USP — UNIVERSIDADE DE SÃO PAULO

Aprendizagem automática de redes soma-produto

Relatório Parcial de Projeto de Iniciação Científica
PIBIC Projeto 800585/2016-0

Bolsista: Renato Lui Geh
Orientador: Prof. Dr. Denis Deratani Mauá

São Paulo

2018

1 Introdução

Aprendizado de máquina é uma área da Inteligência Artificial cujo objetivo é estatisticamente ajustar os parâmetros de um certo modelo matemático dado um conjunto de dados a fim de que seja possível fazer previsões acuradas do que se está modelando. Para isso, faz-se uso de modelos estatísticos e métodos computacionais. Modelos probabilísticos baseados em grafos (PGM, do inglês *Probabilistic Graphical Model*), são uma classe destes modelos estatísticos em que se usa grafos para representá-los graficamente.

Modelos probabilísticos baseados em grafos representam uma distribuição de probabilidade de forma compacta. Estes modelos representados por grafos facilitam tanto a compreensão humana ao estudá-los, quanto possibilitam que vários problemas já existentes em Teoria dos Grafos sejam utilizados como solução para problemas em PGMs. Extrair conhecimento de PGMs é análogo a extrair a probabilidade de um certo evento ocorrer dado que eventos distintos tenham ocorrido. Tal extração de conhecimento é chamada de inferência. Fazer inferência exata em PGMs clássicas, ou seja, acharmos a probabilidade exata de um certo evento, é intratável. Uma solução para este problema é utilizarmos métodos para inferência aproximada nestes modelos. No entanto, tais algoritmos aproximados são muitas vezes difíceis de analisar. Além disso, como os algoritmos de aprendizado do modelo utilizam inferência como subrotina, por consequência o aprendizado torna-se aproximado.

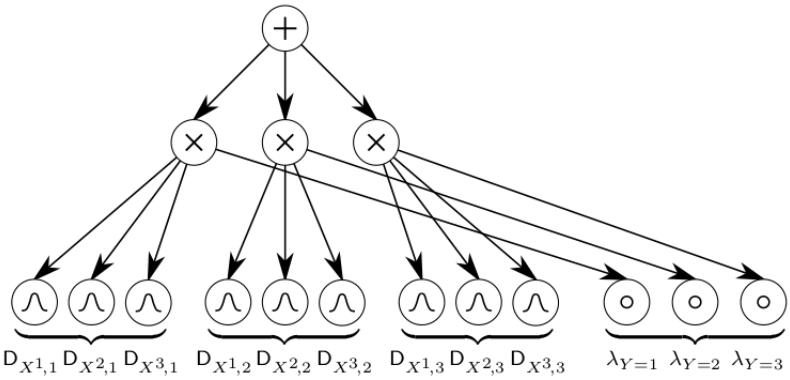

Figura 1

Redes soma-produto (SPN, de *Sum-Product Network*) são PGMs que representam uma distribuição de probabilidade tratável. Proposto em 2011, SPNs computam inferência exata em tempo linear ao número de arestas de seu grafo se sua estrutura obedecer a certas propriedades [14]. A Figura 1 mostra um exemplo de rede soma-produto representando o modelo Naïve Bayes com três atributos. SPNs apresentam uma série de características interessantes, como sua arquitetura profunda que permite representar funções de forma mais eficiente quanto mais profundo seu grafo [4]. Outras interessantes propriedades teóricas incluem uma generalização de SPNs para qualquer semianel em que o produto tenha escopo disjunto [8]. Com relação a aplicações, SPNs tiveram resultados impressionantes em diversas áreas, como enovelamento de proteínas [7], modelagem de sinais [12], classificação e reconstrução de imagens [10, 14, 5], reconhecimento de atividade [1] e linguagem natural [2].

Neste projeto, pretende-se desenvolver uma biblioteca livre e gratuita para inferência e algoritmos de aprendizado estado-da-arte em SPNs, analisando-se experimentalmente tais algoritmos em tarefas reais, como compleição e classificação de imagens.

2 Atividades desenvolvidas

O objetivo do projeto é o desenvolvimento de uma biblioteca livre e gratuita para inferência e aprendizado de SPNs. O nome dado a biblioteca foi GoSPN e encontra-se disponível em `https://github.com/RenatoGeh/gospn`. A linguagem de programação escolhida foi Go. Essa escolha deu-se por causa de sua sintaxe simples, tempo de compilação e execução rápidos, e suporte nativo para programação concorrente e paralela. Foram implementadas diversas funcionalidades à biblioteca, que serão enumeradas a seguir e em seguida exploradas em mais detalhes.

1. Inferência por marginais e MAP;
2. Derivadas em função da rede e pesos;
3. Aprendizado generativo por descida de gradiente;
4. Aprendizado estrutural de Gens-Domingos;
5. Geração de estrutura densa (Poon-Domingos);
6. Suporte para conjuntos de dados ARFF para variáveis discretas;
7. Testes de completude e decomponibilidade.

Os itens 4 e 6 foram implementados antes do início da bolsa.

Redes soma-produto são DAGs cujos nós podem ser nós somas, em que o valor da soma é a média ponderada de seus filhos; nós produtos, cujos valores são o produtório de seus filhos; ou folhas, que representam distribuições de probabilidade univariadas. O valor de uma SPN é o valor de sua raíz. O escopo de um nó da SPN é a união entre o escopo de seus filhos. O escopo de uma folha é o escopo da distribuição de probabilidade.

Computar inferência por marginais equivale a achar a probabilidade de um evento dada uma evidência. Em redes soma-produto, computar tal probabilidade consiste em achar o valor da raíz da SPN dados valores pré-determinados de certas variáveis. Foram implementadas duas funções na biblioteca para computar os marginais. A primeira é um método de classe que utiliza recorrência para achar o valor dos subsequentes filhos de forma *top-down*. A segunda é uma função estática que usa programação dinâmica para evitar recomputar valores. Adicionalmente, esta função usa uma estrutura de dados para armazenar os nós visitados, evitando usar a pilha de chamada.

A probabilidade maximum-a-posteriori (MAP) é uma estimativa do evento mais provável de se ocorrer dada uma evidência. De forma análoga a computar os marginais, GoSPN permite usar duas funções para computar o MAP. Computar o MAP em SPNs é NP-difícil [11, 3]. Em GoSPN, implementamos um algoritmo aproximado apresentado por [14].

Aprendizado generativo por descida de gradiente em redes soma-produto é realizado computando as derivadas da rede em função dos seus parâmetros. GoSPN possue várias funções para achar as derivadas dadas instâncias. Também foram implementadas versões para computar as derivadas em lote. Após achadas as derivadas, é possível atualizar os parâmetros da rede aplicando o gradiente. Em GoSPN, foi implementado apenas o aprendizado de parâmetros por descida de gradiente de forma generativa.

Diz-se aprendizado de parâmetros quando atualizam-se apenas os pesos da SPN. O aprendizado estrutural de SPNs ajusta tanto os pesos quanto o grafo da rede. Foi criada uma implementação do esquema de algoritmo de aprendizado estrutural de Gens e Domingos. Em [10], é dada certa liberdade quanto a certos pontos do algoritmo. O esquema é composto por três casos executados recursivamente:

`LearnSPN` [10]

Input Dataset $\mathbf{D} = (I, X)$, onde I é o conjunto de instâncias e X o conjunto de variáveis
Output SPN representando a distribuição de probabilidade de I sobre as variáveis X

if $|X| = 1$ **then**
    retorna distribuição de probabilidade univariada de I
else
    Particione X em $P_1, P_2, \ldots, P_m$ tal que todo P_i é independente de P_j, $i \neq j$
    if $m > 1$ **then**
        retorna nó produto cujos filhos são as recorrências `LearnSPN`(I, P_i), para todo $1 \leq i \leq m$
    else
        Rode *clustering* em I tal que $Q_1, Q_2, \ldots, Q_n$ são os clusters de I
        retorna nó soma cujos filhos são as recorrências `LearnSPN`(Q_i, X) com peso $|Q_i|/|I|$
    end if
end if

Em GoSPN, foram implementados os algoritmos de *clustering* DBSCAN e *k-means*. Para os testes de independência foram implementados o teste de χ^2 e *G-test*.

Para replicação dos resultados obtidos em [14], assim como validação dos algoritmos de aprendizado de parâmetros, fez-se necessária a implementação da estrutura densa descrita em [14]. O algoritmo de geração de tal estrutura cria uma rede adequada para conjuntos de dados que apresentam dependência local entre variáveis. Este tipo de estrutura é aplicável a compleição e classificação de imagens, já que *pixels* vizinhos apresentam coloração semelhante. A geração desta estrutura depende principalmente de três parâmetros, o número de

nós somas por cada região retangular da imagem, denotada por m; o número de distribuições gaussianas por pixel, denotada por g; e o menor nível de resolução da imagem em pixels, r.

Para validação dos algoritmos, é interessante utilizar conjuntos de dados de diferentes aplicações do mundo real. Para tanto, foi necessário adotar um formato padronizado. Foi implementado suporte para o formato ARFF, que é extensamente usado pela comunidade.

Além da validação empírica, também é importante verificar propriedades da rede. Uma SPN é completa quando todo nó soma tem mesmo escopo que seus filhos, e é consistente quando nenhuma variável tem valorações distintas nos filhos de um mesmo nó produto. Quando uma SPN é completa e consistente diz-se que ela é válida. Uma SPN válida computa a probabilidade exata de uma evidência. A SPN S é decomponível se, em um nó produto de S, os escopos dos filhos são disjuntos. Decomponibilidade implica em consistência, além do que SPNs consistentes não são mais compactas que SPNs decomponíveis [13]. Foram implementados funções que verificam se uma SPN é completa ou decomponível.

Além de funções que tem relação direta com redes soma-produto, também foi necessário implementar algoritmos clássicos de teoria dos grafos. Foram criadas funções para busca-em-profundidade, busca-em-largura, achar uma *spanning tree* de um grafo, e encontrar a ordem topológica de um DAG.

3 Problemas encontrados

Não foram encontradas implementações do algoritmo de aprendizado generativo por descida de gradiente. O artigo de Poon e Domingos comenta que aprendizado por descida de gradiente se dá naturalmente por *backpropagation*[14], no entanto o código anexado não apresenta nenhuma implementação do algoritmo. Adicionalmente, uma busca por implementações disponíveis indica a falta de uma implementação generativa por descida de gradiente. O algoritmo descrito no Item 3 foi implementado por meio das descrições contidas nos artigos [14, 9].

O tamanho da rede gerada pelo algoritmo de estrutura densa de Poon e Domingos também mostrou-se problemática. O número de nós no grafo gerado com uma imagem 46×56 e com parâmetros $m = 4$, $g = 4$ e $r = 4$ foi 1.701.333. O tempo total gasto no aprendizado de tal SPN foi de aproximadamente 6 horas e ocupou 1GB de memória. No entanto, tais parâmetros usados são considerados pequenos, já que os parâmetros usados no artigo original foram $m = 20$, $g = 4$ e $r = 4$ em uma imagem de tamanho 64×64. Ao gerarmos o grafo com tais parâmetros, excedeu-se a memória de 16GB. No artigo original, foi usado um *cluster* de 48 máquinas com 8 processadores e 16GB de memória cada, o que explica a discrepância.

4 Resultados

Até o momento foi apenas possível extrair resultados do algoritmo de aprendizado estrutural de Gens e Domingos. Tanto a implementação como a extração dos resultados foram realizadas antes do início da bolsa. Foram utilizados três conjuntos de dados: Caltech-101 [6], Olivetti Faces e Digits.

O primeiro contém imagens de diferentes dimensões e resoluções de 101 categorias de objetos. Foram escolhidas as categorias carro, motos e faces. As imagens foram padronizadas e redimensionadas em imagens em escala de cinza com 150 de largura e 65 de altura em resolução de 8-bits. O *dataset* está disponível em `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`.

Figura 2: Uma amostra do conjunto de dados Caltech-101.

O segundo é um conjunto de imagens de faces, com diferentes expressões e ângulos. As imagens têm dimensões 46×56 com resolução de 8-bits em escala de cinza. Como a distribuição de cores é irregular, com nenhum pixel tendo valores extremos como a cor branca, o algoritmo de independência entre variáveis apresentou resultados inconsistentes. Tal problema foi resolvido diminuindo a resolução das imagens para 3-bits. A qualidade das imagens não sofreu grande alteração com a mudança. Pode-se encontrar o *dataset* em `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`.

Figura 3: Uma amostra do conjunto de dados Olivetti. A imagem do lado esquerdo é uma amostra do *dataset* antes da diminuição de resolução. A imagem do lado direito mostra as imagens com resolução de 3-bits.

O último conjunto de dados é um *dataset* próprio. Foram criadas 700 imagens de dígitos de 0 a 9, com cada dígito tendo 70 amostras. A variância das imagens em cada categoria é baixa pois o *dataset* tem como amostra apenas a caligrafia de uma pessoa.

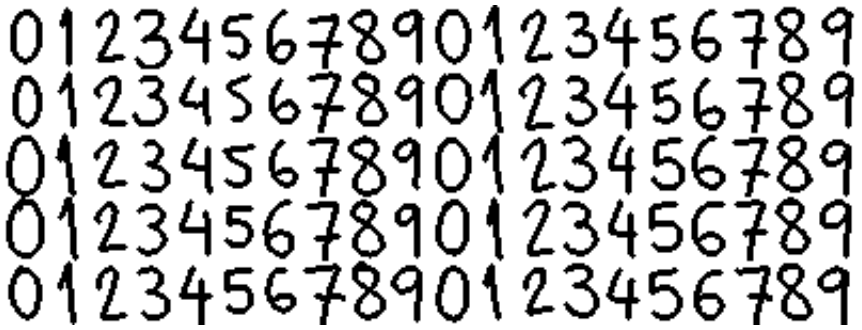

Figura 4: Uma amostra do conjunto de dados Digits.

Foram utilizados Caltech-101 e Digits para testes de classificação, e Olivetti para compleição de imagens. Para cada iteração, foi usado uma porcentagem p para validação cruzada, onde p é a porcentagem do *dataset* usado para treino, e $1 - p$ para teste.

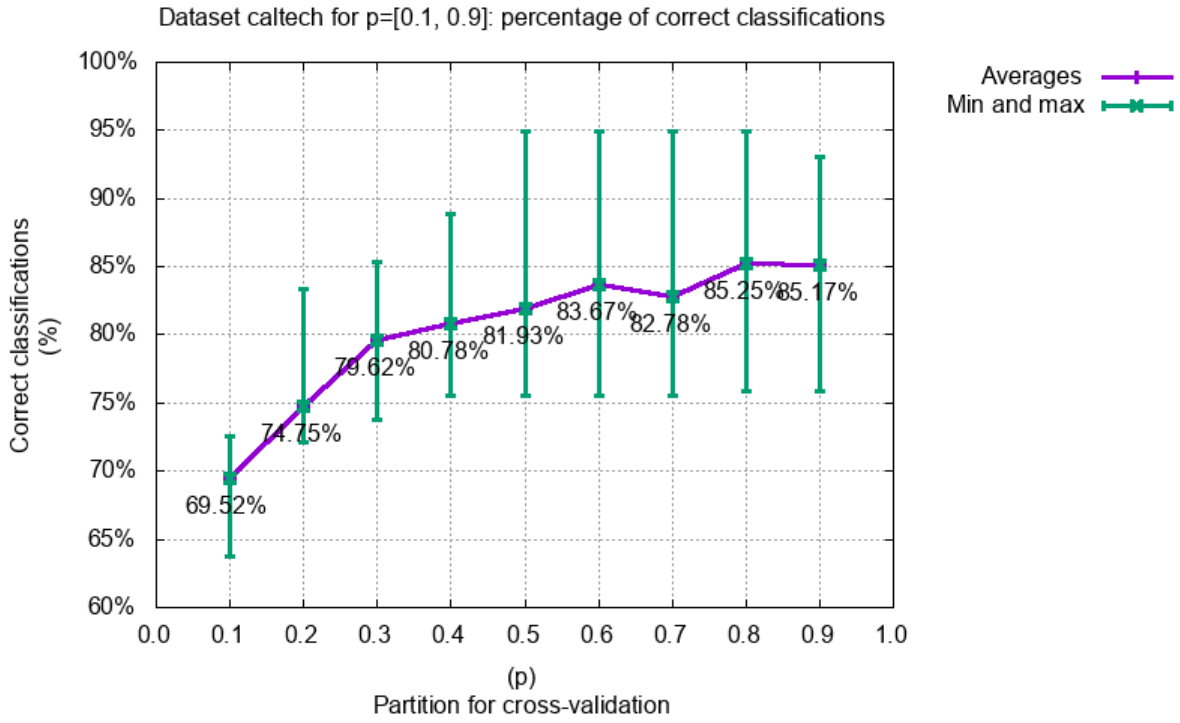

Figura 5: O gráfico mostra a porcentagem de acerto no conjunto de dados Caltech-101 normalizado para p variáveis.

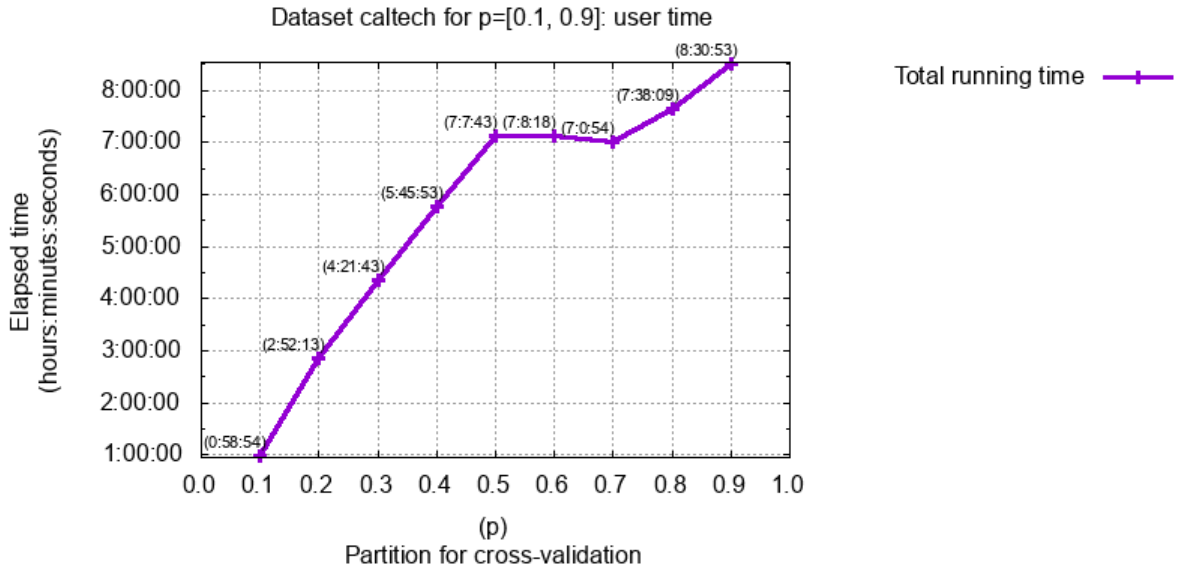

Figura 6: O gráfico acima mostra o tempo de aprendizado para cada p no *dataset* Caltech-101. O tempo de inferência foi muito menor, na escala de segundos.

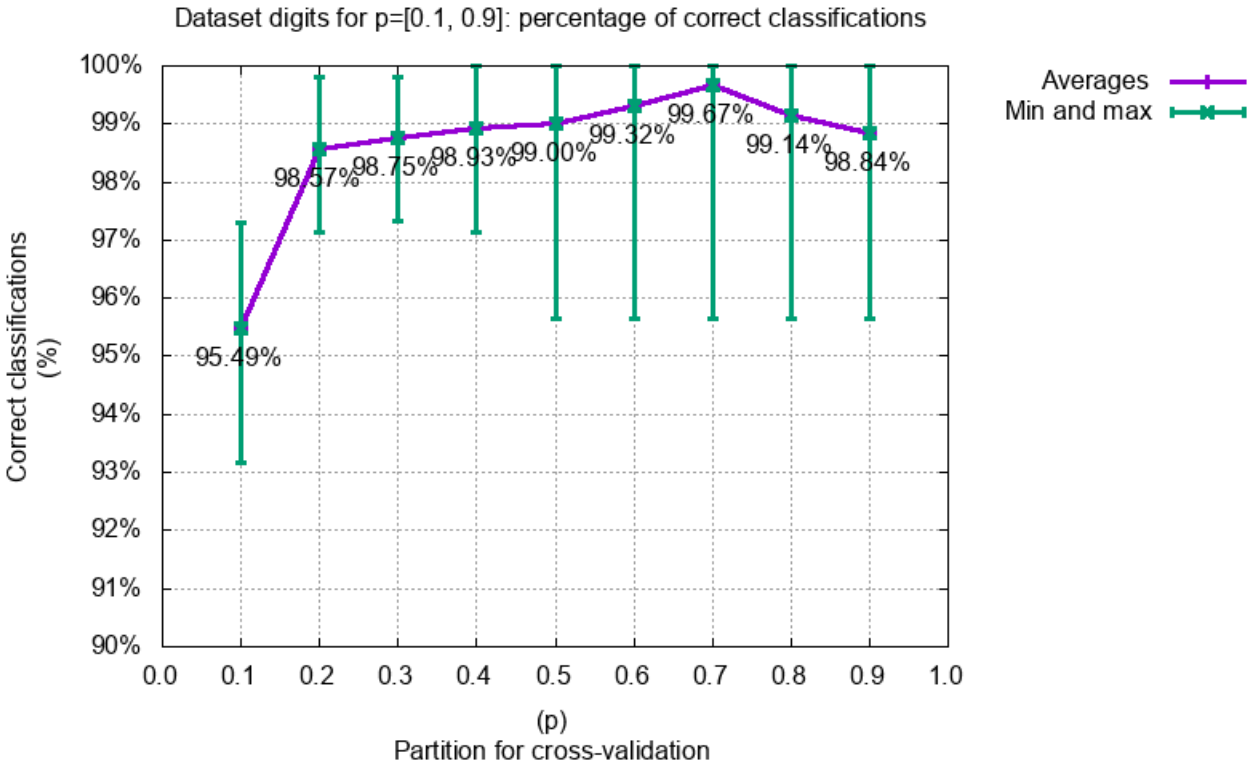

Figura 7: O gráfico mostra a porcentagem de acerto no conjunto de dados Digits.

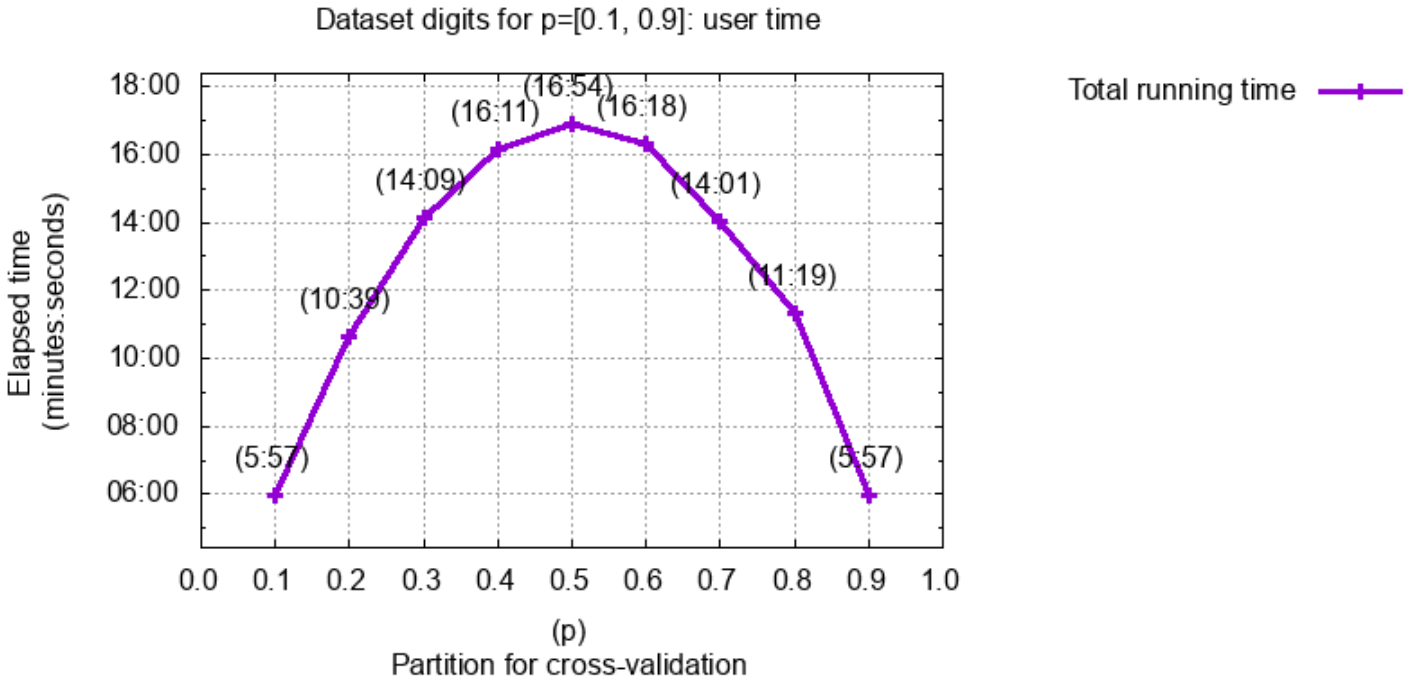

Figura 8: O gráfico mostra o tempo de aprendizado para cada p em Digits.

Para compleição de imagens no conjunto de dados Olivetti, foram aplicados dois testes. O conjunto Olivetti possue 10 imagens de cada face de uma pessoa. No primeiro teste, o modelo utilizou tanto as imagens de outras pessoas como 9 imagens da pessoa a ser completada. Chamou-se este método como teste com conhecimento prévio. Chama-se de teste sem conhecimento prévio quando apenas utilizamos as faces de outras pessoas somente.

Figura 9: O resultado da compleição da parte esquerda da face dada a parte direita do rosto. Neste teste houve conhecimento prévio. O modelo percebe a presença de óculos e pêlo facial.

Figura 10: A mesma face, porém sem conhecimento prévio. O modelo já não percebe a presença de óculos com a mesma taxa de acerto.

Referências

[1] Mohamed R. Amer e Sinisa Todorovic. "Sum-Product Networks for Activity Recognition". Em: *IEEE Transactions on Pattern Recognition and Machine Intelligence (TPAMI 2015)* (2015).

[2] Wei-Chen Cheng et al. "Language Modelling with Sum-Product Networks". Em: *Annual Conference of the International Speech Communication Association 15 (INTERSPEECH 2014)* (2014).

[3] Diarmaid Conaty, Denis D. Mauá e Cassio P. de Campos. "Approximation Complexity of Maximum A Posteriori Inference in Sum-Product Networks". Em: *Uncertainty in Artificial Intelligence 2017 (UAI 2017)* (2017).

[4] Olivier Delalleau e Yoshua Bengio. "Shallow vs. Deep Sum-Product Networks". Em: *Advances in Neural Information Processing Systems 24 (NIPS 1011)* (2011).

[5] Aaron Dennis e Dan Ventura. "Learning the Architecture of Sum-Product Networks Using Clustering on Variables". Em: *Advances in Neural Information Processing Systems* 25 (2012).

[6] L. Fei-Fei, R. Fergus e P. Perona. "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories". Em: *IEEE CVPR-2014 Workshop on Generative-Model Based Vision* (2014).

[7] Abram L. Friesen e Pedro Domingos. "Recursive Decomposition for Non-convex Optimization". Em: *International Joint Conference on Artificial Intelligence 24 (IJCAI 2015)* (2015).

[8] Abram L. Friesen e Pedro Domingos. "The Sum-Product Theorem: A Foundation for Learning Tractable Models". Em: *International Conference on Machine Learning 33 (ICML 2016)* (2016).

[9] Robert Gens e Pedro Domingos. "Discriminative Learning of Sum-Product Networks". Em: *Advances in Neural Information Processing Systems 25 (NIPS 2012)* (2012).

[10] Robert Gens e Pedro Domingos. "Learning the Structure of Sum-Product Networks". Em: *International Conference on Machine Learning* 30 (2013).

[11] Robert Peharz. "Foundations of Sum-Product Networks for Probabilistic Modeling". Tese de dout. Graz University of Technology, 2015.

[12] Robert Peharz et al. "Modeling Speech with Sum-Product Networks: Application to Bandwidth Extension". Em: *IEEE International Conference on Acoustics, Speech, and Signal Processing 39 (ICASSP 2014)* (2014).

[13] Robert Peharz et al. "On Theoretical Properties of Sum-Product Networks". Em: *International Conference on Artificial Intelligence and Statistics 18 (AISTATS 2015)* (2015).

[14] Hoifung Poon e Pedro Domingos. "Sum-Product Networks: A New Deep Architecture". Em: *Uncertainty in Artificial Intelligence* 27 (2011).