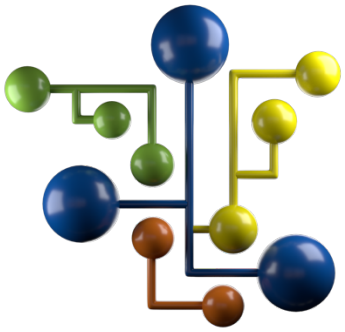


Data Science Academy

www.datascienceacademy.com.br

Deep Learning I

Principais Tipos de Redes Neurais Profundas Parte 1

Autoencoders, Deep Belief Networks e Generative Adversarial Networks

O aprendizado profundo continua em franca evolução em muitos domínios e em muitos problemas principais de aprendizagem de máquina. Aqui estão apenas alguns dos registros de referência que o aprendizado profundo alcançou nos últimos anos:

- Text-to-speech synthesis (Fan et al., Microsoft, Interspeech 2014)
- Language identification (Gonzalez-Dominguez et al., Google, Interspeech 2014)
- Large vocabulary speech recognition (Sak et al., Google, Interspeech 2014)
- Prosody contour prediction (Fernandez et al., IBM, Interspeech 2014)
- Medium vocabulary speech recognition (Geiger et al., Interspeech 2014)
- English-to-French translation (Sutskever et al., Google, NIPS 2014)
- Audio onset detection (Marchi et al., ICASSP 2014)
- Social signal classification (Brueckner & Schulter, ICASSP 2014)
- Arabic handwriting recognition (Bluche et al., DAS 2014)
- TIMIT phoneme recognition (Graves et al., ICASSP 2013)
- Optical character recognition (Breuel et al., ICDAR 2013)
- Image caption generation (Vinyals et al., Google, 2014)
- Video-to-textual description (Donahue et al., 2014)
- Syntactic parsing for natural language processing (Vinyals et al., Google, 2014)
- Photo-real talking heads (Soong and Wang, Microsoft, 2014)

Percebeu algo em comum nestas aplicações de Deep Learning? Sim, são todas bem recentes, a partir de 2013.

Com base nessas realizações, podemos facilmente projetar que o aprendizado profundo vai afetar muitas aplicações na próxima década. Algumas das demonstrações mais impressionantes de aprendizagem profunda aplicada incluem o seguinte:

- Automated image sharpening
- Automating image upscaling
- WaveNet: generating human speech that can imitate anyone’s voice
- WaveNet: generating believable classical music
- Speech reconstruction from silent video
- Generating fonts
- Image autofill for missing regions
- Automated image captioning
- Turning hand-drawn doodles into stylized artwork

Nós provavelmente não perceberemos todas as principais aplicações comerciais até que estejam na nossa frente. Compreender os avanços na arquitetura redes neurais profundas é importante para entender as ideias das aplicações no futuro, principalmente em Inteligência Artificial.

À medida que a pesquisa pressionava o estado da arte a partir de redes feed forward multicamadas, para arquiteturas mais recentes como CNNs e Redes Neurais Recorrentes, a disciplina viu mudanças na forma como as camadas foram configuradas, como os neurônios foram construídos e como conectamos as camadas. Arquiteturas de rede evoluíram para tirar proveito de tipos específicos de dados de entrada.

Avanços em tipos de camada

As camadas se tornaram mais variadas com os diferentes tipos de arquiteturas. Deep Belief Networks (DBNs) demonstraram sucesso com o uso de Boltzmann Machines (RBMs) como camadas em pré-treinamento para criar recursos. As CNNs usaram novos e diferentes tipos de funções de ativação em camadas e mudaram a forma como as camadas são conectadas (de totalmente conectado a patches conectados localmente). As Redes Neurais Recorrentes exploraram o uso de conexões que modelaram melhor o domínio do tempo em dados de série temporal.

Avanços nos tipos de neurônios

As Redes Neurais Recorrentes criaram, especificamente, avanços nos tipos de neurônios (ou unidades) aplicados no trabalho em torno das redes LSTM. Eles introduziram novas unidades específicas para redes neurais recorrentes, como a célula de memória LSTM e as unidades recorrentes (GRUs).

Arquiteturas híbridas

Continuando o tema da correspondência de dados de entrada com o tipo de arquitetura, vimos arquiteturas híbridas surgirem para tipos de dados que tenham tanto tempo quanto dados de imagem envolvidos. Por exemplo, a classificação de objetos em vídeo foi demonstrada com sucesso pela combinação de camadas de CNNs e Redes Neurais Recorrentes em uma única rede híbrida. As arquiteturas de redes neurais híbridas podem nos permitir aproveitar o melhor dos dois mundos em alguns casos.

Vejamos agora, as quatro principais arquiteturas de redes neurais profundas e como usamos as redes menores para construí-las. Embora Deep Learning seja uma área de pesquisa bastante ativa neste momento, podemos listar quatro grandes arquiteturas de redes neurais profundas:

- Unsupervised Pretrained Networks (UPNs)
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks
- Recursive Neural Networks

Duas dessas arquiteturas você encontra com frequência na solução de problemas complexos: Convolutional Neural Networks para modelagem de imagem e Recurrent Neural Networks para sequência modelagem. Mas as arquiteturas vêm evoluindo ou novas alternativas híbridas permitem resolver problemas até então impossíveis!

No grupo de redes pré-treinadas não supervisionadas, encontramos três arquiteturas específicas:

- Autoencoders
- Deep Belief Networks (DBNs)
- Generative Adversarial Networks (GANs)

Vamos analisar as principais características dessas 3 arquiteturas, que serão estudadas no curso Deep Learning II.

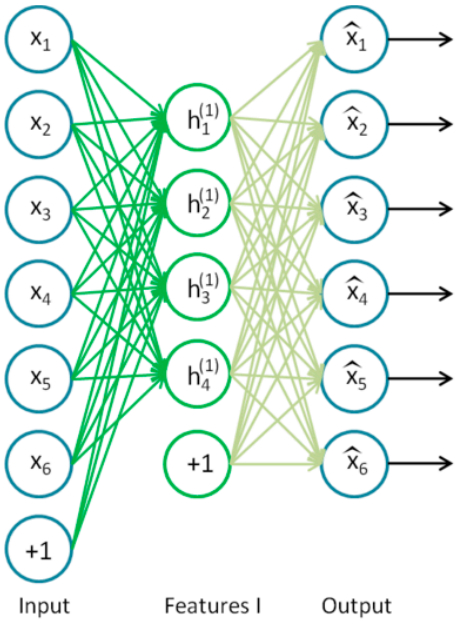
Autoencoders

Usamos Autoencoders para aprender representações compactadas de conjuntos de dados. Normalmente, nós os usamos para reduzir a dimensionalidade de um conjunto de dados. A saída da rede de Autoencoder é uma reconstrução dos dados de entrada da forma mais eficiente.

O Autoencoder é um auto-associador. A finalidade de um auto-associador (AA) é receber na saída a imagem mais precisa da entrada quanto possível. Existem dois tipos de AA - geradoras e sintetizadoras. Uma Máquina de Boltzmann Restrita pertence ao primeiro tipo e um Autoencoder representa o segundo tipo.

Os Autoencoders compartilham uma forte semelhança com redes neurais Multilayer Perceptron em que eles têm uma camada de entrada, camadas ocultas de neurônios e, em seguida, uma camada de saída. A principal diferença a notar entre um diagrama de rede Multilayer Perceptron e um diagrama de Autoencoder é a camada de saída em um Autoencoder, que tem o mesmo número de unidades que a camada de entrada.

Um Autoencoder é uma rede neural com uma camada aberta. Usando o algoritmo de aprendizado não supervisionado e a retropropagação, ele define um valor objetivo igual ao vetor de entrada, ou seja, y = x.



Na imagem acima, o Autoencoder está tentando construir a função h(x) = x. Em outras palavras, ele tenta encontrar uma aproximação de uma função, assegurando que um feedback da rede neural é aproximadamente igual aos valores dos parâmetros de entrada. Para a solução do problema ser não-trivial, o número de neurônios na camada aberta deve ser menor do que a dimensão dos dados de entrada (como na imagem).

Ele permite a compressão de dados, quando o sinal de entrada é passado para a saída da rede. Por exemplo, se o vetor de entrada é um conjunto de níveis de brilho de uma imagem de 10x10 pixels de tamanho (100 características), o número de neurônios da camada oculta é de 50, a rede é forçada a aprender a comprimir a imagem. O requisito h(x) = x significa que, com base nos níveis de ativação dos cinquenta neurônios da camada oculta, a camada de saída é para restaurar 100 pixels da imagem inicial. Essa compressão é possível se existirem interconexões ocultas, correlação característica ou qualquer estrutura. Desta forma, uma operação Autoencoder lembra o método de Análise de Componentes Principais (PCA) no sentido de que os dados de entrada ficam reduzidos. Teremos um capítulo inteiro dedicado aos Autoencoders no curso Deep Learning II, onde também estudaremos a Análise de Componentes Principais (PCA).

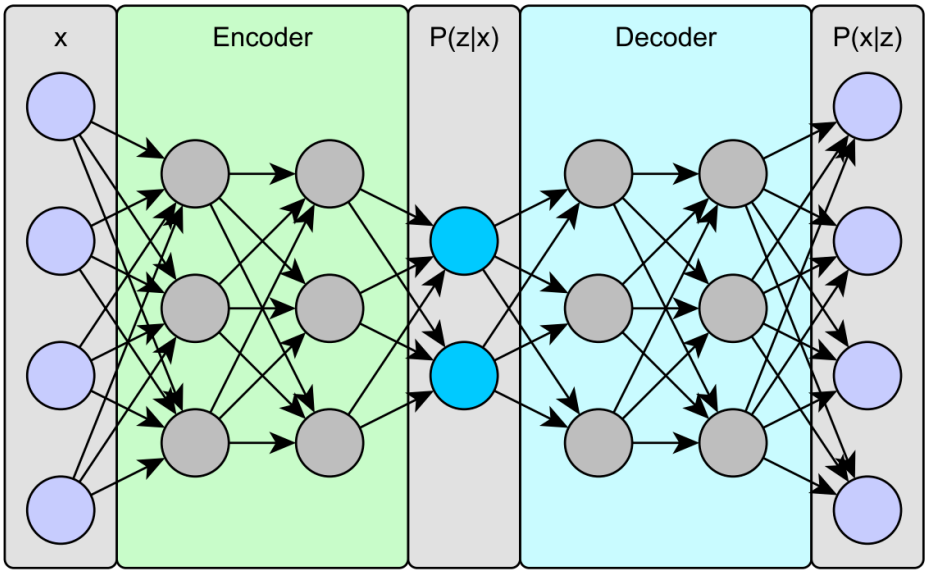
Uma variação do Autoencoder, é o Autoencoder Esparso, que tem um número de neurônios escondidos significativamente maior do que a dimensão de entrada, mas que têm uma ativação esparsa. A ativação esparsa ocorre quando número de neurônios inativos na camada oculta é significativamente maior do que o número de ativos. Se nós descrevermos a espacialidade informalmente, então, um neurônio pode ser considerado ativo se o valor da sua função é próximo de 1. Se uma função sigmóide está em uso, então, o valor do neurônio inativo deve ser próximo de 0 (para a função da tangente hiperbólica o valor deve ser perto de -1).

Existe uma outra variação de um Autoencoder chamado de Denoising Autoencoder (Vincent et al., 2008). Este é o mesmo Autoencoder, mas o seu treinamento tem algumas peculiaridades. Ao treinar esta rede, os dados "corrompidos" de entrada são substituídos por 0. Ao mesmo tempo, há dados "corretos" para comparar com os dados de saída. Desta forma, um Autoencoder pode restaurar os dados danificados.

Os Autoencoders dependem de backpropagation para atualizar seus pesos. A principal diferença entre RBMs e a classe mais geral de Autoencoders é a forma como eles calculam os gradientes.

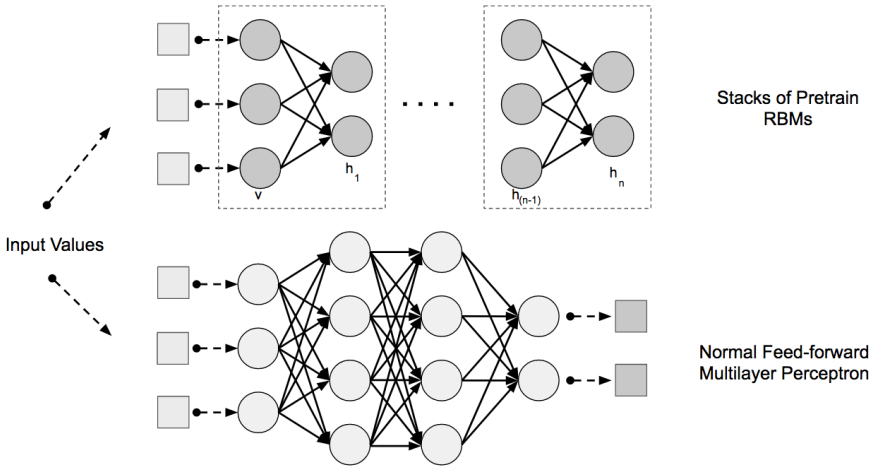
Um tipo mais recente de modelo de Autoencoder é o Variational Autoencoder (VAE) introduzido por Kingma e Welling21 (figura abaixo). O VAE é semelhante aos Autoencoders de compressão e desativação, na medida em que são todos treinados de forma não supervisionada para reconstruir entradas.

No entanto, os mecanismos que os VAEs usam para realizar treinamento são bastante diferentes. Em um Autoencoder de compressão / desativação, as ativações são mapeadas para ativações em todas as camadas, como em uma rede neural padrão; comparativamente, um VAE usa uma abordagem probabilística para o passe para a frente.



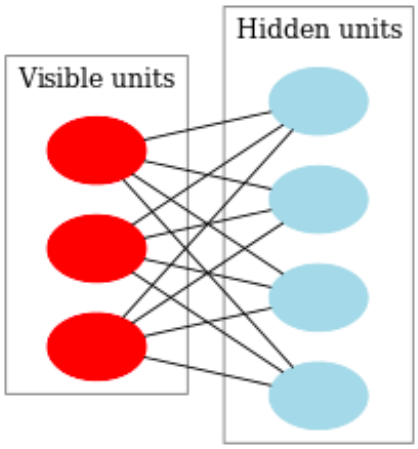
Deep Belief Networks

DBNs são compostos de camadas de Máquinas Boltzmann Restritas (RBMs) para a fase pré-treinamento e, em seguida, uma rede de alimentação para a fase de ajuste fino. A figura abaixo mostra a arquitetura de rede de um DBN.



As Máquina de Boltzmann Restrita (RBM) começaram a ser usadas com as redes neurais recorrentes com um feedback que era muito difícil de se treinar. Devido a esta dificuldade de aprendizagem, mais modelos recorrentes restritos apareceram, assim, algoritmos de aprendizagem mais simples puderam ser aplicados. Uma rede neural de Hopfield era um desses tais modelos. John Hopfield foi a pessoa que introduziu o conceito da energia de rede, sendo comparada a dinâmica das redes neurais com a termodinâmica.

O próximo passo no caminho para uma RBM foram as máquinas regulares de Boltzmann. Elas diferem de uma rede de Hopfield por ter natureza estocástica e seus neurônios são divididos em dois grupos que descrevem estados visíveis e ocultos (semelhante a um modelo oculto de Markov). Uma máquina de Boltzmann restrita é diferente de uma regular na ausência de conexões entre os neurônios de uma camada. Abaixo a estrutura de uma RBM:



A peculiaridade deste modelo é que nos atuais estados de neurônios de um grupo, os estados de neurônios de um outro grupo vão ser independentes um do outro.

Uma RBM é interpretada semelhante a um modelo oculto de Markov. Temos um número de estados que podemos observar (neurônios visíveis) e um número de estados ocultos que não podemos ver diretamente (neurônios ocultos). Nós podemos chegar a uma conclusão baseada na probabilidade sobre os estados ocultos que confiam nos estados que podemos observar. Depois que tal modelo foi treinado, nós temos uma oportunidade para tirar conclusões sobre os estados visíveis sabendo que as ocultas estão seguindo o teorema de Bayes. Isso permite gerar dados a partir da distribuição da probabilidade usada para treinar o modelo.

Dessa forma, podemos formular o objetivo de treinamento de um modelo: os parâmetros do modelo devem ser ajustados de maneira que o vetor restaurado a partir do estado inicial seja mais próximo que o original. Um vetor restaurado é um vetor recebido por uma inferência probabilística de estados ocultos, os quais, por sua vez, foram recebidos por uma inferência probabilística de estados visíveis, ou seja, a partir do vetor original.

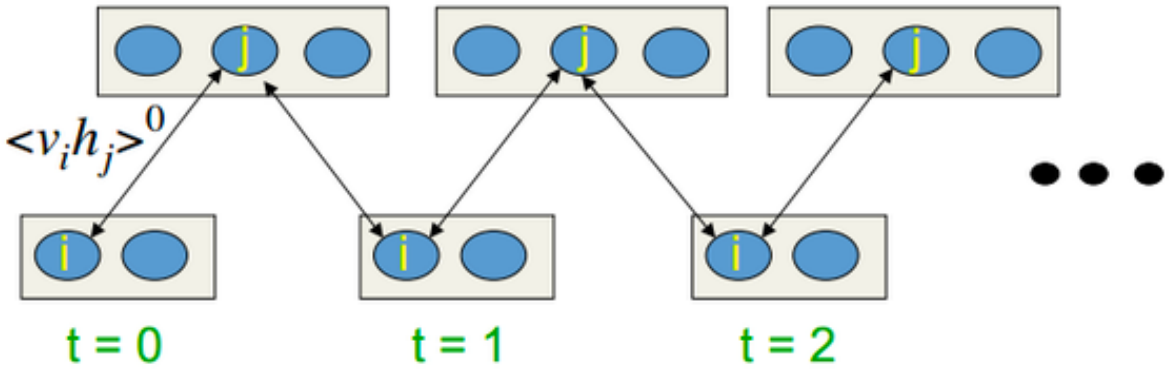
Uma das inovações dos DBNs, foi o Algoritmo de aprendizagem Divergência Comparativa CD-k, inventado pelo professor Hinton em 2002 e é extremamente simples. A ideia principal é que os valores de expectativa matemática sejam substituídos por valores definidos. A noção do processo de amostragem de Gibbs.

O CD-k se parece com:

- O estado dos neurônios visíveis é definido como sendo igual ao padrão de entrada;
- As probabilidades dos estados das camadas ocultas são desenhadas;
- Cada neurônio da camada escondida é atribuído o estado "1" com probabilidade igual ao seu estado atual;

- Probabilidades dos estados de camadas visíveis são desenhados com base na camada oculta;
- Se a iteração atual é menor do que k, então, retorna para a etapa 2;
- As probabilidades dos estados das camadas ocultas são desenhadas;

Nas palestras de Hinton, ela se parece com:



Em outras palavras, quanto maior o tempo que nós fazemos amostragem, mais preciso será o gradiente. O professor afirma que o resultado recebido para CD-1, ou seja, apenas uma iteração de amostragem, já está bom.

Visite o endereço abaixo, para um excelente artigo de introdução a Deep Learning:

https://www.toptal.com/machine-learning/an-introduction-to-deep-learning-from-perceptrons-to-deep-networks

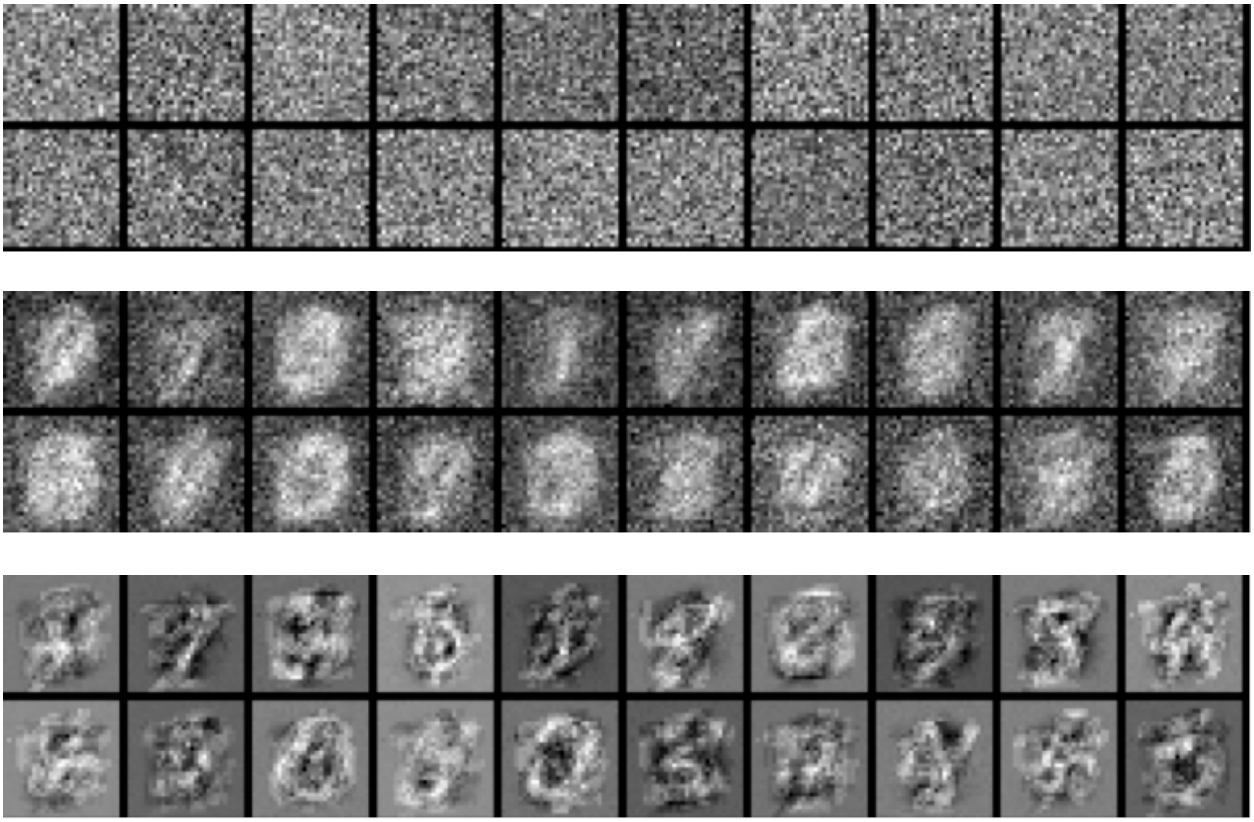
Usamos RBMs para extrair recursos de nível superior dos vetores de entrada brutos. Para isso, queremos definir os estados e pesos da unidade oculta de modo que quando mostramos ao RBM um registro de entrada e pedimos ao RBM para reconstruir a gravação do registro, ele gera algo muito próximo ao vetor de entrada original. Hinton fala sobre esse efeito em termos de como as máquinas "sonham com dados".

O propósito fundamental dos RBMs no contexto da aprendizagem profunda e DBNs é aprender esses recursos de nível superior de um conjunto de dados em uma forma de treinamento não supervisionado. Descobriu-se que poderíamos treinar melhores redes neurais ao permitir que RBMs aprendessem progressivamente recursos de nível superior usando os recursos aprendidos a partir de uma camada de pré-treinamento RBM de nível inferior como a entrada para uma camada de pré-treinamento de RBM de nível superior.

Aprender esses recursos de forma não supervisionada é considerada a fase de pré-treinamento das DBNs. Cada camada escondida do RBM na fase de pré-treinamento aprende

progressivamente recursos mais complexos a partir da distribuição dos dados. Esses recursos de ordem superior são combinados progressivamente em formas não-lineares para fazer engenharia de recursos de forma automatizada.

Para entender visualmente a construção de características com camadas de RRBMs, veja as figuras abaixo, que mostram a progressão dos renderizações de ativação em um RBM à medida que ele aprende os dígitos MNIST.



Podemos ver como uma camada de RBM extrai fragmentos de dígitos à medida que treina. Esses recursos são então combinados em camadas de nível superior para criar recursos progressivamente mais complexos (e não-lineares).

Como esses dados brutos são modelados no processo de modelagem generativa com cada camada do RBM, o sistema é capaz de extrair recursos cada vez mais elevados dos dados de entrada bruta produzidos por nosso processo de vetorização dos dados de entrada. Esses recursos avançam através dessas camadas de RBMs em uma direção, produzindo recursos mais elaborados na camada superior.

Em seguida, usamos essas camadas de recursos como os pesos iniciais em uma rede neural de feed-forward orientada para backpropagation tradicional.

Esses valores de inicialização ajudam o algoritmo de treinamento a orientar os parâmetros da rede neural tradicional para melhores regiões do espaço de busca de parâmetros. Esta fase é conhecida como a fase de ajuste fino de DBNs.

Na fase de ajuste fino de um DBN, usamos uma transposição normal com uma menor taxa de aprendizado para fazer um "backpropagation" suave. Consideramos que a fase de pré-treinamento é uma busca geral do espaço de parâmetros de uma forma não supervisionada baseada em dados brutos. Em contraste, a fase de ajuste fino é sobre a especialização da rede e seus recursos para a tarefa que realmente nos interessa (como a classificação).

A fase de pré-treinamento com RBM aprende os recursos de ordem superior dos dados, que usamos como bons valores iniciais de inicialização para nossa rede de feed-forward. Queremos levar esses pesos e ajustá-los um pouco mais para encontrar bons valores para o nosso modelo final de rede neural.

O objetivo normal de uma rede profunda é aprender um conjunto de recursos. A primeira camada de uma rede profunda aprende como reconstruir o conjunto de dados original. As camadas subsequentes aprendem a reconstruir as distribuições de probabilidade das ativações da camada anterior. A camada de saída de uma rede neural está ligada ao objetivo geral. Isso geralmente é regressão logística, com o número de recursos igual ao número de entradas da camada final e o número de saídas igual ao número de classes.

Generative Adversarial Networks (GANs)

Modelos de GANs mostraram ser bastante habilidosos para sintetizar novas imagens com base em outras imagens de treinamento. Podemos ampliar esse conceito para modelar outros domínios, como o seguinte:

- Som
- Video
- Gerando imagens de descrições de texto

Os GANs são um exemplo de uma rede que usa aprendizado sem supervisão para treinar dois modelos em paralelo. Um aspecto-chave dos GANs (e modelos generativos em geral) é como eles usam uma contagem de parâmetros que é significativamente menor do que o normal em relação à quantidade de dados nos quais estamos treinando a rede. A rede é forçada a representar eficientemente os dados de treinamento, tornando-o mais eficaz na geração de dados semelhantes aos dados de treinamento.

Se tivéssemos um grande corpus de imagens de treinamento (como o conjunto de dados ImageNet), poderíamos construir uma rede neural generativa que dê saída a imagens (em oposição a classificações). Consideramos que essas imagens de saída geradas são amostras do

modelo. O modelo generativo em GANs gera essas imagens enquanto uma rede secundária de "discriminadores" tenta classificar essas imagens geradas.

Esta rede discriminadora secundária tenta classificar as imagens de saída como reais ou sintéticas. Ao treinar GANs, queremos atualizar os parâmetros de forma que a rede gerará imagens de saída mais precisas com base nos dados de treinamento. O objetivo aqui é tornar as imagens bastante realistas para que a rede discriminadora seja enganada ao ponto de não distinguir a diferença entre os dados de entrada reais e sintéticos.

Um exemplo de representação de modelo eficiente em GANs é como eles geralmente possuem cerca de 100 milhões de parâmetros ao modelar um conjunto de dados como o ImageNet. No decorrer do treinamento, um conjunto de dados de entrada como o ImageNet (200 GB) torna-se próximo de 100 MB de parâmetros. Este processo de aprendizagem tenta encontrar a maneira mais eficiente de representar os recursos nos dados, como grupos semelhantes de pixels, bordas e outros padrões (como veremos mais detalhadamente em "Redes Neurais Convolucionais (CNNs)"). Usaremos o ImageNet também!

Ao modelar imagens, a rede discriminadora normalmente é uma CNN padrão. Usando uma rede neural secundária, pois a rede discriminadora permite que o GAN treine ambas as redes em paralelo de forma não supervisionada. Essas redes de discriminadores tomam imagens como entrada e, em seguida, produzem uma classificação.

O gradiente da saída da rede discriminadora em relação aos dados de entrada sintética indica como fazer pequenas alterações nos dados sintéticos para torná-lo mais realista.

A rede generativa em GANs gera dados (ou imagens) com um tipo especial de camada chamado camada deconvolucional. Durante o treinamento, utilizamos a reprodução de backpropagation em ambas as redes para atualizar os parâmetros da rede geradora para gerar imagens de saída mais realistas. O objetivo aqui é atualizar os parâmetros da rede geradora para o ponto em que a rede discriminante é suficientemente "enganada" pela rede geradora porque a saída é tão realista quanto as imagens reais dos dados do treino.

Uma variante de GANs é a Deep Convolutional Generative Adversarial Network (DCGAN). A figura abaixo retrata imagens de quartos gerados a partir de um DCGAN.

Sim, isso mesmo! Essas imagens foram geradas automaticamente por um modelo de Deep Learning, chamado Deep Convolutional Generative Adversarial Network.

Essa rede recebe números aleatórios (de uma distribuição uniforme) e gera uma imagem do modelo de rede como saída. À medida que os números aleatórios de entrada mudam, vemos o DCGAN gerar diferentes tipos de imagens. Simplesmente fantástico!

Os GANs se concentram em tentar classificar os registros de treinamento como sendo da distribuição do modelo ou da distribuição real. Quando o modelo discriminador faz uma predição em que há uma diferença entre as duas distribuições, a rede geradora ajusta seus parâmetros. Eventualmente, o gerador converge em parâmetros que reproduzem a distribuição de dados reais e o discriminador não consegue detectar a diferença.

Com os Variational Autoencoders (VAEs), resolvemos esse mesmo problema com modelos gráficos probabilísticos para reconstruir a entrada de forma não supervisionada. Os VAEs tentam maximizar um limite inferior na probabilidade de registro dos dados de modo que as imagens geradas parecem cada vez mais reais.

Outra diferença interessante entre GANs e VAEs é como as imagens são geradas. Com GANs básicos, a imagem é gerada com código arbitrário e não temos como gerar uma imagem com recursos específicos. Os VAE, em contraste, têm um esquema de codificação/decodificação específico com o qual podemos comparar a imagem gerada com a imagem original. Isso nos dá o efeito colateral de poder codificar para tipos específicos de imagens a serem geradas.