Materiais:

Q: Onde encontro os materiais de apoio e os arquivos notebooks para execução dos exercícios?

R: O material de apoio consta no ícone de folha a4 ao lado do título da disciplina. O material prático da disciplina se encontrará como uma aula extra também no menu ao lado esquerdo da Sala Virtual.

Q: Há material prático para esta disciplina?

R: Sim, o professor Martin elaborou uma aula extra para aplicar os conceitos vistos em aula de forma prática utilizando alguns algoritmos em Python. O material encontra-se disponível no icone A4 ao lado do título da disciplina, na aula extra.

Ferramentas:

Q: Como usar os arquivos .py (utils e visualization) no Colab?

R: você não vai conseguir executar os arquivos .py DIRETAMENTE no Colab ou Jupyter como podemos fazer por exemplo com a IDE Spyder. O professor Martin utiliza os arquivos .py de forma a persistir algumas funções, dessa maneira, ele IMPORTA os arquivos .py (ao executar as linhas de comando: # implementação local import utils import visualization as vis). Dessa maneira ele não precisa "sujar" o notebook com códigos que são de funções auxiliares. Espero ter lhe ajudado, qualquer coisa por favor me avise.

Conceitos e Exercícios:

Q: Qual a diferença em relação as suas funções, de uma camada convolucional e de uma camada densa?

R: A principal diferença entre a camada Convolucional e a camada Densa é que a Camada Convolucional usa menos parâmetros, forçando os valores de entrada a compartilhar os parâmetros.

A camada densa usa uma operação linear, o que significa que cada saída é formada pela função baseada em cada entrada. Cada entrada usa a função e a rede deve aprender cada relação da entrada com a saída. Como resultado, aparecem n * m conexões (ou pesos) onde n denota o número de entradas e, m denota o número de saídas.

A camada convolucional utiliza um filtro para operar a operação de convolução que, na maioria das vezes, é de tamanho pequeno. Uma saída das camadas de convolução é formada por apenas um pequeno tamanho de entradas que depende do tamanho do filtro e os pesos são compartilhados por todos os pixels. Ou seja, a saída é construída usando os mesmos coeficientes para todos os pixels, usando os pixels vizinhos como entrada.

Q: Se o gradiente estocástico consegue escapar do mínimo local devido ao seu componente estocástico, porque o mesmo não ocorre com o ponto de sela? Em teoria ele poderia não passar pelas partes com ponto de sela em algumas iterações, assim como ele não passa pelo mínimo local

R: Esse raciocínio faz completo sentido. O processo de otimização de funções é um tópico muito complexo, porém é de extrema relevânicia para Deep Learning. Devido à função de custo e arquitetura não-linear das redes profundas, estamos lidando com problemas não-convexos de otimização. Dentro desta classe de problemas, existem empecilhos para encontrar soluções ótimas, pois a paisagem da função de custo acaba por ser dominada por mínimos locais e pontos de sela, por exemplo.

Primeiramente, note que, independente de estarmos usando Gradiente Descendente (GD) ou Gradiente Descendente Estocástico (SGD), a função de otimização (não-convexa) é a mesma, e os gradientes são uma propriedade desta função. Quando usamos SGD, temos menos chances de entrar em posições ruins de um ponto de sela, pois nossa "caminhada" é muito mais irregular. Contudo, ainda é possível cair em uma região "rasa" (gradientes praticamente zerados) de um ponto de sela.

Mesmo que nossa caminhada seja mais aleatória, as chances de escapar desta região diminuem significativamente. Outro ponto a se considerar é que, para problemas de otimização altamente dimensionais (que é o caso de otimização de redes neurais profundas), alguns estudos apontam que a probabilidade de encontrar pontos de sela é maior do que o de mínimos locais (esta afirmação é feita com base na análise das derivadas de segunda ordem em posições onde o gradiente de primeira ordem da função é zero).

Além disso, existem diferentes tipos de pontos de sela, com diferentes níveis de "complexidade". Um exemplo de ponto de sela complexo é a "Monkey Saddle". Esta classificação depende principalmente da intensidade da curvatura da região. Intuitivamente, quanto menos curvada for a região do ponto de sela, maior será o tamanho da região com gradientes fracos ou zerados. Isto influenciará na velocidade de "escape".

Existem trabalhos que tentam provar, teoricamente, quanto tempo o (S)GD leva para sair de um ponto de sela. Apesar de chegarem à conclusão de que é possível sair de pontos de sela utilizando somente SGD, é importante ressaltar que: (1) alguns destes estudos trabalham com restrições e premissas quanto à curvatura da paisagem da função de custo e quanto ao tipo de ponto de sela; (2) outros trabalhos propõem modificações no GD para evitar pontos estacionários de primeira ordem (e.g., pontos de sela e mínimos locais) como, por exemplo, adicionar pequenas perturbações nos gradientes dependendo dos valores dos gradientes locais de primeira e segunda ordem); (3) geralmente estamos lidando com tempo e recursos computacionais limitados, então sempre é interessante acelerar o processo de otimizacão ao máximo possível.

Como visto em aula, uma outra solução para lidar com pontos de sela seria adicionar informação de velocidade nos gradientes. Desta forma, mesmo que o processo de otimização nos leve à um ponto com gradientes fracos, ainda temos um histórico acumulado que nos ajuda a sair deste ponto. Esta solução também não é perfeita mas, na prática (e na teoria também, em alguns casos), os melhores resultados são encontrados usando técnicas como Momentum.

Alguns links para referências sobre alguns dos artigos comentados, assim como algumas leituras mais superficiais em formato de blog. Estas referências discutem a origem de pontos de sela, bem como mecanismos de otimização para lidar com elas. Bons estudos!REFERÊNCIAS:- Du, Simon S., et al. "Gradient descent can take exponential time to escape saddle points." Advances in neural information processing systems 30 (2017). https://arxiv.org/abs/1705.10412- Daneshmand, Hadi, et al. "Escaping saddles with stochastic gradients." International Conference on Machine Learning. PMLR,

2018. https://arxiv.org/abs/1803.05999- Staib, Matthew, et al. "Escaping saddle points with adaptive gradient methods." International Conference on Machine Learning. PMLR, 2019. https://arxiv.org/abs/1901.09149- Avdiukhin, Dmitrii, and Grigory Yaroslavtsev. "Escaping Saddle Points with Compressed SGD." Advances in Neural Information Processing Systems 34 (2021). https://arxiv.org/abs/2105.10090- Ziyin, Liu, et al. "SGD May Never Escape Saddle Points." arXiv preprint arXiv:2107.11774 (2021). https://arxiv.org/abs/2107.11774- Wang, Jun-Kun, Chi-Heng Lin, and Jacob Abernethy. "Escaping saddle points faster with stochastic momentum." arXiv preprint arXiv:2106.02985 (2021). https://arxiv.org/abs/2106.02985- IFT 6169: Theoretical principles for deep learning - SGD Escapes Saddle Points: https://mitliagkas.github.io/ift6085-2019/ift-6085-bonus-lecture-saddle-points-notes.pdf- http://www.offconvex.org/2016/03/22/saddlepoints/- https://bair.berkeley.edu/blog/2017/08/31/saddle-efficiency/- https://davidlibland.github.io/posts/2018-11-10-saddle-points-and-sdg.html

Q: Não entendi o conteito de campo receptivo (Aula 05 tempo 28:27).

R: o campo receptivo nada mais é que uma pequena janela nos pixels da imagem de entrada enviada para o neurônio oculto. É definido pelo tamanho do filtro de uma camada dentro da rede neural de convolução. Esses Campos Receptivos se tornam mais e mais amplos ao longo das convoluções (para se ter uma informação mais global).

O campo receptivo dessa forma, aumenta conforme a profundidade da rede, mesmo que os tamanhos de kernel das camadas convolucionais sejam iguais. Por exemplo, se a primeira e a segunda camada forem compostas por filtros com kernel 3x3, o campo receptivo efetivo da segunda camada será 5x5, pois cada "célula" processada pela primeira camada representa uma região de 3x3).

Assim, a Rede Neural consegue preservar as correlações entre pixels vizinhos da imagem, pois atua sobre esses campos receptivos locais realizando o processo de convolução. Dessa forma, consegue-se reduzir a sensibilidade à translação, rotação e distorção das imagens.

Q: No vídeo da parte 1, minuto 43:04 foi comentado sobre retropropagar os gradientes, para ajustar o modelo. E em seguida no vídeo 4 da aula 4, também vimos sobre o processo de transferência e que algumas camadas são congeladas no transfer learning. A dúvida é: Quando a gente fizer a retropropagação dos gradientes, faremos apenas das camadas não congeladas ou faremos para todas as camadas?

R: Quando dizemos que uma camada está "congelada", isso significa que os pesos desta camada não sofrerão atualizações. Se, por algum motivo, houverem camadas não-congeladas antes da camada congelada (e.g., NC -> C -> NC, onde C = congelada, NC = não-congelada), temos que retropropagar os gradientes até que cheguem em todas as camadas não-congeladas (mas isto não implica em atualizar as camadas congeladas, estamos apenas usando os resultados de suas operações e a derivada da função para saber a contribuição de cada um dos pesos da camada não-congelada). Dito isto, não é comum congelar somente camadas intermediárias do modelo, mas sim congelar as camadas do início do processamento até o fim (e.g., C -> C -> NC ,ou C -> NC -> NC).

Q: A técnica de Transfrer Learning pode e é utilizada em outras aplicações que não sejam relacionadas a visão computacional? Por exemplo, ploblemas mais comuns de Machine Learning que possuem entrada dados tabulares.

R: Apesar de ser mais usado nessa área da Mineração de dados, técnicas de transfer learning são sim usadas em dados tabulares como dados escolares. É possível pesquisar em

bibliotecas digitais como IEEE por esse tema central transfer learning e ir aprofundando em outras aplicações também.

** FAQ gerado com base em comentários até o dia 30/11/2022.*