Conexões resíduais

Created	@May 19, 2025 7:44 PM
Tags	

Contexto e Motivação para Conexões Residuais

Anteriormente, vimos que MLPs (Perceptrons Multicamadas) são eficazes para problemas de regressão e classificação, e que camadas convolucionais são adequadas para trabalhar com imagens. Aumentar a profundidade de uma rede neural geralmente leva a melhores resultados. No entanto, redes neurais mais profundas enfrentam problemas como o gradiente que se dissipa e o gradiente explosivo. Esse fenômeno, que causa a piora da acurácia em redes muito profundas, não é completamente compreendido, mas uma hipótese aceita é a do "gradiente quebrado" (shattered gradient), onde pequenas mudanças nos pesos geram alterações erráticas nas camadas subsequentes.

O que são Conexões Residuais (Skip Connections)

Modificações aditivas, conhecidas como **conexões residuais** ou **skip connections**, permitem treinar redes mais profundas. Em vez do processamento sequencial tradicional onde a saída de uma camada é simplesmente a entrada da próxima ($h_2 = f_2(f_1(x, \theta_1), \theta_2)$), as conexões residuais criam "caminhos alternativos". Elas funcionam adicionando a entrada de um bloco de camadas à sua saída, por exemplo: $h_2 = h_1 + f_2[h_1, \theta_2]$.

Esses "ramos no grafo computacional" adicionam uma mudança à entrada. Cada mudança aditiva é conhecida como **residual block** ou **residual layer**. Uma restrição dessa abordagem é que a saída da função residual (f_i) deve ter o mesmo tamanho da entrada (h_{i-1}) para que a adição possa ocorrer.

Ao "abrir" as expressões com conexões residuais, percebe-se que existem múltiplos caminhos desde a entrada até a saída. Por exemplo, com 4 blocos residuais, a entrada inicial tem 8 caminhos para a saída. Isso contrasta com o processamento sequencial, onde há apenas um caminho direto. A derivada da saída em relação a um parâmetro de uma camada inicial (como f_1) agora inclui

múltiplos termos devido a todos os caminhos que a informação percorre, tornando a propagação do gradiente mais robusta.

Em um bloco residual, a função $f_i[x]$ tecnicamente pode ser uma camada qualquer, mas modificações práticas podem melhorar o resultado. Uma ilustração típica mostra a conexão residual somando a entrada após uma transformação linear e uma função de ativação não-linear. No entanto, inverter a ordem da ativação e da transformação linear permite que a representação sofra adições e subtrações, embora possa surgir um problema se a entrada for toda negativa. Uma solução para isso é iniciar a rede com uma transformação linear antes de aplicar os blocos residuais. É possível incluir mais de uma transformação dentro do mesmo bloco residual.

Problemas de Variância e Batch Normalization

Ao adicionar conexões residuais, podemos dobrar a profundidade das redes convolucionais e manter o aumento de performance. No entanto, ainda existem fenômenos que limitam a profundidade, relacionados à variância das ativações em redes com skip connections.

A inicialização He visa manter a variância de uma variável aleatória constante após uma transformação linear seguida por ReLU. Porém, com a adição da skip connection ($h_{out} = h_{in} + f(h_{in}))$), a variância da saída se comporta de maneira diferente. Assumindo que a entrada (X) e a saída da função residual (Y) não são correlacionadas, a variância da soma ($Var(X+Y)$) é a soma das variâncias ($Var(X) + Var(Y)$). Como a variância da saída da função residual ($f(h_{in})$) tende a ter uma variância similar à da entrada (h_{in}) devido à inicialização He, a variância da saída de um bloco residual (h_{out}) tende a dobrar em comparação com a entrada (h_{in}).

Em uma rede com muitas camadas, essa duplicação da variância em cada bloco residual pode rapidamente exceder a precisão de ponto flutuante, levando à explosão das ativações.

A solução para esse problema é usar **Batch Normalization**.

Como funciona o Batch Normalization

Batch Normalization é uma normalização aplicada às ativações em camadas ocultas da rede. Ela desloca e reescala a média e o desvio padrão dos mini-

batches para valores que são aprendidos durante o treinamento. Para funcionar, é necessário utilizar mini-batchs com tamanho maior que 1.

O processo envolve os seguintes passos:

1. Calcular a média empírica (μ_h) e o desvio padrão empírico (σ_h) do mini-batch atual. Essas quantidades são escalares.
2. Padronizar as ativações para que tenham média 0 e desvio padrão 1: $h' = \frac{h_i - \mu_h}{\sigma_h + \epsilon}$, onde ϵ é uma pequena constante para evitar divisão por zero.
3. Reescalar as ativações padronizadas por um parâmetro γ (aprendido) e deslocá-las por um parâmetro β (aprendido): $\hat{h} = \gamma h' + \beta$.

Em uma camada densa, existe um γ, um β, um μ e um σ para cada unidade oculta. Apenas γ e β são parâmetros treináveis, enquanto μ e σ são calculados para cada mini-batch. γ e β são geralmente inicializados com 1 e 0, respectivamente. Em uma camada convolucional, existe um γ, um β, um μ e um σ por canal.

Durante o treinamento, a média e o desvio padrão são calculados para cada mini-batch e usados para padronizar os dados. Em inferência, quando não há mini-batchs, é necessário ter uma média e um desvio padrão para atualizar os dados. Isso é feito mantendo uma **média móvel** (ou média exponencial móvel) das médias e desvios padrão calculados durante o treino.

Benefícios da Combinação de Skip Connections e Batch Norm

Com a Batch Norm ativa, é possível treinar redes muito mais profundas sem a explosão ou dissipação do gradiente. A Batch Norm torna a rede invariante a mudanças de escala nos parâmetros. Se os parâmetros dobrarem, as ativações e a variância também dobram, mas a primeira etapa da Batch Norm compensa esse aumento. Isso resulta em um forward mais estável, permite o uso de taxas de aprendizado mais elevadas, e regulariza o processo de treinamento.

Usando skip connections e batch norm em conjunto, as redes convolucionais podem ter aplicações em praticamente todas as áreas de visão computacional. Elas permitem o treinamento de redes com cerca de 1000 camadas ocultas e

ainda são o padrão para a maioria das atividades de visão computacional, apesar do sucesso recente de transformers.

Arquiteturas Baseadas em Conexões Residuais

- **ResNet**: Redes convolucionais compostas por blocos residuais. Cada bloco foi projetado por "tentativa e erro". ResNets apresentam bons resultados, mas podem ter um elevado número de parâmetros. O **Bloco Residual com Gargalo (Bottleneck Residual Block)** foi introduzido para diminuir o número de parâmetros treináveis. Ele usa uma convolução 1×1 para reduzir o número de canais, seguida por outras convoluções, e outra convolução 1×1 para aumentar o número de canais de volta ao tamanho original, permitindo a adição residual. ResNet-200, por exemplo, utiliza este bloco e atinge 4.8% de erro no ImageNet.
- **DenseNet**: Em vez de adicionar, concatena os canais da entrada com os canais da saída. Embora menos comum que a ResNet, atinge resultados similares. A representação não é concatenada quando ocorre downsampling.
- **U-Net**: Uma arquitetura encoder-decoder que também utiliza skip connections. Nestas arquiteturas, as skip connections ligam camadas do encoder (que diminuem a resolução espacial) com camadas correspondentes do decoder (que aumentam a resolução espacial), o que é útil para tarefas como segmentação semântica.

As conexões residuais, combinadas com Batch Normalization, foram fundamentais para o desenvolvimento de redes neurais profundas e seu sucesso em diversas tarefas de visão computacional.