con una **capa oculta de menor dimensión**, esta capa oculta puede fungir como una reducción de la dimensionalidad.



AutoEncoders

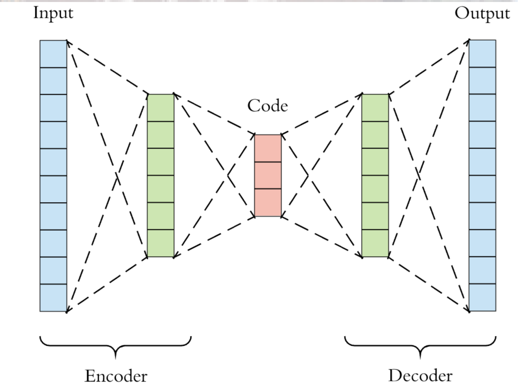
Una manera de formular una red que reduzca la dimensión de los datos consiste en:

- Tomar como entrada los vectores en dimensión alta
- Llevarlos a una capa oculta de baja dimensionalidad
- Reconstruir los datos originales a partir de esta capa oculta.

Este tipo de redes se llama **AutoEncoders**. De esta forma, se trata de una red **no supervisada**.

AutoEncoders

Los AutoEncoders constan de dos pares: **Encoder**, que codifica un vector de menor dimensión; y **Decoder** que decodifica para reconstruir el vector original.



AutoEncoders

Podemos formalizar el forward de un AutoEncoder de la siguiente forma:

- **Encoder** ($h^{(0)} = x \in \mathbb{R}^d$, $h^{(r)} \in \mathbb{R}^{m_r}$ y $m_r < m_{r-1} ... m_1 < m_0$):

$$h^{(r)} = g(W^{(r)}x + b^{(r)})$$

- **Decoder** ($h_{r+k} \in \mathbb{R}^{m_{r+k}}$, con $k = 1, ..., r$; $m_{r+k} < m^{(r+k+1)} .. < m_{2r}$ y $m_{2r} = m_0$):

$$h^{(r+k)} = g(W^{(r+k)}x + b^{(r+k)}), k = 1, ..., r - 1)$$

$$\hat{x} = f(W^{(2r)}x + b^{(2r)})$$

Función de riesgo

Ya que el objetivo de los AutoEncoders es reconstruir los datos de entrada, la función de riesgo se determina de la siguiente forma:

$$R(\theta) = \frac{1}{2}\sum_{x} ||x - \hat{x}||^2$$

AutoEncoder lineal

El caso más simple de AutoEncoder puede definirse de la siguiente forma:

- **Encoder** ($W \in \mathbb{R}m \times d, m < d$):
$$h = Wx$$
- **Decoder** ($U \in \mathbb{R}^{d \times m}$):
$$\hat{x} = Uh$$

En este caso, tenemos un **AutoEnncoder lineal**, pues $b = 0$ y las funciones de activación son la identidad. Además:

$$\hat{x} = UWx$$

Por tanto, esperamos que $U = W^{-1}$.

Solución AutoEncoder lineal

Debido a la simplicidad de este AutoEncoder, podemos solucionarlo de forma analítica.

Debemos buscar el mínimo de la función:

$$\arg\min_{\theta} \frac{1}{2} \sum_{x} ||x - UWx||^2$$

Esta solución corresponde a los componentes principales en PCA [2].

PCA

La función de pérdida se puede reescribir de la siguiente forma:

$$
\begin{aligned}
\arg\min_{\theta} \frac{1}{2}||X - XWU||_F^2 &= \arg\min_{\theta} tr\{(X - XWU)(X - XWU)^T\} \\
&= \arg\min_{\theta} tr\{(X - XWU)^T(X - XWU)\} \\
&= \arg\min_{\theta} tr\{X^TX - X^TXWU - WUX^TX + WUX^TXWU\} \\
&= \arg\min_{\theta} tr\{X^TX\} - 2tr\{X^TXWU\} + tr\{WUX^TXWU\} \\
&= \arg\min_{\theta} -2tr\{X^TXWU\} + tr\{X^TXWUWU\} \\
&= \arg\min_{\theta} -2tr\{X^TXWU\} + tr\{X^TXWU\} \\
&= \arg\min_{\theta} -tr\{X^TXWU\}
\end{aligned}
$$

PCA

Podemos cambiar la posición de la matriz de decoder:

$$\arg\min_{\theta} -tr\{X^T XWU\} = \arg\min_{\theta} -tr\{UX^T XW\}$$

Y, finalmente, cambiando el signo, obtener un problema de maximización:

$$\arg\min_{\theta} -tr\{X^T XWU\} = \arg\max_{\theta} tr\{UX^T XW\}$$

Introducimos una restricción: queremos que la transformación sea ortogonal, i.e. $W^{-1} = W^T$, por tanto $U = W^T$. La función de riesgo final es:

$$\arg\max_{W} tr\{W^T X^T XW\}$$

Descomposición en valores singulares

Antes de solucionar esta función de riesgo, definiremos la **descomposición en valores singulares** (SVD).

El SVD es una descomposición que aproxima una matriz (la matriz de datos):

$$X^T X = VSV^*$$

- $V \in \mathbb{R}^{d\times d}$ son eigenvectors de XX^T
- $V^* \in \mathbb{R}^{d\times d}$ son eigenvectors de $X^T X$ y $V^* V = Id$.
- S es una magtriz diagonal de eigenvalores de $X^T X$ o XX^T (**valores singulares**).

Máximo de la función

El problema de mazximización se puede tranformar en:

$$\arg\max_{W} tr\{W^T X^T X W\} = \arg\max_{W} tr\{W^T (V S V^*) W\}$$

La solución es, entonces, los r eigenvectores (columnas de V^* o renglones de V) asociados a los eigenvalores más altos. Así:

$$W = V^*_{r\times d}$$

Donde r es la dimensión reducida y $V^*_{r\times d}$ es una matriz con sólo los r vectores con eigenvalores más grandes.

Normalización en PCA

En el algoritmo de PCA, es común normalizar los datos de la siguiente forma:

$$x^{norm} = \frac{x - \hat{\mu}}{\hat{\sigma}}$$

donde:

$$\hat{\mu} = \frac{1}{N} \sum_x x$$

$$\hat{\sigma}^2 = \frac{1}{N} \sum_x (x - \hat{\mu})^2$$

PCA y varianza

Al normalizar así los vectores, notamos que las entradas de la matriz $X^T X$ son de la forma:

$$(X^T X)_{i,j} = \frac{1}{\hat{\sigma}_i \hat{\sigma}_j} \sum_{k=1}^{N} (x_k^{(i)} - \hat{\mu}_i)(x_k^{(j)} - \hat{\mu}_j)$$

En otras palabras, $X^T X$ es la **matriz de covarianza** de las variables que representan los datos.

Escoger los eigenvectores con eigenvalores más altos, equivale a **maximizar la varianza** de los componentes.

Dnoising AutoEncoders

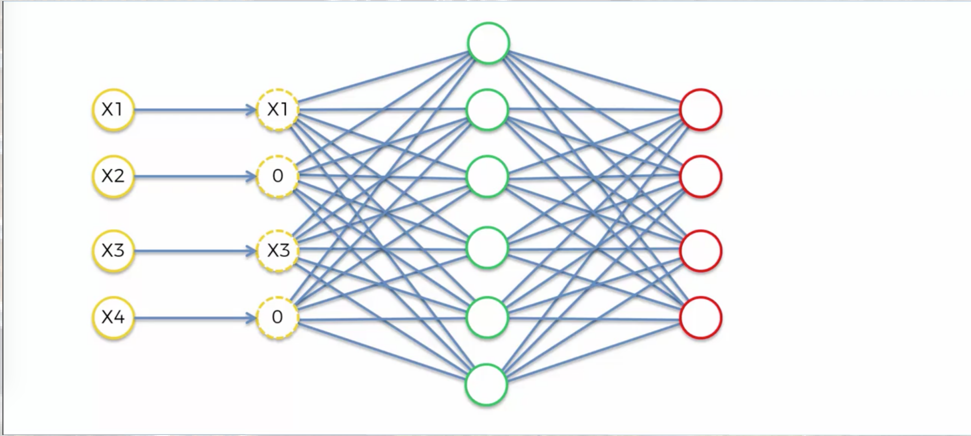
La formulación de los AutoEncoders, nos permite plantear el siguiente problema:

> *Dado datos con ruido, decodificar el mensaje original eliminando este ruido.*

Un AutoEncoder que puede solucionar este problema se conoce como **Denoising AutoEncoder** [3].

Denoising AutoEncoder

El ruido consiste en pérdida de información. “Eliminar” el ruido consiste en recuperar esta información.



Denoising AutoEncoder

En el denising AutoEncoder, la entrada no es exactamente a la salida, si no que es una versión de esta con ruido.

$$x^{noise} = corrupt(x)$$

El objetivo es, tomando como entrada x^{noise}, reconstruir el dato x original. En el caso más simple (una capa):

$$h = g(Wx^{noise} + b)$$

$$\hat{x} = f(Uh + c)$$

Denoising AutoEncoder

De igual forma que en el AutoEncoder, se debe aprender un codificador y de un decodificador. La función de riesgo es similar a la de los AutoEncoders, con la diferencia de que $\hat{x}$ se obtiene a partir de datos con ruido:

$$R(\theta) = \sum_{x} ||\hat{x} - x||^2$$

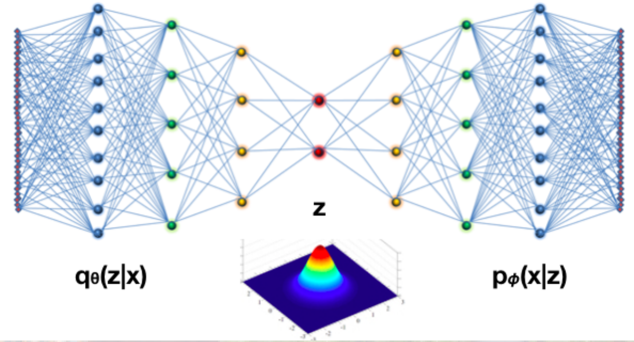
En este caso, necesitamos un conjunto de datos de entrenamiento que tenga pares de las versiones ruidosas con las versiones limpias:

$$\{(x^{noise}, x) : x^{noise} = corrupt(x)\}$$

Variational AutoEncoders

Un tipo particular de AutoEncoders, que definen un **modelo generativo**, son los **Variational AutoEncoders** (VAE).

Los VAEs estiman una distribución en un **espacio latente** de elementos z.



Variational AutoEncoders

La ventaja de los VAEs es que pueden generar datos x a partir de puntos z en el espacio latente a partir del decodificador y la probabilidad:

$$p(x|z) \propto p(z|x)p(x)$$

En este caso, se asume que existe una **distribución normal**.

Variational AutoEncoders

La función de riesgo que buscaremos minimizar es la divergencia KL:

$$
\begin{aligned}
D[q(z|x)||p(z|x)] &= \int q(z|x) \ln \frac{q(z|x)}{p(z|x)} dz \\
&= \int q(z|x) \ln \frac{q(z|x)p(x)}{p(x|z)p(z)} dz \\
&= \int q(z|x) \ln \frac{q(z|x)}{p(x|z)p(z)} dz + \int q(z|x) \ln p(x) dz \\
&= D[q(z|x)||p(z,x)] - H(q,p)
\end{aligned}
$$

VAEs

Calcular $p(x) = \int p(x, z)dz$ se vuelve intratable. Por tanto, se modifica la funcón de la siguiente forma:

$$\begin{aligned}\int q(z|x) \ln \frac{q(z|x)}{p(x|z)p(z)} dz + \int q(z|x) \ln p(x) dz &= \int q(z|x) \ln \frac{q(z|x)}{p(x|z)p(z)} dz + \ln p(x) \int q(z|x) dz \\ &= \int q(z|x) \ln \frac{q(z|x)}{p(x|z)p(z)} dz + \ln p(x)\end{aligned}$$

Y ya que la divergencia es mayor a 0, tenemos la desigualdad:

$$\begin{aligned}\ln p(x) &\geq \int q(z|x) \ln \frac{p(x|z)p(z)}{q(z|x)} dz \\ &= \int q(z|x) \ln \frac{p(z)}{q(z|x)} dz + \int q(z|x) \ln p(x|z) dz\end{aligned}$$

VAEs

Así, la función de riesgo estará definida por:

$$\begin{aligned}\arg\max_{\theta} R(\theta) &= \arg\max_{\theta} \int q(z|x) \ln \frac{p(z)}{q(z|x)} dz + \int q(z|x) \ln p(x|z) dz \\ &= \arg\max_{\theta} -[D[q(z|x)||p(z)] + H(q(z|x), p(x|z))] \\ &= \qquad = \arg\min_{\theta} D[q(z|x)||p(z)] +\end{aligned}$$

Esta función se conoce como **Evidence Lower BOund** (ELBO). Optimizar esta función equivale a minimizar la divergencia KL antes expuesta.

References

Robert Bellman.
Curse of dimensionality.
Adaptive control processes: a guided tour. Princeton, NJ, 1961.

Hervé Bourlard and Yves Kamp.
Auto-association by multilayer perceptrons and singular value decomposition.
Biological cybernetics, 59(4-5):291–294, 1988.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol.
Extracting and composing robust features with denoising autoencoders.
In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

The End