Universidade de São Paulo
Instituto de Matemática e Estatística
Aluna: Nara Avila Moraes
Nro. USP: 5716734

2o. semestre, 2025
Exercício 01 - 22/08

MAC5921/IME: Deep Learning.
Prof.: Nina S. T. Hirata

Implementação de modelos de regressão utilizando Scikit-Learn e PyTorch

(1.1) Regressão linear utilizando scikit-learn e pytorch.

(1.2) Regressão logística utilizando scikit-learn e pytorch

Relatório para o Ítem 1.1

Relatório Comparativo: Métodos de Regressão Linear

Neste estudo foram implementados e comparados quatro métodos de regressão linear aplicados ao mesmo conjunto de dados: (i) regressão linear analítica (equação normal), (ii) regressão linear com a biblioteca `scikit-learn`, (iii) regressão linear iterativa com Batch Gradient Descent (BGD) e (iv) regressão linear iterativa com Stochastic Gradient Descent (SGD).

Resultados de Ajuste

Os métodos analítico e via `scikit-learn` apresentaram resultados praticamente idênticos em termos de coeficientes (w_0, w_1) e métricas de qualidade, como R^2, MSE, SSE e SSR. Isso já era esperado, visto que ambos utilizam a solução fechada da equação normal.

Nos métodos iterativos, tanto o BGD quanto o SGD atingiram valores de R^2 semelhantes, confirmando a convergência para soluções próximas à analítica. Entretanto, a trajetória até a convergência apresentou características distintas: o BGD exibe uma curva de loss suave e monotônica, enquanto o SGD apresenta oscilações acentuadas devido à atualização baseada em amostras individuais, refletindo seu caráter estocástico.

Custo Computacional

O método analítico requer a inversão da matriz $X^T X$, o que pode ser custoso para grandes dimensões. No entanto, no dataset atual, o tempo de execução foi desprezível. O `scikit-learn`, que encapsula essa solução, teve comportamento semelhante. Já os métodos iterativos apresentaram maior tempo de execução, sendo o BGD mais lento que o SGD, embora este último exiba maior variabilidade no processo de convergência.

Table 1: Métricas de desempenho da Regressão Linear para diferentes métodos

Método	Par	R^2	MSE	SSE	SSR	SST
Scikit-learn	Shoe → Height	0.710977	60.299	15496.8	38121.2	53618.1
PyTorch	Shoe → Height	0.710977	60.299	15496.8	38121.2	53618.1
BGD	Shoe → Height	0.710937	60.308	15497.2	38120.8	53618.0
SGD	Shoe → Height	0.710933	60.309	15497.2	38120.8	53618.0
Scikit-learn	Sex → Shoe	0.466478	0.116	29.59	25.88	55.47
PyTorch	Sex → Shoe	0.466478	0.116	29.59	25.88	55.47
BGD	Sex → Shoe	0.466453	0.117	29.60	25.87	55.47
SGD	Sex → Shoe	0.466449	0.117	29.60	25.87	55.47

Particularidades

- **Analítico:** fornece a solução exata em uma única etapa, mas não escala bem para bases massivas.
- **Scikit-learn:** prático e otimizado, equivalente à solução analítica, com tratamento robusto de dados.
- **BGD:** garante convergência suave, porém demanda mais tempo de processamento em datasets maiores.
- **SGD:** converge rapidamente em termos de épocas, com custo computacional reduzido, mas apresenta flutuações que exigem maior cuidado na escolha da taxa de aprendizado.

Table 2: Custo computacional da Regressão Linear

Método	Iterações	Gradiente final	Tempo (s)
Scikit-learn	–	–	0.0009
PyTorch	–	–	0.0006
BGD	200000	0.00037	48.13
SGD	∼400	0.0223	0.6277

Conclusão

A análise evidenciou que, para problemas de pequena dimensão, a solução analítica e o uso de bibliotecas otimizadas como o `scikit-learn` são os métodos mais diretos e eficientes. Já em contextos de grande escala, os métodos iterativos tornam-se mais viáveis, sendo o SGD especialmente útil pela eficiência computacional, apesar da necessidade de ajustes finos em seus hiperparâmetros.

Relatório para o Ítem 1.2

Relatório Comparativo: Regressão Logística Binária

Objetivo

O objetivo deste experimento foi comparar quatro implementações distintas de regressão logística binária para o alvo `sex` (0 = Female, 1 = Male), utilizando como preditores as variáveis `height`, `weight`, `age` e `shoe_number`. As implementações foram:

1. **scikit–learn**, como baseline de referência;
2. **PyTorch**, modelo direto com otimização via `Adam`;
3. **Batch Gradient Descent (BGD)**, atualização por época com todo o conjunto;
4. **Stochastic Gradient Descent (SGD)**, atualização em mini-lotes.

Em todas as abordagens, as features foram padronizadas (média 0, desvio 1) para garantir estabilidade numérica.

Métricas principais (teste)

Table 3: Regressão Logística (alvo: *Sex*). Métricas no conjunto de teste.

Método	Acc	Prec	Rec	F1	ROC–AUC
Scikit–learn (Pipeline)	0.8615	0.9500	0.8444	0.8941	0.9322
PyTorch (BGD)	0.8615	0.9500	0.8444	0.8941	0.9311
PyTorch (SGD)	0.8615	0.9500	0.8444	0.8941	0.9290

Observações

1. **Padronização**: foi determinante para estabilizar o treinamento. Sem ela, observou-se saturação dos gradientes e convergência lenta/instável, sobretudo em BGD e SGD.
2. **Curvas de aprendizado**:
   - O BGD apresentou uma curva de perda suave, convergindo de forma estável;
   - O SGD mostrou comportamento serrilhado, com maior ruído. Esse ruído, no entanto, pode ser benéfico, ajudando o modelo a escapar de mínimos locais rasos.
3. **Baseline**: o scikit–learn ofereceu um baseline sólido, rápido e de fácil interpretação, servindo como referência para os demais métodos.
4. **Flexibilidade**: o PyTorch reproduziu os resultados do baseline, mas com maior flexibilidade para ajustes de arquitetura, regularização e integração futura em modelos neurais.

Table 4: Matriz de confusão no teste (métodos scikit, BGD e SGD).

	TN	FP	FN	TP
Todos (scikit, BGD, SGD)	18	2	7	38

Custo Computacional

- **scikit–learn** e **PyTorch** direto apresentaram tempos de execução reduzidos (milissegundos), ideais para prototipagem.
- **BGD** foi mais lento por época, pois cada atualização utiliza o dataset completo.
- **SGD** exigiu maior número de épocas para alcançar desempenho similar, mas cada época foi mais barata em custo computacional.

Conclusões

As quatro abordagens produziram resultados consistentes e coerentes. As principais conclusões são:

1. Para tarefas práticas de classificação binária simples, recomenda-se iniciar com **scikit–learn**, pela rapidez e confiabilidade.
2. Quando há necessidade de **controle detalhado** sobre o processo de treinamento ou integração com redes neurais, o uso de **PyTorch** é mais indicado.
3. O **BGD** é preferível quando se busca estabilidade e curvas de aprendizado mais suaves, útil para análise didática ou diagnóstica.
4. O **SGD** é recomendado em cenários de grandes volumes de dados, onde eficiência e regularização implícita via ruído são desejáveis.

Este experimento reforça um princípio fundamental do aprendizado de máquina: a escolha do método não deve ser apenas pela acurácia, mas também considerando **custo computacional, estabilidade do treinamento e contexto de aplicação**. Cada abordagem traz pontos fortes que podem ser explorados conforme o objetivo final do projeto.