Semana 03 – Exercício Semanal

PRO6006 - Métodos de Otimização Não Linear com aplicações em aprendizado de máquina

Nathan Sampaio Santos – 8988661

O exercício em questão visa avaliar as diferentes implementações do método de classificação denominado Regressão Logística. As diferentes implementações são:

- **Batch Gradient Descent**: Utiliza todo o conjunto de dados para calcular o gradiente em cada iteração.
- **Stochastic Gradient Descent**: Utiliza um único exemplo ou um pequeno lote (mini-batch) de dados para calcular o gradiente.
- **Adam (Adaptive Moment Estimation):** Um algoritmo adaptativo que combina as vantagens do Momentum e do RMSprop, ajustando a taxa de aprendizado para cada parâmetro.

Assim, o objetivo deste relatório é analisar as diferentes implementações e seus resultados.

O CONJUNTO DE DADOS

Para testar o uso dos diferentes métodos, utilizamos uma base de diagnóstico de câncer, a qual possui 30 parâmetros listados a seguir que caracterizam quantitativamente um tumor. Além desses parâmetros, também é obtido a coluna *target*, binária, que indica a confirmação ou não de um câncer.

- *mean radius*
- *mean texture*
- *mean perimeter*
- *mean area*
- *mean smoothness*
- *mean compactness*
- *mean concavity*
- *mean concave points*
- *mean symmetry*
- *mean fractal dimension*
- *radius error*
- *texture error*
- *perimeter error*
- *area error*
- *smoothness error*
- *compactness error*
- *concavity error*
- *concave points error*
- *symmetry error*
- *fractal dimension error*
- *worst radius*
- *worst texture*
- *worst perimeter*
- *worst area*
- *worst smoothness*
- *worst compactness*
- *worst concavity*
- *worst concave points*
- *worst symmetry*
- *worst fractal dimension*

Todos esses dados podem ser encontrados publicamente na base de dados UCI ou dentro da biblioteca sklearn do Python.

```
from LogisticRegression      import LogisticRegression
from sklearn.datasets        import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing   import StandardScaler
from sklearn.metrics         import accuracy_score
import matplotlib.pyplot as plt

cancer_data = load_breast_cancer()
X = cancer_data.data
y = cancer_data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=7)

scaler         = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled  = scaler.transform(X_test)
```

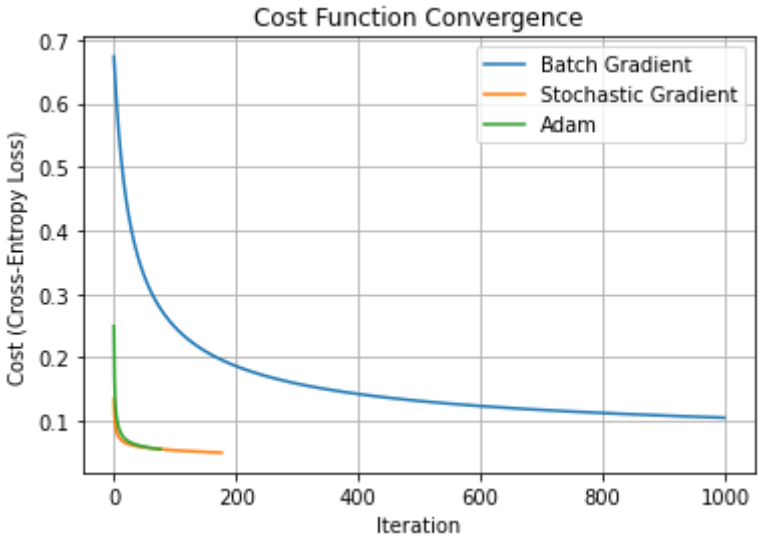
Após obter a base de dados, separamos ela em dois conjuntos: teste e treinamento, com a proporção 20/80, respectivamente. Isso é feito para que a análise da acurácia não seja contaminada por teste em dados que foram utilizados no treinamento do modelo.

Além disso, normalizamos os dados, isto é, subtraímos a média e dividimos pelo desvio padrão, para que todos os coeficientes dos parâmetros tenham pesos equivalentes na busca de uma função que minimize a função custo previamente determinada.

ANÁLISE DE CONVERGÊNCIA

Com os hiperparâmetros ajustados para learning_rate = 0.01, n_iterations_max = 1000 e tol = 0.00001, obtemos o seguinte gráfico de convergência mostrado abaixo. Nele, vemos que os métodos Adam e Stochastic Gradiente são consideravelmente melhores do que o Batch Gradient, de forma que amos convergem para valores de custo muito menores e em menos interações.

Além disso, o método Adam mostrou-se superior a todos, pois cada interação sua é mais veloz do que o Stochastic Gradient, fazendo com que o seu tempo de execução fosse o menor dentre todos os métodos testados.



Vê-se também a performance dos modelos abaixo. Em geral, não há diferença estatística relevante entre os resultados dos diferentes modelos, porém pode-se dizer que dentre todos, o Batch Gradiente Model é o pior, pois ele não chegou a convergir efetivamente, além de ter resultados ligeiramente piores do que o os outros modelos analisados.

```
------------------------------
Batch Gradient Model Performance:
Accuracy: 0.9737
Precision: 0.9610
Final Cost: 0.105220
Batch Gradient Confusion Matrix:
[[37  3]
 [ 0 74]]
```

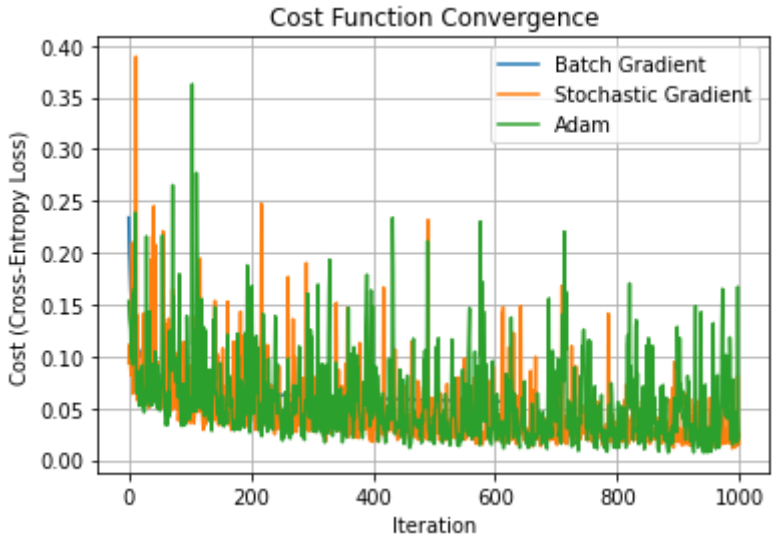
```
Convergiu após 177 iterações.
------------------------------
Stochastic Gradient Model Performance:
Accuracy: 0.9825
Precision: 0.9737
Final Cost: 0.050214
Stochastic Gradient Confusion Matrix:
[[38  2]
 [ 0 74]]
```

```
Convergiu após 77 iterações.
------------------------------
Adam Model Performance:
Accuracy: 0.9825
Precision: 0.9737
Final Cost: 0.055612
Adam Confusion Matrix:
[[38  2]
 [ 0 74]]
```

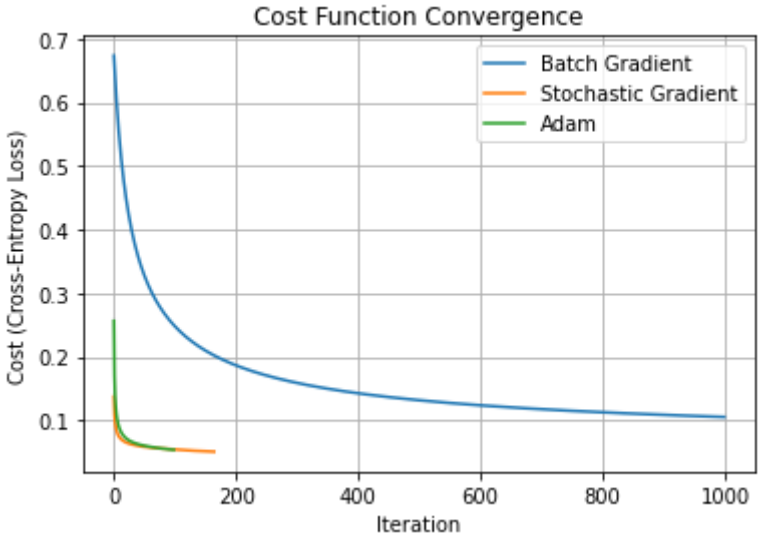
SENSIBILIDADE A HIPERPARÂMETROS

Mantendo-se o modelo da mesma forma

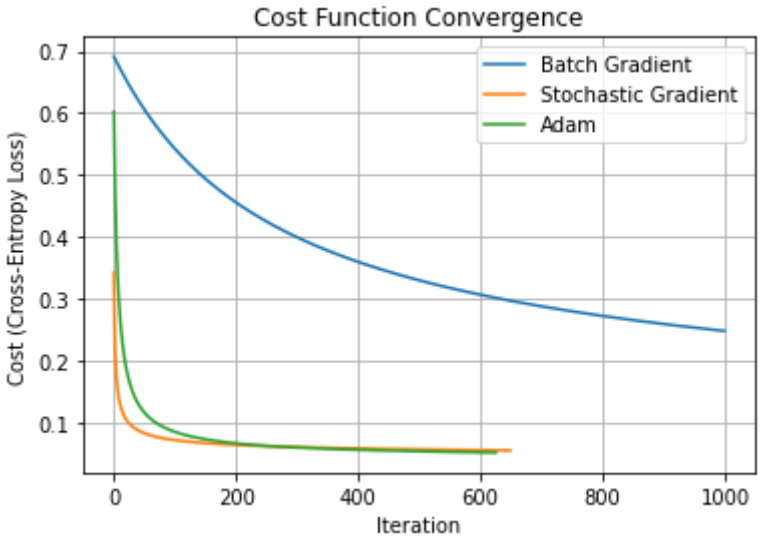
- alpha = 1 (Muito Alta): O modelo diverge. Os passos são tão grandes que oscilam entre um pondo de menor custo e outro de alto custo, não convergindo, portanto para um mínimo estável,



- alpha = 0.01 (Ideal): A convergência é rápida e estável.



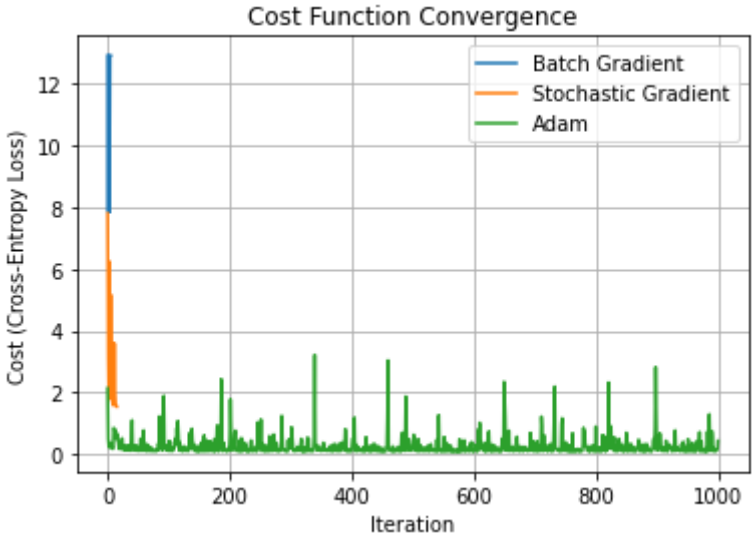
- alpha = 0.001 (Muito Baixa): A convergência é extremamente lenta. O modelo aprende, mas precisaria de muito mais interações para atingir um bom resultado.



IMPACTO DA REGULARIZAÇÃO

Sem a normalização dos dados mencionada anteriormente, o modelo não chega até uma convergência aceitável, pois as diferentes ordens de grandeza dos coeficientes dos dados geram perturbações na busca dos parâmetros da função sigmoidal.

O comportamento da função de custo dos diferentes métodos pode ser visto abaixo, demonstrando que o correto pré processamento dos dados de entrada é crucial para obter-se o resultado desejado em processos de aprendizado de máquina.



USO DE IA

O uso de modelos de Inteligência Artificial foi marginal neste trabalho, com a ferramenta auxiliando apenas a validar raciocínios e executar pequenas tarefas pré-determinadas, como por exemplo criar os comandos de plot de imagens.

Além disso, ela foi importante para consolidar alguns conceitos, como o da implementação do método de Stochastic Gradient Descent, onde no meu entendimento inicial, ela não deveria ser calculada com base em todo o conjunto de dados de modo cumulativo, mas sim em amostras parciais do conjunto.

Após ela me corrigir, fui em busca de fontes externas de implementações deste método para confirmar a assertividade da IA, e ao perceber que ela estava certa, mudei minha abordagem do método desta classe.

CONCLUSÃO

A análise prática dos três otimizadores revela um claro trade-off entre estabilidade, velocidade de convergência e complexidade de implementação.

- O **Batch Gradient Descent**, embora conceitualmente simples e estável, é amplamente impraticável para os datasets modernos.
- O **Stochastic Gradient Descent** é amplamente utilizado em diversas aplicações, oferecendo convergência rápida, mas exigindo um ajuste cuidadoso de hiperparâmetros.
- O **Adam** se estabelece como a opção mais robusta e eficiente na maioria dos cenários, combinando velocidade, estabilidade e menor sensibilidade a hiperparâmetros.