**Deep Learning II**

1. **Introdução**

Instalar Anaconda Python

Ambiente Virtual com Pacotes e Versões Específicas

**Exibição de pacotes:**

conda list

**Atualizar Conda (Gerenciador de Pacotes):**

conda update conda

**Atualizar Anaconda:**

conda update anaconda

**Atualizar todos os pacotes:**

conda update -all

**Atualizar pacote específico:**

conda update NOME\_PACOTE

**Informações:**

conda info

**Informações de Versões de Pacote:**

conda search NOME\_PACOTE

conda search numpy

**Criar Ambiente Virtual:**

conda create -n NOME\_AMB\_VIRTUAL python==3.7.2 numpy==1.16.1 scipy

**Selecionar Novo Ambiente Virtual**

conda activate NOME\_AMB\_VIRTUAL

**Sair do Ambiente Virtual Criado**

conda deactivate

Google Colab – Ambiente Virtual Anaconda na nuvem

**Prunining:** Remoção de neurônios e conexões fracas que não trazem um cálculo considerável na rede, assim são removidos para diminuir o processamento, e assim também pode-se evitar o overfting.

**Parâmetros:** Valores dentro do modelo, por exemplo, pesos, baias.

**Hiperparâmetros:** Parâmetros externos ao modelo, exemplo: Número de épocas, learning rating (taxa de aprendizado).

Ténicas de seleção de um modelo de dados:

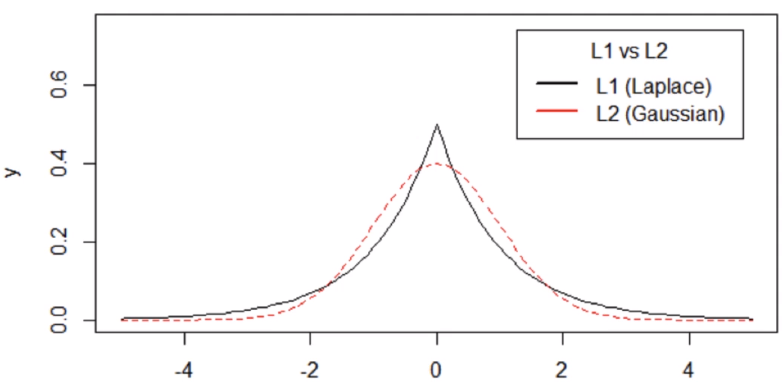
1. **Grid Search Model Selection:** algoritmo de força-bruta e tentativa e erro.
2. **Random Search Model Selection:** Algoritmo que gera valores aleatórios para os hiperparâmetros afim de selecionar o modelo. Há a definição de um valor máximo. Tem uma listagem dos valores já gerados para usar novamente.

**Regularização:** Técnicas para reduzir overfitting, para que o modelo seja o mais generalizado possível, ou seja, quando for apresentado novos conjuntos de dados o modelo seja capaz de responder corretamente.

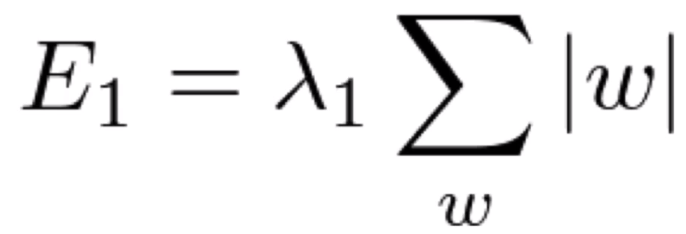
**Algumas técnicas de regularização:** earling stop (para o treinamento quando o erro começa a subir nos dados de validação), descartar neurônios (dropout), pruning, grid Search.

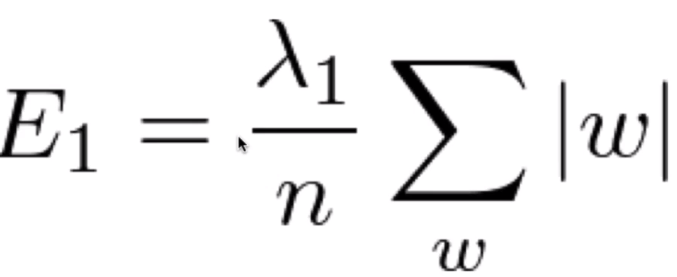
**Regularização L1 e L2**

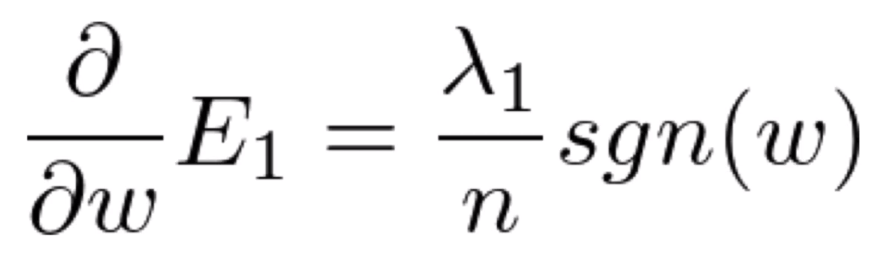
Mais uma camada matemática dentro do backpropagation. Adicionar uma penalidade aos pesos na etapa de treinamento. Se aplicam somente aos pesos, não se aplicam ao baias.



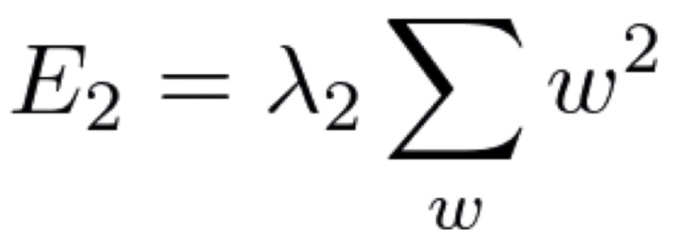
**L1 (Lasso):** Adiciona penalidade equivalente ao valor absoluto da magnitude dos coeficientes. Irá empurrar muitas conexões pra perto de zero, e quando o peso é próximo de zero, ele é descarta da rede.





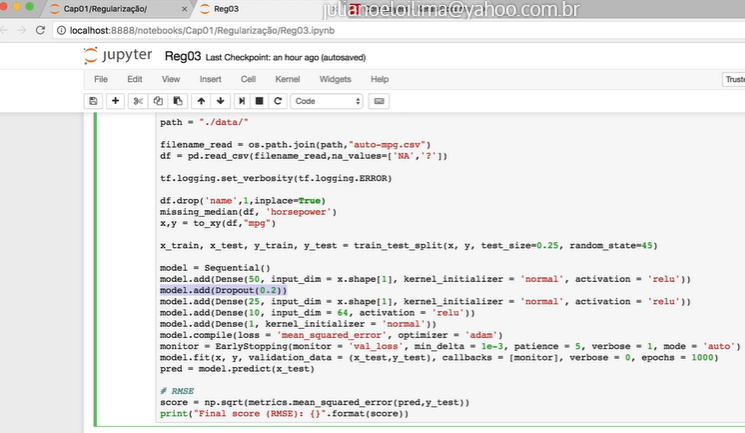


**L2:** Adiciona penalidade equivalente ao quadrado da magnitude dos coeficientes.



****

**Dropout:** Alguns neurônios são escolhidos aletoriamente para serem desconsiderados durante os cálculos**.** Objetivo: evitar overfting**.**

****

**Desconsidera 20% dos neurônios.**

**DataSet Augmentation:** Criar dados falsos (fake) e adicioná-los ao conjunto de treinamento.

1. **Redes Neurais Recorrentes – Parte 1**

**CNN:** Feed Foward – Segue um mesmo fluxo: Input, Camada de Convolução (Camadas Ocultas), Max Pooling, Camadas de Função de Ativação, Camada Densa e Camada de Saída, calcula os erros, realiza o backpropagation e atualiza os pesos.

**RNN:** Redes Neurais Recorrentes. Nas CNNs, os neurônios passam os valores para os próximos neurônios, nas RNNs, os valores podem ser passados para os próprios neurônios, assim nas RNN’s, os neurônios terão características para permitir esses novos processos.

**Recurrent Neural Network:** Ou Redes Neurais Recorrentes Vanila, são o modelo mais simples. Podem tem os gradientes valores muito altos ou muito baixos, o que é um problema para o aprendizado. As redes “Long/Short Term Memory” (LSTM) e “Gated Recurrent Unit” (GRU) são variações da RNN.

1. **Redes Neurais Recorrentes – Parte 2**
2. **Autoencoders**
3. **Generative Adversarial Networks**
4. **Restricted Boltzmann Machines**
5. **Deploy do Modelo de Deep Learning**
6. **Deep Q-Learning**
7. **Avaliação**