VI CONGRESSO NACIONAL DE ENGENHARIA MECÂNICA
VI NATIONAL CONGRESS OF MECHANICAL ENGINEERING
18 a 21 de agosto de 2010 – Campina Grande – Paraíba - Brasil
August 18 – 21, 2010 – Campina Grande – Paraíba – Brazil

PROJETO DE PONTES USANDO ALGORITMOS DE APRENDIZADO DE MÁQUINAS

Aloísio Carlos de Pina, aloisiopina@bol.com.br [1]
Vitor de Souza Colimodio, colimodio@poli.ufrj.br [2]
Adriano Armani da Silva, adrianoarmani@poli.ufrj.br [2]
Armando Carlos de Pina Filho, pina-filho@deg.ee.ufrj.br [3]

[1]Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Civil, CEP 21945-970, Rio de Janeiro, RJ, Brasil
[2]Universidade Federal do Rio de Janeiro, Escola Politécnica, Departamento de Engenharia Civil, CEP 21945-970, Rio de Janeiro, RJ, Brasil
[3]Universidade Federal do Rio de Janeiro, Escola Politécnica, Programa de Engenharia Urbana, CEP 21949-909, Rio de Janeiro, RJ, Brasil

Resumo: *O projeto de uma ponte é um processo muito complexo. Ele começa com a busca por localizações em potencial para uma ponte, junto com alguns esboços que necessitam passar por aprovação pública. Os projetistas necessitam exercitar seu julgamento com respeito a muitos aspectos, tais como estética, custo, determinação da carga, modos de falha, construtibilidade e manutenibilidade. O objetivo deste trabalho é mostrar a aplicação das técnicas mais modernas de Aprendizado de Máquinas na tomada de decisões no projeto de pontes, dadas as suas especificações. Para isso, foi usado um conjunto de dados fornecido pelo Professor Yoram Reich, do Departamento de Engenharia Civil e Centro de Pesquisa de Projetos de Engenharia, Universidade de Carnegie Mellon, Estados Unidos. O conjunto de dados foi pré-processado para identificar e eliminar as variáveis irrelevantes para o processo de aprendizado. Foram selecionados os algoritmos de aprendizado a serem aplicados e então foi realizada uma extensa avaliação experimental. Os resultados de todos os algoritmos utilizados foram comparados, realizando testes estatísticos para avaliar sua precisão e significância, a fim de determinar o modelo mais adequado ao problema.*

Palavras-chave: *pontes, decisões de projeto, Aprendizado de Máquinas*

1. INTRODUÇÃO

Denomina-se ponte, de modo geral, uma obra pela qual uma via de comunicação ou de canalização passa sobre um vale ou depressão do terreno. No entanto, atualmente, a denominação de ponte aplica-se especialmente a obras que atravessam correntes de água. Se essas obras atravessam vales secos, ou de forma que com as suas obras de apoio não reduzem, em caso algum, a seção de vazão da corrente atravessada, designam-se então por viadutos (EMPE, 2007).

Na realização de um projeto de uma ponte, são consideradas diversas variáveis para se chegar ao projeto final. Dentre elas pode-se citar: o comprimento, a largura, as matérias-primas, como serão os vãos, a finalidade da ponte, etc. Há vários tipos de pontes, dentre os quais se destacam as que serão apresentadas a seguir.

Ponte cantilever: é uma estrutura que, em geral, possui duas consolas, com um feixe de suspensão entre suas extremidades livres, sendo o cantilever o grande suporte (Fig. (1)). Ao longo da estrutura, as tensões de flexão e cisalhamento variam.

Ponte em arco: é uma estrutura semicircular havendo, em cada uma das extremidades, suportes de sustentação (Fig. (2)). O peso da ponte é naturalmente desviado para os suportes pelo semicírculo, o design do arco. Uma ponte desse tipo está sempre sujeita à força de compressão, que é empurrada para fora, em direção às pilastras, pela curva do arco. Entretanto, não é importante, em um arco, a tração, que pode ser desprezada, já que seus efeitos são muito reduzidos pela curva natural do arco e sua capacidade de dissipar a força para fora.

Ponte pênsil: é uma estrutura cujos cabos (cordas ou correntes) são pendurados sobre o obstáculo a ser vencido e a plataforma fica suspensa nesses cabos, que são pendurados em duas altas torres, que sustentam a maior parte do peso da plataforma (Fig. (3)). Sendo grandes os vãos, a sobrecarga é quase desprezível em relação ao peso próprio e a viga de rigidez pode então ser esbelta, já que precisa resistir somente a flexões devido a possíveis desigualdades de sobrecarga ao longo do vão, e deve poder flexionar com o alargamento dos cabos, não precisando ser tão rígida. As torres devem ser rígidas a fim de resistir a efeitos dinâmicos do vento.

Figura 1. Howrah Bridge, em Calcutá, na Índia: exemplo de ponte cantilever (ISI, 2010).

Figura 2. Aqueduto Segovia, na Espanha: exemplo de ponte em arco (Panoramio, 2010).

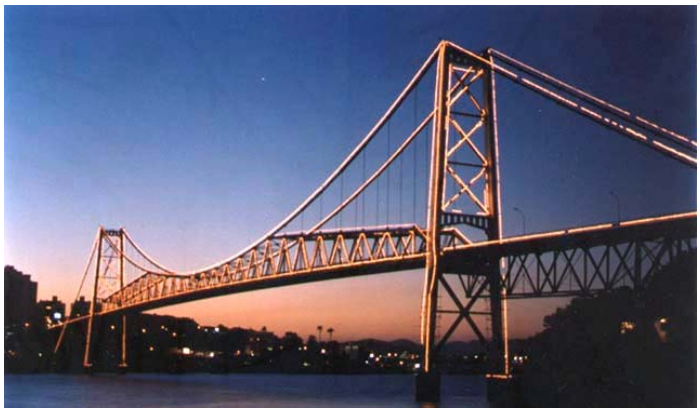

Figura 3. Ponte Hercílio Luz, em Florianópolis, Brasil: exemplo de ponte pênsil (ImageShack, 2010).

Ponte de treliças: é uma estrutura composta de muitas pequenas vigas que, juntas, são capazes de suportar grandes pesos e vencer grandes distâncias – treliças. Seus membros individuais são submetidos apenas a forças de tração e compressão, mas não a forças de flexão. As treliças podem ser simples (Fig. (4)) ou contínuas (Fig. (5)) e podem estar acima (Fig. (6)) ou abaixo (Fig. (4) e Fig. (5)) do leito de passagem da ponte, sendo mais uma variável a ser considerada no projeto de uma ponte.

O problema proposto é o seguinte: qual o melhor tipo de ponte a ser construída, sendo conhecidas as outras variáveis? Yoram Reich desenvolveu um sistema, chamado BRIDGER (Reich, 1990), a fim de auxiliar designers e arquitetos, em especial no que diz respeito a estética, no projeto de uma ponte. Esse sistema foi desenvolvido com técnicas de aprendizado de máquinas.

O objetivo desta pesquisa é buscar uma solução para o problema, utilizando também sistemas de aprendizado de máquinas, testando vários algoritmos diferentes, e depois fazer uma comparação dos resultados fornecidos por cada algoritmo, concluindo qual deles melhor desempenhou sua função. Na Seção 2, será apresentada a teoria básica sobre aprendizado de máquinas. Na Seção 3, o processo experimental será detalhado. E por fim, na Seção 4, serão discutidas as conclusões deste trabalho.

Figura 4. The First Bridge, em Wuhan, na China: exemplo de ponte de treliças simples, com as treliças abaixo do leito de passagem da ponte (Tumblr, 2010).

Figura 5. Rhinecliff Bridge, em Kingston, nos Estados Unidos: exemplo de ponte de treliças contínuas, com as treliças abaixo do leito de passagem da ponte (Flickr, 2009).

Figura 6. The General Hertzog Bridge, sobre o Rio Orange, na África do Sul: exemplo de ponte cujas treliças estão acima de seu leito de passagem (Harington, 2006).

2. APRENDIZADO DE MÁQUINAS

Um programa de Aprendizado de Máquinas (Mitchell, 1997) é um programa capaz de aprender com a experiência. Até hoje não se sabe uma forma de fazer um computador aprender tão bem quanto uma pessoa. No entanto foram desenvolvidos algoritmos que são eficientes em certas tarefas de aprendizado.

Um algoritmo de aprendizado aprende através de exemplos. Um exemplo é uma combinação de valores das variáveis (ou atributos) do problema. Portanto, para construir um conjunto de dados deve-se:

- Definir as variáveis do problema;
- Coletar experimentalmente seus valores;
- Construir exemplos com os valores coletados.

Deve-se definir as seguintes variáveis:

- Uma variável objetivo: variável que representa a informação que se deseja aprender.
- Variáveis auxiliares: variáveis cujos valores podem ser determinados e podem ajudar a prever o valor da variável objetivo.

As variáveis podem ser contínuas ou discretas. Se uma variável objetivo é contínua, ela é chamada de valor objetivo e o aprendizado é chamado de regressão; se ela é discreta, é chamada de classe e o aprendizado é chamado de classificação. Portanto, cada exemplo é uma combinação de valores das variáveis auxiliares para os quais se sabe qual é a classe ou valor objetivo associado.

Valores contínuos podem ser usados como se fossem discretos, usando-se um pré-processamento chamado discretização. A discretização permite:

- Aplicar algoritmos que só aceitam atributos discretos em problemas onde haja atributos contínuos.
- Usar um classificador para resolver um problema de regressão, embora os resultados possam não ser tão bons.

Atributos Irrelevantes podem significar desperdício computacional ou pior, podem dificultar o aprendizado. Deve-se, então, determiná-los e retirá-los dos dados antes do aprendizado. Alguns atributos são claramente irrelevantes e podem ser detectados e retirados manualmente, como um atributo que tem um valor para cada exemplo. Porém nem sempre é fácil saber quais atributos são irrelevantes. Diz-se que há ruído nos dados quando há neles alguma incoerência. Um ruído pode ser causado por medições erradas, fenômenos atípicos ou erros de digitação.

Um algoritmo de aprendizado recebe como entrada um conjunto de exemplos que é usado no aprendizado e então se torna capaz de responder o valor objetivo para qualquer combinação de valores de atributos. Uma combinação de atributos para a qual não se sabe o valor objetivo é chamada caso-teste e o conjunto de dados usado no aprendizado é chamado de Conjunto de Treinamento.

A resposta dada pelo programa treinado nem sempre é a correta, dependendo muito dos dados e do algoritmo que está sendo usado. Deve-se, então, separar parte dos exemplos para avaliar o desempenho do programa. Esses exemplos constituem o Conjunto de Teste. O processo se dá como segue:

- Dado um conjunto de dados, separa-se em treinamento e teste.
- O conjunto de treinamento é usado para treinar o algoritmo.
- Cada exemplo do conjunto de teste é transformado num caso-teste que é fornecido ao algoritmo.
- Comparam-se as respostas do programa com os valores objetivos reais e avalia-se o desempenho.

3. AVALIAÇÃO EXPERIMENTAL

Aqui serão apresentadas detalhadamente as ferramentas que foram utilizadas neste processo de aprendizado de máquinas. Na Subseção 3.1, será descrito o conjunto de dados utilizado. Na Subseção 3.2, será explicada a metodologia experimental utilizada. Na Subseção 3.3, serão apresentados os tipos de algoritmos utilizados. Finalmente, na Subseção 3.4, serão mostrados os resultados obtidos.

3.1. Conjunto de Dados

Para esta pesquisa foi utilizado um conjunto de dados disponibilizado no repositório virtual de Aprendizado de Máquinas da Universidade da Califórnia, em Irvine (Asuncion e Newman, 2007). O conjunto de dados em questão foi criado por Yoram Reich e Steven J. Fenves (Reich e Fenves, 1992) e contém informações de projetos de pontes de Pittsburgh, nos Estados Unidos.

Existem duas versões do conjunto de dados: a primeira sendo a versão original e a segunda contendo descrições após discretização de propriedades numéricas. Ambos os conjuntos de dados consistem de 108 exemplos, cada um composto pelos valores de 12 atributos e o valor objetivo. A variável objetivo (TYPE) é o tipo de ponte a ser construída, que pode ser: treliças simples, contínuas, cantilever, em arco, pênsil, ou madeira. Os outros atributos determinam especificações da ponte, descritas a seguir:

1. IDENTIF: identificador do exemplo;
2. RIVER: rio que a ponte atravessa (Allegheny, Ihoa, Monongahela ou Youghigheny);
3. LOCATION: localização da ponte;
4. ERECTED: época em que a ponte foi construída;
5. PURPOSE: propósito da construção;
6. LENGTH: comprimento total da ponte;
7. LANES: número de vias;
8. CLEAR-G: especifica se um requerimento de vão vertical para navegação foi imposto no projeto ou não;
9. T-OR-D: localização vertical da pista na ponte: dentro da estrutura ou no topo;
10. MATERIAL: material com o qual a ponte foi construída;
11. SPAN: comprimento do vão principal da ponte;
12. REL-L: comprimento relativo do vão principal da ponte em relação ao comprimento total;

O atributo IDENTIF foi removido antes do processo de aprendizado, uma vez que ele é obviamente um atributo irrelevante e sua presença poderia prejudicar o aprendizado.

3.2. Metodologia Experimental

A fim de determinar qual algoritmo fornece a melhor acurácia na solução do problema do projeto de pontes, foi usado o Weka (Witten e Frank, 2005), um programa de aprendizado de máquinas *open source* que contém vários algoritmos de aprendizado disponíveis para teste (http://www.cs.waikato.ac.nz/ml/weka/).

Usar só um par de conjuntos de treinamento e de teste não fornece uma estimativa do desempenho estatisticamente significativa. Se o conjunto de dados é grande, pode ser dividido em vários pares de conjuntos de treinamento e de teste independentes. O desempenho do algoritmo é medido fazendo-se a média das performances para cada um desses pares. Se o conjunto de dados não é muito grande, normalmente usa-se o método chamado validação cruzada. Usando validação cruzada, para construir cada par de conjuntos de treinamento e de teste, o conjunto de dados é dividido aleatoriamente em n conjuntos disjuntos de igual tamanho (chamados de partições), T_1, ..., T_n. Então são conduzidas n rodadas. Em cada rodada, o conjunto de teste é T_i e o conjunto de treinamento é a união de todos os outros Tj, $j \neq i$. O desempenho então é dado pela média das performances das n rodadas.

Se o programa se adaptar demais aos dados de treinamento (*overfitting*) ele não generalizará bem, ou seja, haverá um grande erro de teste (Hastie, 2001). Em contraste, se o programa não for treinado o suficiente, haverá subadequação (*underfitting*) e novamente resultará em pobre desempenho. Entre esses dois extremos existe uma complexidade de modelo ótima que fornece o erro de teste mínimo. Uma forma de evitar *overfitting* é usar um conjunto de validação. Parte dos exemplos é separada para verificar o desempenho do sistema durante o treinamento. Assim, o treinamento pára quando começar a haver queda de desempenho com o conjunto de validação.

3.3. Algoritmos Utilizados

Vinte e oito algoritmos foram utilizados nesta pesquisa, de vários paradigmas de aprendizado. A Tabela (1) apresenta uma lista com os 28 algoritmos utilizados e seus respectivos paradigmas de aprendizado. Em seguida, serão apresentados brevemente os paradigmas de aprendizado.

Tabela 1. Algoritmos e paradigmas de aprendizado.

Paradigma	Algoritmo
Redes Bayesianas	*Bayes Net*
	Naive Bayes
	Naive Bayes Simple
	Naive Bayes Updateable
Aprendizado de Funções	*Logistic*
	Multilayer Perceptron
	RBF Network
	Simple Logistic
	SMO
Aprendizado Baseado em Instâncias	*IB1*
	IBk
	*K**
	LWL
Aprendizado Baseado em Regras	*Conjunctive Rule*
	Decision Table
	JRip
	NNge
	OneR
	PART
	Ridor
	ZeroR
Árvores de Decisão	*Decision Stump*
	J48
	LMT
	NB Tree
	Random Forest
	Random Tree
	REP Tree

3.3.1. Redes Bayesianas

Redes Bayesianas são algoritmos de Aprendizado de Máquinas capazes de fornecer predições associadas a valores de probabilidades. Dentre outras vantagens, elas permitem o tratamento de fenômenos associados ao tempo, considerando a dependência temporal dos dados. O estudo de Redes Bayesianas é complexo e exige um conhecimento razoável sobre probabilidade.

Entretanto, o classificador Naive Bayes é uma simplificação conceitualmente fácil de ser entendida e de ser aplicada. O Classificador Naive Bayes aprende a partir dos dados de treinamento a probabilidade condicional de cada atributo dado o valor da classe. A classificação é feita aplicando-se a regra de Bayes para calcular a probabilidade de cada classe, dados os valores dos atributos no caso-teste, e escolhendo a que resulta em maior probabilidade:

$$c_{caso-teste} = \arg\max_{c} P(c \mid a_1, a_2, ..., a_n) \tag{1}$$

onde c é a classe e a_1, a_2, ..., a_n são os valores dos atributos.

Esse cálculo só é possível devido à forte suposição de que todos os atributos são independentes dada a classe. Embora essa suposição raramente seja verdade, o classificador Naive Bayes alcança performances surpreendentes. Portanto, após a aplicação da regra de Bayes e assumindo a independência dos atributos dada a classe:

$$c_{caso-teste} = \arg\max_{c} P(c) \cdot P(a_1 \mid c) \cdot P(a_2 \mid c) \cdot ... \cdot P(a_n \mid c) \tag{2}$$

O cálculo das probabilidades é feito pela simples contagem de exemplos no conjunto de treinamento:

$$P(c) = \frac{|S_c|}{|S|} \quad , \quad P(a_i, c) = \frac{|S_{a_i,c}|}{|S|} \tag{3}$$

onde S é o conjunto de treinamento, S_c é o conjunto de instâncias do conjunto de treinamento que tem classe c, e $S_{ai,c}$ é o conjunto de instâncias que tem classe c e o i-ésimo atributo é a_i. Pela regra de Bayes:

$$P(a_i \mid c) = \frac{P(a_i, c)}{P(c)} = \frac{|S_{a_i,c}|}{|S|} \cdot \frac{|S|}{|S_c|} = \frac{|S_{a_i,c}|}{|S_c|} \tag{4}$$

3.3.2. Aprendizado de Funções

Os coeficientes das funções são aprendidos de modo que, dados os valores dos atributos, o valor objetivo é retornado. Os algoritmos mais famosos de aprendizado de funções são as Redes Neurais Artificiais. Elas constituem uma abordagem robusta para aproximar valores objetivos tanto discretos quanto contínuos. São em parte inspiradas pela observação de que sistemas de aprendizado biológicos são constituídos de redes complexas de neurônios interconectados. Analogamente, uma rede neural artificial é um conjunto de unidades simples interconectadas, onde cada unidade recebe um número de entradas e produz uma única saída, que pode se tornar entrada para outras unidades. Normalmente as redes neurais são baseadas em unidades chamadas *perceptrons* (Fig. (7)).

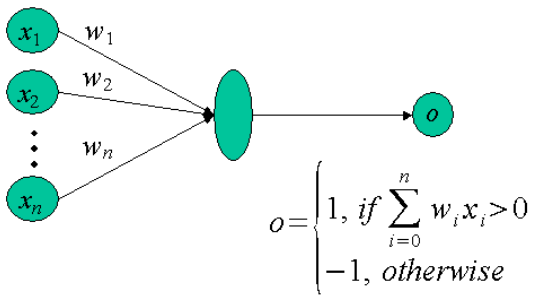

Figura 7. Perceptron.

O perceptron recebe um vetor de entradas, calcula uma combinação linear dessas entradas e então gera a saída 1 se o resultado for maior que 0, e -1 caso contrário. A contribuição de cada entrada x_i na saída é determinada pelo peso w_i. Aprender com um perceptron resume-se a calcular os pesos associados a cada uma das entradas. Para isso é usada a regra de treinamento do perceptron:

$$w_i \leftarrow w_i + \Delta w_i$$
$$\Delta w_i = \eta \cdot (t - o) \cdot x_i \tag{5}$$

onde *t* é valor objetivo do exemplo, *o* é a saída do perceptron e η é a taxa de aprendizado.

Um único perceptron tem capacidade de aprendizado muito limitada. Por isso, em geral, usam-se redes com múltiplas camadas. Com exceção das entradas e da saída, as demais camadas de uma rede neural multicamadas são chamadas de camadas escondidas. Tais camadas são compostas por outro tipo de neurônio artificial: a unidade sigmóide. Com uma única, suficientemente grande camada escondida, é possível representar qualquer função contínua das entradas e com duas camadas, até funções descontínuas podem ser representadas. O método de treinamento mais comum para uma rede neural multicamadas chama-se retropropagação e consiste em usar os erros nas saídas para atualizar os todos os pesos, do fim até o início da rede.

3.3.3. Aprendizado Baseado em Instâncias

A idéia chave do aprendizado baseado em instâncias é que a classe de um caso-teste é provavelmente a mesma de exemplos com valores de atributos similares ao dele. O aprendizado baseado em instâncias também é conhecido como *nearest-neighbor* (vizinho mais próximo) ou aprendizado preguiçoso, já que o aprendizado consiste apenas em armazenar os exemplos de treinamento. Um caso-teste é classificado encontrando-se os exemplos mais próximos a ele de acordo com alguma medida de distância, e atribuindo a classe de acordo com tais exemplos.

O algoritmo de aprendizado baseado em instâncias mais conhecido é o *KNN* (*K Nearest Neighbors*, *k* vizinhos mais próximos) que consiste em atribuir ao caso-teste a classe que receber a maioria de votos dentre os *k* vizinhos mais próximos, de forma que quanto mais próximo um vizinho estiver do caso-teste, maior será o valor de seu voto. Na prática, tem-se bons resultados com *k* entre 5 e 10.

Para identificar os vizinhos mais próximos de um caso-teste, é necessária uma métrica de distância. A distância pode ser, por exemplo, a soma das distâncias entre os valores dos atributos no exemplo e no caso-teste. Para atributos numéricos, a distância entre dois valores x_i e x_j pode ser o valor absoluto de sua diferença, normalizado para evitar que alguns atributos tenham mais peso do que outros na distância total:

$$d(x_i, x_j) = \left| \frac{x_i - x_j}{x_{\max} - x_{\min}} \right| \tag{6}$$

Para atributos simbólicos, usa-se uma simples métrica de envoltório:

$$d(x_i, x_j) = \begin{cases} 0 & se\ \ i = j \\ 1 & caso\ contrário \end{cases} \tag{7}$$

3.3.4. Aprendizado Baseado em Regras

São algoritmos que determinam um conjunto de regras que formam um relacionamento entre os atributos e as classes. Dada uma classe, seus membros no conjunto de treinamento são chamados exemplos positivos, e o restante são exemplos negativos. Os algoritmos de indução de regras tipicamente empregam uma abordagem de cobertura de conjunto: a definição de uma classe é formada construindo-se uma regra que cubra muitos exemplos positivos e poucos ou nenhum negativo, então “separando” os exemplos recentemente cobertos e começando outra vez no restante.

Uma regra é composta de um consequente e de uma parte antecedente ou corpo. O consequente é a classe predita. O corpo é uma conjunção de antecedentes, onde cada um deles é uma condição que envolve um único atributo. Para atributos simbólicos, essa condição é um simples teste de igualdade. Para atributos numéricos, a condição é tipicamente a inclusão em um intervalo. Diz-se que uma regra cobre um exemplo, e reciprocamente o exemplo satisfaz a regra, se todas as condições na regra forem verdadeiras para o exemplo. Estratégias para aprender uma regra:

- *Top-Down*: Começa com a regra mais geral; repetidamente adiciona condições que eliminam os exemplos negativos e mantêm os positivos; pára quando só positivos forem cobertos.
- *Bottom-Up*: Começa com uma regra específica; repetidamente retira condições para cobrir mais exemplos positivos; pára quando qualquer generalização cobrir exemplos negativos.

A escolha de quais condições devem ser adicionadas ou retiradas da regra é dada segundo uma heurística de avaliação, que pode ser a acurácia da regra no conjunto de treinamento:

$$H(n_+, n_-) = \frac{n_+}{n_+ + n_-} \tag{8}$$

onde *n+* é o número de exemplos positivos cobertos pela regra e *n-* é o número de exemplos negativos cobertos pela regra. A classificação de um caso-teste é realizada fazendo-se o *match* de cada regra com ele, e selecionando aquelas que ele satisfizer.

3.3.5. Árvores de Decisão

É um dos métodos de aprendizado mais usados e mais práticos, usado em problemas onde o valor objetivo é discreto. Cada nó na árvore especifica o teste de algum atributo e cada ramo originado em tal nó corresponde a um dos possíveis valores para esse atributo.

Para treinar uma árvore de decisão, a cada passo escolhe-se o atributo que sozinho melhor classifique os exemplos de treinamento. O primeiro atributo escolhido se torna o nó raiz da árvore. Para cada valor possível do atributo do nó raiz é criado um nó descendente. Para escolher o atributo correspondente a cada nó descendente, são considerados apenas os exemplos de treinamento cujo valor do atributo na raiz seja aquele representado pelo ramo. O procedimento é então repetido para cada nó descendente.

A medida usada para decidir qual é o melhor atributo a ser usado a cada passo é chamada ganho de informação. Dada uma coleção de exemplos *S*, o ganho de informação de um atributo *A* é dado por:

$$Ganho(S,A) = Entropia(S) - \sum_{v \in Valores(A)} \frac{|S_v|}{|S|} Entropia(S_v) \tag{9}$$

onde *Valores*(*A*) é o conjunto de todos os valores possíveis de *A*, S_v é o subconjunto de *S* onde *A* tem valor *v* e a *Entropia* é dada por:

$$Entropia(S) = \sum_{i=1}^{c} p_i \log_2 p_i \tag{10}$$

onde *c* é o número de classes e p_i é a proporção de *S* que pertence à classe *i*.

Um caso-teste é classificado começando no nó raiz da árvore, testando o atributo especificado por esse nó, e então descendo pelo ramo correspondente ao valor do atributo no caso-teste. Esse procedimento é repetido para os demais nós no caminho até se chegar a uma folha, que fornece a classificação do caso-teste.

3.4. Resultados

As Tabelas (2) e (3) apresentam a acurácia preditiva alcançada pelos 28 algoritmos para os conjuntos de dados original e modificado, respectivamente. Pode-se notar que, dos 10 melhores algoritmos para ambas as versões do conjunto de dados, 9 são os mesmos, alterando as posições. As Tabelas (4) e (5) mostram o tempo médio de treinamento de cada algoritmo para ambos os conjuntos de dados.

Os resultados foram diferentes para as duas versões do conjunto de dados. Para a versão original, o melhor algoritmo foi o *J48*, o mais simples algoritmo de árvore de decisão. Para a versão discretizada do conjunto de dados, 2 algoritmos obtiveram a mesma acurácia preditiva: *LMT* e *Simple Logistic*. Nesse caso, o tempo de treinamento foi usado como critério de decisão. Analisando a Tab. (5), é possível verificar que *Simple Logistic* é quase 3 vezes mais rápido que *LMT*, e portanto é uma melhor escolha.

Tabela 2. Ranking de acurácia preditiva para o conjunto de dados original.

Algoritmo	Acurácia	Algoritmo	Acurácia
J48	71,4286%	*IBk*	62,8571%
Random Forest	70,4762%	*IB1*	62,8571%
Naive Bayes	70,4762%	*Logistic*	62,8571%
Naive Bayes Updateable	70,4762%	*LWL*	62,8571%
LMT	69,5238%	*RBF Network*	60,9524%
Naive Bayes Simple	69,5238%	*REP Tree*	60,9524%
Multilayer Perceptron	69,5238%	*Decision Table*	60,0000%
Simple Logistic	69,5238%	*JRip*	59,0476%
SMO	67,6190%	*Ridor*	59,0476%
Bayes Net	65,7143%	*Decision Stump*	57,1429%
*K**	64,7619%	*Random Tree*	57,1429%
NB Tree	64,7619%	*Conjunctive Rule*	57,1429%
NNge	63,8095%	*OneR*	55,2381%
PART	62,8571%	*ZeroR*	41,9048%

Tabela 3. Ranking de acurácia preditiva para o conjunto de dados modificado.

Algoritmo	Acurácia
LMT	69,52%
Simple Logistic	69,52%
NB Tree	68,57%
SMO	68,57%
J48	67,62%
Multilayer Perceptron	66,67%
Naive Bayes	65,71%
Naive Bayes Simple	65,71%
Naive Bayes Updateable	65,71%
Bayes Net	64,76%
*K**	63,81%
Random Forest	63,81%
IBk	62,86%
RBF Network	61,90%

Algoritmo	Acurácia
JRip	61,90%
PART	61,90%
REP Tree	60,95%
LWL	60,95%
NNge	60,00%
Random Tree	60,00%
IB1	59,05%
Logistic	58,10%
Decision Table	58,10%
Conjunctive Rule	57,14%
Decision Stump	57,14%
OneR	53,33%
Ridor	49,52%
ZeroR	41,90%

Tabela 4. Ranking de tempo de treinamento para o conjunto de dados original.

Algoritmo	Tempo
SMO	6,86
Multilayer Perceptron	6,09
NB Tree	3,69
LMT	3,38
Simple Logistic	1,33
Logistic	1,05
RBF Network	0,36
PART	0,28
Random Forest	0,27
Ridor	0,17
JRip	0,16
Decision Table	0,13
NNge	0,11
Conjunctive Rule	0,09

Algoritmo	Tempo
REP Tree	0,05
J48	0,03
Naive Bayes	0,03
Random Tree	0,02
Bayes Net	0,02
OneR	0
IBk	0
ZeroR	0
Decision Stump	0
*K**	0
IB1	0
Naive Bayes Updateable	0
Naive Bayes Simple	0
LWL	0

Tabela 5. Ranking de tempo de treinamento para o conjunto de dados modificado.

Algoritmo	Tempo
Multilayer Perceptron	12,16
LMT	6,97
SMO	5,83
RBF Network	4,44
NB Tree	2,70
Simple Logistic	2,39
Logistic	1,25
Random Forest	0,09
JRip	0,05
Decision Table	0,05
Ridor	0,05
Bayes Net	0,02
NNge	0,02
PART	0

Algoritmo	Tempo
IBk	0
Random Tree	0
REP Tree	0
ZeroR	0
OneR	0
J48	0
Decision Stump	0
*K**	0
IB1	0
Naive Bayes Simple	0
Naive Bayes	0
Conjunctive Rule	0
LWL	0
Naive Bayes Updateable	0

Comparando os resultados para ambos os conjuntos de dados, pode ser visto que a discretização dos atributos numéricos não é uma boa opção, uma vez que o algoritmo *J48* alcançou, para o conjunto de dados original, uma performance mais alta que *Simple Logistic* alcançou para o conjunto de dados modificado, e com tempo médio de treinamento menor. Portanto, pôde-se concluir que o melhor algoritmo para este problema é a árvore de decisão *J48*, aplicada ao conjunto de dados original.

4. CONCLUSÃO

Usando o sistema Weka, foi possível determinar que o melhor método para se definir o tipo ideal de ponte a ser construída é a árvore de decisão *J48*, usando o conjunto de dados com atributos numéricos. Embora a acurácia preditiva pareça baixa (71%), deve-se notar que a resposta fornecida pelo algoritmo deveria ser usada como suporte no processo de projeto, mas a análise final e subsequente decisão deve ser tomada por um expert.

A baixa acurácia pode ser resultado de *underfitting*: apenas 108 exemplos estavam disponíveis e só 97 foram usados para treinamento. Coletando mais exemplos reais em outras cidades, com maior detalhamento dos atributos, é possível que um dos algoritmos tenha um resultado mais favorável.

5. AGRADECIMENTOS

Esta pesquisa foi financiada parcialmente pela CAPES e pela FAPERJ.

6. REFERÊNCIAS

Asuncion, A. and Newman, D., 2007, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, USA. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

EMPE – Exploração Multidisciplinar de Problemas de Engenharia, 2007, "Pontes" [http://empe.fe.up.pt/].

Flickr, 2009, "Kingston-Rhinecliff Bridge over the Hudson River" [http://www.flickr.com/photos/jag9889/].

Harington, G. D., 2006, "The General Hertzog Bridge, a truss bridge over Orange River at Aliwal North, South Africa" [http://en.wikipedia.org/wiki/File:General_Hertzog_Bridge_over_Orange_River_at_Aliwal_North.jpg].

Hastie, T., Tibshirani, R. and Friedman, J., 2001, The Elements of Statistical Learning - Data Mining, Inference, and Prediction, Springer Series in Statistics.

ImageShack – Online Media Hosting, 2010, "Ponte Hercílio Luz" [http://imageshack.us/].

ISI – Indian Statistical Institute, 2010, "Forum for Information Retrieval Evaluation" [http://www.isical.ac.in/].

Mitchell, T. M., 1997, Machine Learning, McGraw-Hill.

Panoramio, 2010, "Acueducto Segovia" [http://www.panoramio.com/].

Reich, Y. and Fenves, S. J., 1992, "Inductive learning of synthesis knowledge", International Journal of Expert Systems: Research and Applications, Vol. 5, No. 4, pp. 275-297.

Reich, Y., 1990, "Converging to "Ideal" Design Knowledge by Learning", Proceedings of the First International Workshop on Formal Methods in Engineering Design, Colorado Springs, CO, pp. 330-349.

Tumblr, Inc., 2010 [http://www.tumblr.com/].

Witten, I. H. and Frank, E., 2005, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann.

7. DIREITOS AUTORAIS

BRIDGE DESIGN USING MACHINE LEARNING ALGORITHMS

Aloísio Carlos de Pina, aloisiopina@bol.com.br [1]
Vitor de Souza Colimodio, colimodio@poli.ufrj.br [2]
Adriano Armani da Silva, adrianoarmani@poli.ufrj.br [2]
Armando Carlos de Pina Filho, pina-filho@deg.ee.ufrj.br [3]

[1]UFRJ, COPPE, Civil Engineering Program, CEP 21945-970, Rio de Janeiro, RJ, Brazil
[2]UFRJ, Polytechnic School, Department of Civil Engineering, CEP 21945-970, Rio de Janeiro, RJ, Brazil
[3]UFRJ, Polytechnic School, Urban Engineering Program, CEP 21949-909, Rio de Janeiro, RJ, Brazil

Abstract*. Bridge design is a very complex process. It starts with the search for potential locations for a bridge, along with some drafts which need public approval. The designers need to exercise their judgement concerning many aspects, such as aesthetics, cost, load determination, failure modes, constructability and maintainability. The objective of this research is to show the application of modern Machine Learning techniques to the decision-making process of a bridge design, based on specification properties. The data set used was provided by Professor Yoram Reich, from the Department of Civil Engineering and Engineering Design Research Center, Carnegie Mellon University, United States. The obtained data set was pre-processed in order to identify and eliminate the variables irrelevant to the learning process. The learning algorithms were selected and then an extensive experimental evaluation was performed. The results of the experiments were compared by means of statistical tests to evaluate their precision and significance, in order to determine the model best suited to the problem.*

Keywords*: bridges, design decisions, Machine Learning*