Laboratório 4 - Machine Learning
Ensembles

Dimmy K. S. Magalhães[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brazil

dksmagalhaes@inf.ufpr.br

1. Introdução

O presente trabalho consiste nos resultados obtidos no quarto laboratório da disciplina de Aprendizagem de Máquina. Utilizando o conhecimento sobre Ensembles foram realizados experimentos com Bagging, Boosting e RSS utilizando a base de dígitos afim de responder os seguintes questionamentos:

1.1. Bagging

Avalie o desempenho dos seguintes algortimos de ensembles em termos de acurácia e custo computacional (tempo de CPU).

1. Bagging
2. Boosting (Adaboosting, Gradient Boosting)
3. RSS (Randon Forest, ExtraTrees)

Para o Bagging avalie o Perceptron e Árvore de decisão com classificadores de base e verifique qual é quantidade de classificadores necessária para maximizar a taxa de reconhecimento na base de teste. Qual foi a melhora com relação ao classificador de base.

1.2. Boosting

Os algoritmos baseados em Boosting também são sensíveis ao parâmetro que controla a contribuição de cada classificador na combinação final (learning rate). Compare as implementações de Adaboosting e GradientBoosting e reporte os resultados.

1.3. RSS

Finalmente compare os classificadores baseados em RSS, ou seja, Random Forest e ExtraTrees. Reporte os resultados e compare com as estratégias anteriores.

2. Metodologia

Para executar o experimento foram utilizadas as três técnicas de Ensembles apresentadas (Bagging, Boosting e RSS), todas elas implementadas em Python usando o framework scikit-learn.

Para cada experimento era selecionado um dos parâmetros de entrada (sejam número de classificadores ou learning rate) e o algoritmo era executado dez vezes sendo analisado o pior, melhor e caso médio, sendo este último coletado e apresentado como resultado do experimento. Para para execução do algoritmo um dos parâmetros é testado até que se tenha uma acurácia satisfatória, então este é fixado para que o próximo parâmetro

seja testado. Os resultados colhidos dizem respeito ao valor da acurácia do modelo e o tempo de execução.

Para todos os experimentos foi utilizado um nó da AI Dev Cloud da Intel, possuindo processadores escaláveis Intel Xeon, com 24 cores cada um, cada processador tendo acesso a 95GB de memória RAM. O experimento foi customizado para utilizar um processador para executar cada tarefa.

Em todos os experimentos foi utilizada a técnica de Cross Validation usando a ferramenta KFold com 10 partições e o tempo de execução dos algoritmos é medido em segundos.

3. Experimentos

3.1. Bagging

O experimento 1 diz respeito a execução de ensembles utilizando *bagging* e os classificadores: perceptron e árvores de decisão. Para o experimento 1 os seguintes parâmetros foram considerados:

1. Quantidade de classificadores
2. Classificadores: Perceptron e Árvores de decisão

Os resultados são apresentados na Tabela 1.

Tabela 1. Experimentos Bagging

Qt	Perceptron		Árvore de decisão	
	Acurácia	**Tempo**	**Acurácia**	**Tempo**
2	0.921427	8.3113	0.860229	33.6171
4	0.934608	10.7844	0.894980	59.8976
8	0.943304	17.2674	0.908365	115.2225
16	0.944310	29.6258	0.916397	228.7503
32	0.945657	53.8483	0.920694	447.1759
64	0.945759	103.4834	0.923814	890.7583
128	0.944992	203.9282	0.925622	1779.0250
256	0.945316	402.3259	0.926099	3539.5602

Observa-se em que em todas as instâncias de configuração o ensemble com Perceptron alcança uma acurácia maior em bem menos tempo que as árvores de decisão. A figura 1 demonstra a matriz de confusão encontrada para a melhor configuração de ensemble usando Perceptron.

A figura 2 demonstra a matriz de confusão para o ensemble com melhor acurácia usando árvores de decisão, entretanto com maior tempo de execução.

3.2. Boosting

O experimento 2 diz respeito a execução de ensembles utilizando *boosting* com *Ada Boosting* e *Gradient Boosting*. Para o experimento 2 os seguintes parâmetros foram considerados:

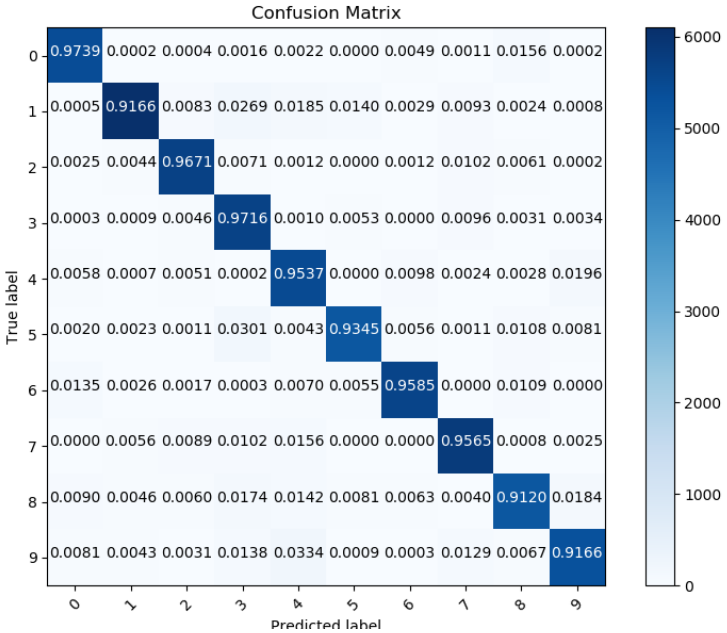

Figura 1. Matriz de confusão para ensemble usando perceptron

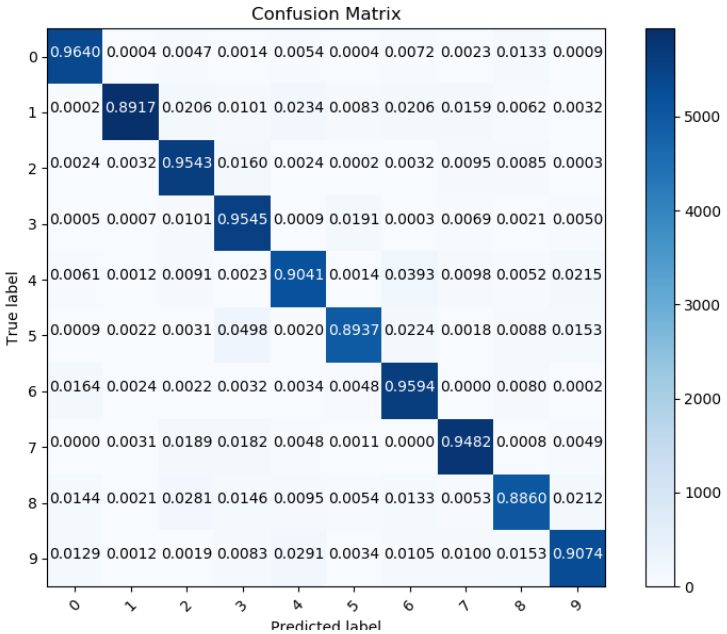

Figura 2. Matriz de confusão para ensemble usando árvores de decisão

1. Quantidade de classificadores
2. *Learning rate*
3. Algoritmos: *Ada Boosting* e *Gradient Boosting*

Os resultados são apresentados na Tabela 2.

A figura 3 demonstra a matriz de confusão encontrada para a melhor configuração de ensemble usando Ada Boosting.

A figura 4 demonstra a matriz de confusão para o ensemble com melhor acurácia

Tabela 2. Experimentos Boosting

Qt	LR	Ada Boosting		Gradient Boosting	
		Acurácia	**Tempo**	**Acurácia**	**Tempo**
2	0.1	0.270163	7.8431	0.845599	53.2366
	0.5	0.270163	7.0777	0.866982	52.8132
	0.8	0.270163	7.7353	0.866760	52.8172
	0.9	0.270163	7.1570	0.860553	53.1382
	1.0	0.270163	7.1267	0.858780	58.2768
4	0.1	0.413873	9.5547	0.870136	99.3792
	0.5	0.405330	9.3408	0.894656	104.9957
	0.8	0.483102	9.5625	0.878645	98.8725
	0.9	0.482778	9.3576	0.866419	98.9809
	1.0	0.482488	9.4687	0.876121	98.7688
8	0.1	0.655816	13.8309	0.884186	197.6871
	0.5	0.743717	13.7579	0.909951	193.9628
	0.8	0.750810	13.8263	0.891195	202.8346
	0.9	0.715189	13.9571	0.885346	198.9569
	1.0	0.714934	13.9995	0.895389	195.7491
16	0.1	0.697439	22.4106	0.908621	397.9614
	0.5	0.812553	22.5838	0.920046	382.5479
	0.8	0.807847	22.6039	0.903727	379.6287
	0.9	0.800890	22.6507	0.894827	378.8563
	1.0	0.768049	22.4812	0.895031	376.3171
32	0.1	0.745405	40.1500	0.930549	789.8409
	0.5	0.829775	39.9497	0.935170	759.8197
	0.8	0.850851	40.3005	0.921205	751.4667
	0.9	0.822921	39.8658	0.902551	778.7042
	1.0	0.773744	40.0896	0.900471	747.8097
64	0.1	0.798145	75.3709	0.941224	1544.2883
	0.5	0.859973	75.0561	0.941513	1491.2963
	0.8	0.852130	75.2171	0.929185	1170.9247
	0.9	0.803022	75.6023	0.914709	1267.2612
	1.0	0.785748	75.3919	0.914078	1490.3830
128	0.1	0.842837	145.5230	0.947447	3048.4440
	0.5	0.849026	144.7693	0.943099	1681.9315
	0.8	0.859172	145.5732	0.930822	1300.3054
	0.9	0.805903	145.5508	0.941232	1700.9851
	1.0	0.821897	145.9262	0.921232	1612.4532

LR: Learning Rate

usando Gradient Boosting

Analisando os algoritmos de boosting, o Gradient Boosting apresenta significativa vantagem, superando o Ada Boosting mesmo com parâmetros reduzidos, isto é, número de classificadores e taxa de aprendizagem.

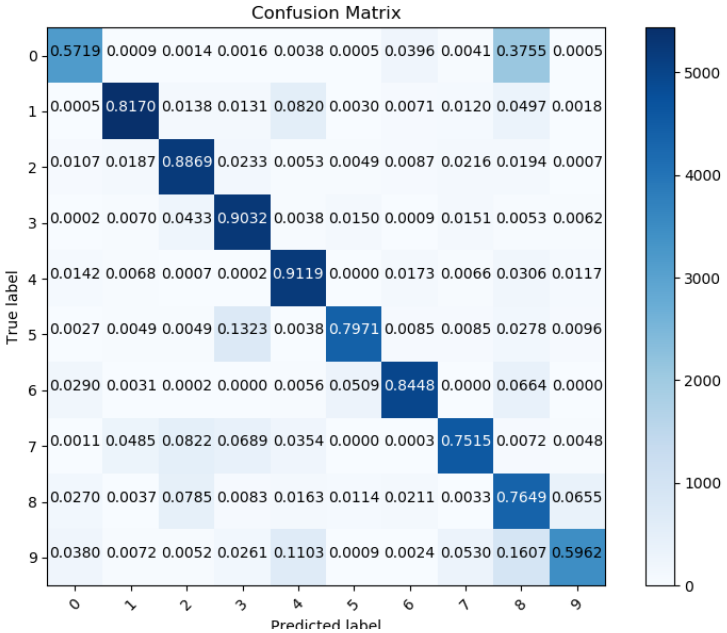

Figura 3. Matriz de confusão para ensemble usando Ada Boosting

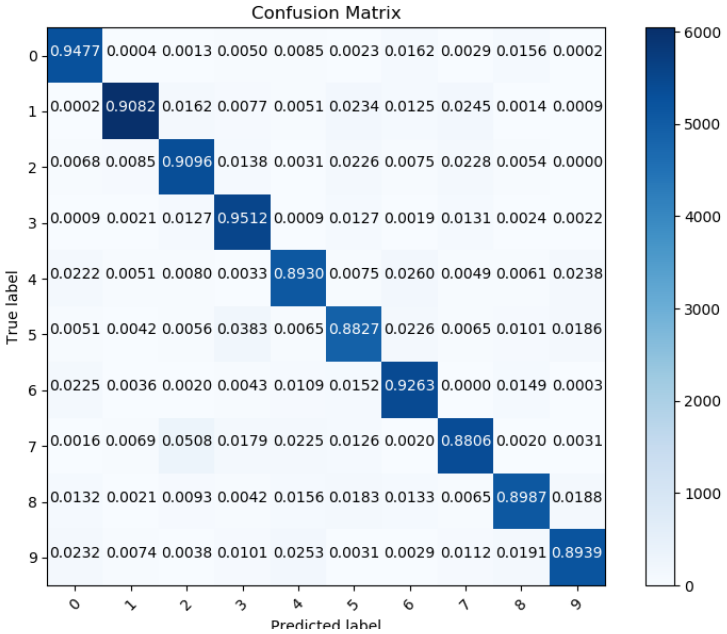

Figura 4. Matriz de confusão para ensemble usando Gradient Boosting

3.3. RSS

O experimento 3 diz respeito a execução de ensembles RSS e os classificadores: *Random Forest* e *Extra Trees*. Para o experimento 3 os seguintes parâmetros foram considerados:

1. Quantidade de classificadores
2. Algoritmos: *Random Forest* e *Extra Trees*

Os resultados são apresentados na Tabela 3.

Tabela 3. Experimentos RSS

Qt	Random Forest		Extra Trees	
	Acurácia	**Tempo**	**Acurácia**	**Tempo**
2	0.807506	8.6360	0.933533	25.8877
4	0.903182	11.4363	0.935017	25.7218
8	0.930584	17.7184	0.930652	24.9771
16	0.946015	30.9745	0.933670	24.9466
32	0.954183	56.5765	0.935869	25.3120
64	0.955939	108.2326	0.938223	25.0681
128	0.958088	211.0302	0.933107	25.2079
256	0.959230	415.5688	0.932442	24.9992

Observa-se em que mesmo atingindo uma acurácia maior (com 256 classificadores), o ensemble com Random Forest consome uma quantidade de tempo proporcional ao número de classificadores, nesse contexto o algoritmo Extra Trees atinge uma acurácia de 93% em todas as instâncias com tempo estatisticamente constante. A figura 5 demonstra a matriz de confusão encontrada para a melhor configuração de ensemble usando Random Forest.

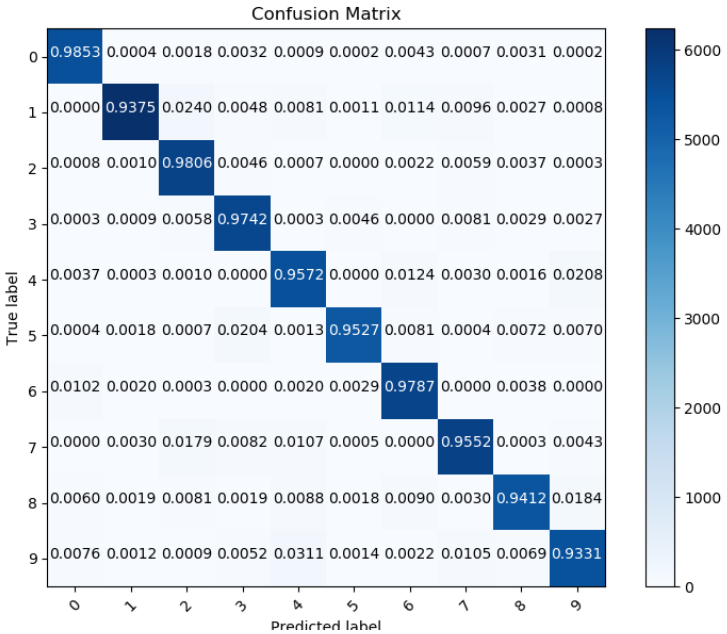

Figura 5. Matriz de confusão para ensemble usando Random Forest

A figura 6 demonstra a matriz de confusão para o ensemble com melhor acurácia usando Extra Trees.

4. Conclusões

Analisando todos os experimentos pode-se inferir que o algoritmo de Ensemble com Random Forest atinge as melhores acurárias, atingindo 95.92% contra 85.99% do Ada Boosting, 94.74% do Gradient Boosting e 94.57% do ensemble com perceptron. Entretanto

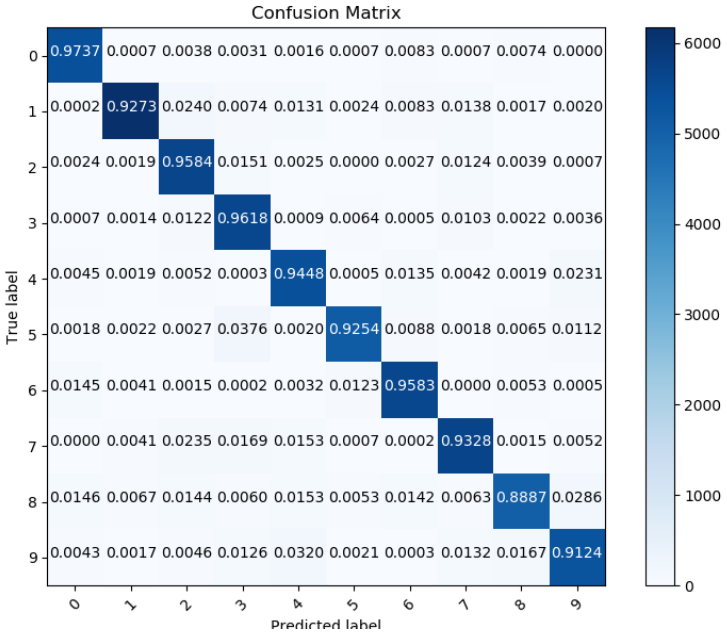

Figura 6. Matriz de confusão para ensemble usando Random Forest

o custo computacional avaliado foi de 415 segundos, muito além dos 103 segundos utilizados pelo algoritmo com perceptron. Ressalta-se que os algoritmos foram executados em nós distintos e dessa forma o tempo de execução de um não interferiu em nenhum momento na execução do outro. Assim conclui-se que se a condição de tempo execução for restritiva então ao algoritmo com Perceptron deve ser selecionado para problema, caso contrário o algoritmo com Random Forest deve ser selecionado.