

<https://www.cs.toronto.edu/~hinton/science.pdf>

<https://www.youtube.com/watch?v=eAQsrtPzufM&list=WL&index=3>

Facilidades: clasificación, visualización, comunicación y almacenamiento de información de alta dimensionalidad

técnica más utilizada: PCA

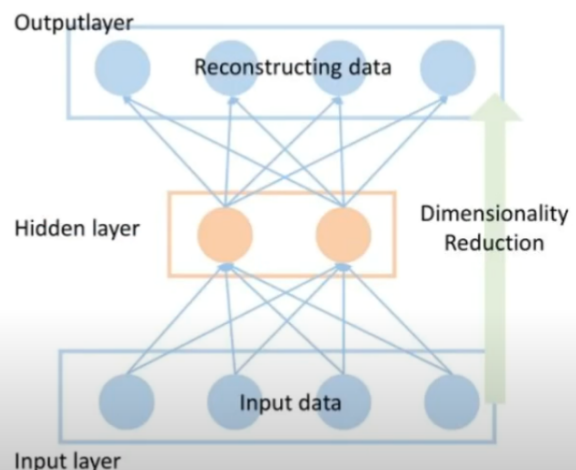
autoencoder: red neuronal multicapa

3 capas: input, hidden layer, output

Google

Autoencoder

- has exactly three layers: Input, output and hidden layer
- nonlinear generalization of PCA
- An encoder transform the high-dimensional data into a low-dimensional code
- A decoder recover the data from the code.



algunos autores consideran al autoencoder como una generalización no lineal del PCA

sabes que las redes neuronales adaptan en capas interiores “patrones”, que va a ser la información de entrada concentrada

(función de activación, si es lineal o no lineal)

autoencoder dos etapas: codificación y decodificación

empieza con valores aleatorios (como cualquier red neuronal)

datos de salida se comparan con datos de entrada, y se utiliza el error para backpropagation y así corregir los gradientes de la red (simplemente como funcionan las redes neuronales)

es complicado optimizar un autoencoder multicapa, depende mucho de los valores iniciales:

Paper Review: Reducing the Dimensionality of Data with Neural Networks

It is difficult to optimize multilayer autoencoder

- multiple hidden layers(2-4) ...
 - with large initial weights:
 - autoencoders typically find poor local minima
 - with small initial weights:
 - the gradients in the early layers are tiny, making it infeasible to train autoencoders with many hidden layers
- If the initial weights are close to a good solution, gradient decent works well. However finding such initial weights is very difficult.

Activar Windows
Ve a Configuración para activar Windows.

Google Hangouts comparte tu pantalla con hangouts.google.com. Dejar de compartir Ocultar

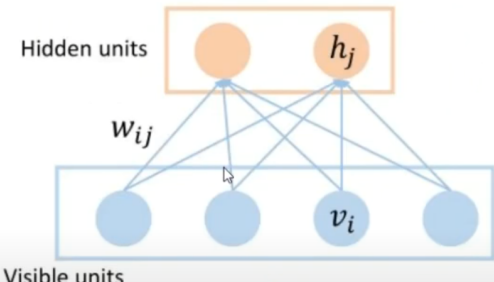
Idea del paper: pretraining para información binaria, la generaliza a real-valued data y muestra que funciona bien para una variedad de datasets

el proceso esta basado en Restricted Boltzman Machine (RBM)

Paper Review: Reducing the Dimensionality of Data with Neural Networks

Restricted Boltzmann machine (RBM)

- The input data correspond to “visible” units of the RBM and the feature detectors correspond to “hidden” units.
- A joint configuration (\mathbf{v}, \mathbf{h}) of the visible and hidden units has an energy given by (1).
- The network assigns a probability to every possible data via this energy function.



$$E(\mathbf{v}, \mathbf{h}) = - \sum_{i \in \text{pixels}} b_i v_i - \sum_{j \in \text{features}} b_j h_j - \sum_{i,j} h_j w_{ij} v_i \quad (1)$$

b_i, b_j : bias

Google Hangouts comparte tu pantalla con hangouts.google.com. Dejar de compartir Ocultar

Similar a PCA, pero trabaja con probabilidades, tiene solo dos capas

Lo que nos interesa de esta red son los pesos que genera su entrenamiento no supervisado

Paper Review: Reducing the Dimensionality of Data with Neural Networks

Pulsa [Esc] para salir del modo de pantalla completa

Pretraining consists of learning a stack of RBMs

- The first layer of feature detectors then become the visible units for learning the next RBM
- This layer-by-layer learning can be repeated as many times as desired.

Google Hangouts comparte tu pantalla con hangouts.google.com... Dejar de compartir Ocultar

Activar Windows Ve a Configuración para activar Windows.

Unrolling Fine-tuning

las w son matrices de pesos

la traspuestas de esas matrices se utilizan para decodificar

cada camino entre capa (menos la del medio) son funciones de activaciones "logísticas", las del medio son lineales

Google

Experiment(2-B): MNIST

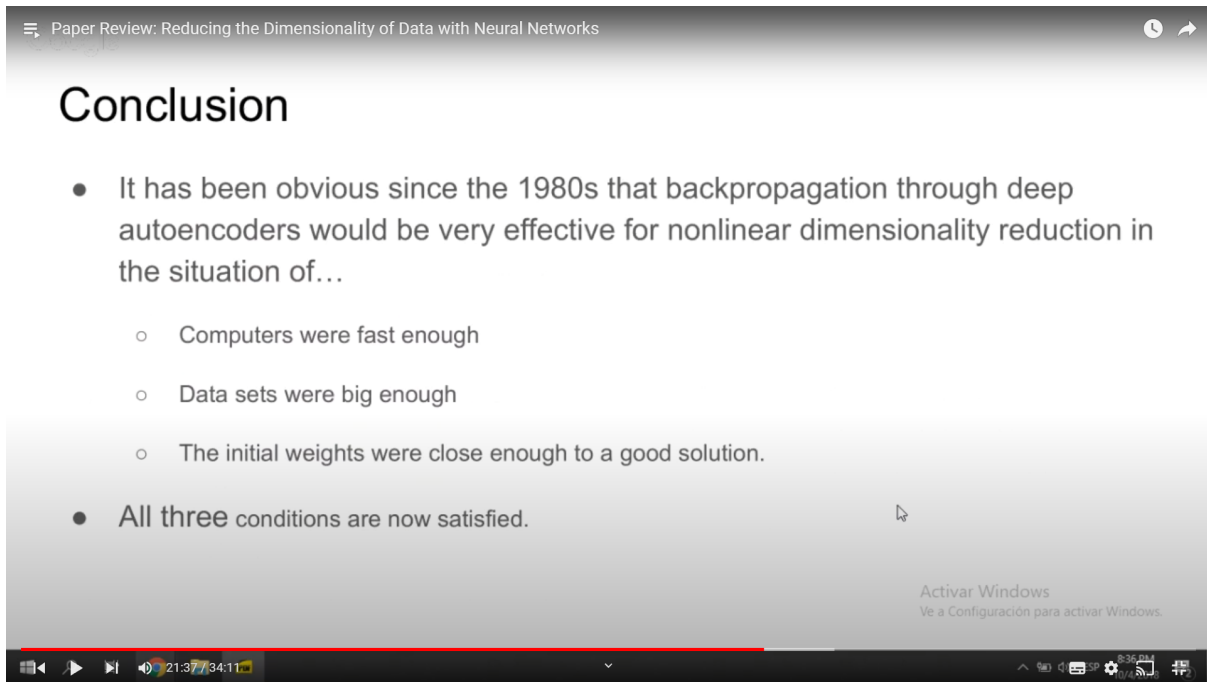
Used AutoEncoder's Network

- The function of layer
 - The 30 units in the code layer were linear and all the other units were logistic.
- Data
 - The network was trained on 60,000 images and tested on 10,000 new images.

Activar Windows Ve a Configuración para activar Windows.

8:29 PM 10/4/2018

cada capa tiene n° numeros reales



Gradient descent can be used for fine-tuning the weights in such “autoencoder” networks, but this works well only if the initial weights are close to a good solution

We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis (PCA) as a tool to reduce the dimensionality of data.

(PCA), which finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions.

Paper: We describe a nonlinear generalization of PCA that uses an adaptive, multilayer “encoder” network to transform the high-dimensional data into a low-dimensional code and a similar “decoder” network to recover the data from the code.

Starting with random weights in the two networks, they can be trained together by minimizing the discrepancy between the original data and its reconstruction.

The required gradients are easily obtained by using the chain rule to backpropagate error derivatives first through the decoder network and then through the encoder network (1).

With large initial weights, autoencoders typically find poor local minima; with small initial weights, the gradients in the early layers are tiny, making it infeasible to train autoencoders with many hidden layers.

We introduce this “pretraining” procedure for binary data (imagenes), generalize it to real-valued data, and show that it works well for a variety of data sets (aka el pretraining es entrenar cada capa de la red de forma independiente)

finding such initial weights requires a very different type of algorithm that learns one layer of features at a time.

“Pretraining consists of learning a stack of restricted Boltzmann machines (RBMs), each having only one layer of feature detectors. The learned feature activations of one RBM are used as the “data” for training the next RBM in the stack. After the pretraining, the RBMs are “unrolled” to create a deep autoencoder, which is then fine-tuned using backpropagation of error derivatives.”

two-layer network called a “restricted Boltzmann machine” in which stochastic, binary pixels are connected to stochastic, binary feature detectors using symmetrically weighted connections.

explicar que es estocastica

puede aprender una distribución de probabilidad sobre su conjunto de entradas

After pretraining multiple layers of feature detectors, the model is “unfolded” (Fig. 1) to produce encoder and decoder networks that initially use the same weights. The global finetuning stage then replaces stochastic activities by deterministic, real-valued probabilities and uses backpropagation through the whole autoencoder to fine-tune the weights for optimal reconstruction.

pasos: We introduce this “pretraining” procedure for binary data, generalize it to real-valued data, and show that it works well for a variety of data sets

cual es el problema

por que no sirve PCA?

cual es la solucion propuesta (tipo de solucion, capaz no el detalle)

15 min (10 slides max)

defenderlo, donde tiene problemas

idea: nonlinear dimensionality reduction

anda mejor que PCA por que “This is most likely the case because the autoencoder has the ability to learn non-linear features since each layers has non-linear activation units and thus can learn a more complex representation of the input”

<https://medium.com/@deeplearning1/reducing-the-dimensionality-of-data-with-neural-networks-c9b45871ff84>

A network model is used that seeks to compress the data flow to a bottleneck layer with far fewer dimensions than the original input data. The part of the model prior to and including the bottleneck is referred to as the encoder, and the part of the model that reads the bottleneck output and reconstructs the input is called the decoder.

After training, the decoder is discarded and the output from the bottleneck is used directly as the reduced dimensionality of the input. Inputs transformed by this encoder can then be fed into another model, not necessarily a neural network model.

AE trains through Gradient descent

<https://es.slideshare.net/ssuser77b8c6/reducing-the-dimensionality-of-data-with-neural-networks>

Por que?

Facilidades: clasificación, visualización, comunicación y almacenamiento de datos de alta dimensionalidad

PCA

Los resultados del paper serán comparados con los de PCA, el cual es un método bastante simple y ampliamente utilizado. Como repaso, PCA se basa en encontrar las direcciones de la mayor varianza en el dataset, y representar los puntos de dicho dataset en estas nuevas "direcciones".

Idea

Describir una generalización no lineal de PCA que usa "encoders" adaptativos de varias capas para transformar los datos de alta dimensionalidad en "codes" de baja dimensionalidad (vendrían a ser los patrones, o los datos "concentrados" si se quiere) y una red similar llamada "decoder" que se encargaría de recuperar los datos de dichos "codes".

Método de inicialización de pesos que permite a las redes autoencoder aprender patrones de baja-dimensionalidad de forma efectiva. Lo vamos a estar viendo ahora en un rato.

Autoencoder

Un autoencoder es una red neuronal multicapa con tres capas como mínimo, la de entrada una oculta y la de salida.

Una característica de las redes neuronales es que adaptan en capas interiores "patrones", que van a ser los datos de entrada concentrada. Por lo tanto, al asignarle menos unidades ("nodos") estamos efectivamente reduciendo la dimensionalidad de los datos de entrada.

El autoencoder se basa en dos etapas, codificación y decodificación de los datos, como podemos ver en el diagrama. La codificación de los datos sucede cuando estos son procesados por los pesos de las conexiones entre la capa de entrada y la capa oculta. La decodificación se realiza al procesar los datos de la capa oculta con los pesos de las conexiones entre la capa oculta y la capa de salida.

Las dos redes (encoder y decoder) pueden ser entrenadas al mismo tiempo al minimizar la discrepancia entre los datos de entrada y los reconstruidos al final del decoder utilizando backpropagation del error.

Ahora, es difícil optimizar los pesos de los autoencoders no lineales que tienen múltiples capas ocultas. Si los pesos iniciales son muy grandes el autoencoder termina encontrando usualmente un mal mínimo local, y si los pesos iniciales son muy chicos la corrección por gradiente termina siendo muy pequeña en las primeras capas, haciéndolo inviable para aprender autoencoders de muchas capas.

Si los pesos iniciales están cerca de una buena solución, el método del descenso por el gradiente funciona bien, pero encontrar estos valores iniciales es difícil.

Necesitamos algoritmo que nos permita entrenar cada capa de features una por una.

La idea introducida por el paper es un proceso de pretraining que veremos ahora, pero antes de tener que hablar un poco sobre como se componen los autoencoders, por lo cual pasaremos a hablar sobre Restricted Boltzmann Machines (RBM).

La idea introducida por el paper es un proceso de pretraining para “información binaria”, generalizarla para “datos valuados realmente” y mostrar que funciona bien para una variedad de datasets.

Restricted Boltzmann Machines (RBM)

Una máquina de Boltzmann restringida es una red neuronal artificial estocástica generativa que puede aprender una distribución de probabilidad sobre su conjunto de entradas.

two-layer network called a “restricted Boltzmann machine” in which stochastic, binary pixels are connected to stochastic, binary feature detectors using symmetrically weighted connections.

Las unidades invisibles y visibles son binarias, y los pesos de cada elemento w_{ij} de la matriz esta asociado a la conexión entre una visible unit y una hidden unit. Dados los pesos y los bias, la energía de la configuración (v, h) esta definida como se ve en pantalla:

Las RBM son redes neuronales de dos capas que funcionan de forma similar a PCA, pero a diferencia de estas trabaja con probabilidades.

A joint configuration (v,h) of the visible and hidden units has an energy given by the formula on screen.

La idea de una función de energía es que asigne “energía baja” a los valores correctos de las variables restantes, y “energía alta” a valores incorrectos. Por lo tanto se busca minimizar los resultados de esta función.

En definitiva, la red le asigna una probabilidad a cada dato posible con esta función de energía.

Una de las razones por las cuales usamos RBMs es que estas nos permiten usar las hidden units obtenidas en una como visible units en otra, y de esa manera encadenar varias de estas para formar nuestro encoder como veremos ahora.

Pretraining

Como podemos ver en pantalla entonces nuestros autoencoder está compuesto por RBMs. Entonces la primera capa del modelo, la de input (en nuestro caso la imagen), sería la capa visible de la primer RBM, y la segunda sería su hidden units correspondiente. Asimismo, esta segunda capa sería las visible units de la segunda RBM y así sucesivamente.

El preentrenamiento entonces se basa en tomar cada RBM por separado, comenzando con la que recibirá los datos originales. Empezamos sus pesos con valores aleatorios y entrenamos la RBM. Una vez entrenada esta primer RBM se utilizarán los datos obtenidos en su hidden layer como los datos de entrada de la próxima RBM y pasaremos a entrenar esta de la misma manera, y así sucesivamente.

Una vez terminado el preentrenamiento se compone el autoencoder, en donde el encoder se forma de la forma descrita anteriormente y el decoder se fabrica utilizando los pesos obtenidos en el encoder. Luego los pesos de toda la red son refinados usando backpropagation sobre el autoencoder, obteniendo el error al comparar los resultados obtenidos de codificar y decodificar datos con el input original.

Luego de entrenar la red, el decoder es descartado y el output del cuello de botella es usado directamente como reductor de dimensionalidad del input. Igualmente el decoder es útil para volver a obtener los datos originales (o al menos una aproximación de estos).

finding such initial weights requires a very different type of algorithm that learns one layer of features at a time.

The first layer of the model would be the input, in this case the image. The second layer on would be hidden layers, representing the input image at a lower dimensionality. Each set of weights are learned individually, so in the first step, W_1 is learned and once W_1 is learned, the data is then all stored as the result of the first transfer and W_2 is learned, and so on.

Restricted Boltzman Machines are [generative models](#), so the weights are bidirectional. This means that an activation in layer two can be used to generate a corresponding set of activations in layer one. Hinton uses this principle and “unfolds” the stacked boltzman machines into a neural network. The weights between the layers are effectively re-used to map the data back to its original dimensionality. This process is then the input weights for a neural network that can be trained with backpropogation to fine-tune the weights using the input weights as the outputs to the network.

Ejemplos

Curvas

Para este dataset se sabe la verdadera dimensionalidad con la que se generaron los datos, y la relación entre la intensidad de los pixeles y los 6 números usados para generarlos es altamente no lineal.

El autoencoder consiste de un encoder con capas de tamaño (28 28)-400-200-100- 50-25-6 y un decoder simétrico.

El autoencoder descubrió como convertir cada imagen de 784 pixeles a 6 números reales que permiten una reconstrucción casi perfecta.

Digitos escritos

784-1000-500-250-30 autoencoder

Caras

625-2000-1000-500-30

De un set de caras conocido. El autoencoder da resultados claramente mejores que PCA.

Documentos

Representamos varias notas de un diario como un vector de probabilidades de las 2000 palabras más comunes (dentro de los artículos dados), y entrenamos un autoencoder 2000-500-250-125-10 en la mitad de estas notas.

Cuando el coseno del ángulo entre los dos “códigos” fue usado para medir la similitud, el autoencoder obtuvo resultados claramente mejores que latent semantic analysis (LSA), un método muy conocido para obtener datos de documentos basado en PCA.

VER CLUSTERS

Clasificación números

Un autoencoder dos-dimensional produjo una mejor visualización de los datos que las primeras dos componentes principales.

Preentrenando capa por capa puede ser usado para clasificación y regresión. En un dataset muy conocido de dígitos escritos a mano, el mejor resultado reportado tenía 1.6% de error para backpropagation iniciado aleatoriamente y 1.4% para svms. Luego de entrenar capa por capa en un autoencoder 784-500-500-2000-10 se alcanzó un 1.2%. El Preentrenamiento alcanza generalizar por que se asegura que la mayoría de la información en los pesos se obtenga de modelar las imágenes.

VER CLUSTERS

A shows the 2 dimensional results of standard PCA and B shows the results of a deep neural network. It is clear from this example that what is now being called an autoencoder produces a much more distinct separation of digits when projected into 2 dimensional space than PCA.

Conclusión

El método mostrado anda mejor que PCA. Esto probablemente se debe a que el autoencoder tiene la habilidad de aprender features no-lineales dado que cada una de las capas tiene unidades de activación no lineales, por lo que pueden aprender una representación más compleja del input.

Como dato de color, este paper publicado en 2006 fue uno de los que popularizó la investigación sobre redes neuronales y hoy en día lleva alrededor de 13 mil citaciones.

Método efectivo de inicialización de pesos que permite a las redes autoencoder aprender patrones de baja-dimensionalidad que funcionan mucho mejor que PCA como herramienta para reducir la dimensionalidad de los datos