Seleção de Modelos de Séries Temporais Utilizando Meta-Protótipos

Ricardo B. C. Prudêncio , Teresa B. Ludermir , Francisco de A. T. de Carvalho

[1]Centro de Informática, Universidade Federal de Pernambuco
Caixa Postal 7851, Cidade Universitária, Recife-PE, Brasil, 50.732-970

{rbcp, tbl, fatc}@cin.ufpe.br

Abstract. *The selection of a good model for forecasting a time series is a task that involves experience and knowledge. A promising approach to acquire knowledge for this task is the use of machine learning algorithms. In this work, we proposed the use of a novel learning algorithm, the Meta-Prototypes (MP), for the model selection problem. This algorithm is related to Symbolic Data Analysis, which is a new area in the field of knowledge discovery. In our work, the MP algorithm was used in a case study and it was compared to some traditional learning algorithms. So far, the MP algorithm obtained the lowest selection error among all the tested algorithms.*

Resumo. *A seleção de um bom modelo para prever uma série temporal é uma tarefa que envolve experiência e conhecimento. Uma abordagem promissora para adquirir conhecimento para essa tarefa é o uso de algoritmos de aprendizado de máquina. Neste trabalho, propomos o uso de um algoritmo recente, os Meta-Protótipos (MP), para a seleção de modelos. Esse algoritmo está relacionado com a Análise de Dados Simbólicos, uma nova área no campo de descoberta de conhecimento. No nosso trabalho, o algoritmo MP foi usado em um estudo de caso e comparado com alguns algoritmos de aprendizado tradicionais. Até o momento, o algoritmo MP obteve o menor erro de seleção dentre os algoritmos avaliados.*

1. Introdução

A previsão de séries temporais tem sido usada em diversas situações no mundo real para auxiliar processos de planejamento e tomada de decisão **?**. Diversos modelos de previsão podem ser usados para prever uma dada série temporal. Contudo, evidências empíricas indicam que não existe um único modelo que seja o melhor para todos os casos **?**. Selecionar o modelo mais adequado para prever uma série, dentre um conjunto de modelos candidatos, pode ser uma tarefa difícil dependendo dos candidatos e das características da série em questão.

Uma abordagem capaz de formalizar conhecimento para ser usado na seleção de modelos é o desenvolvimento de sistemas especialistas **?**. Nesse contexto, cada regra fornecida pelo especialista associa características da série temporal ao melhor modelo para previsão. A principal desvantagem desses sistemas é que o seu processo de aquisição de conhecimento depende de especialistas humanos que são freqüentemente caros e pouco disponíveis. Uma alternativa interessante nesses casos é o uso de algoritmos de aprendizado de máquina **?**, que tratam a seleção de modelos como um problema de classificação, onde o atributo de classe representa o melhor modelo de previsão e os atributos preditores da classe são características das séries temporais.

No nosso trabalho, propomos o uso de um novo algoritmo de aprendizado, os Meta-Protótipos (MP) **?**, para selecionar dentre dois modelos de séries temporais: o Alisamento Exponencial Simples **?** e as redes neurais TDNN (*Time-Delay Neural Network*) **?**. O algoritmo de MP foi desenvolvimento originalmente dentro da área de Análise de Dados Simbólicos **?**, que é uma nova área dentro do campo de descoberta de conhecimento.

O algoritmo MP foi comparado com dois algoritmos de aprendizado tradicionais, as árvores de decisão e o algoritmo de vizinhos mais próximos (k-NN), para a mesma tarefa de seleção. O erro de seleção obtido pelo algoritmo MP (34.34%) foi menor que os erros obtidos tanto pelo algoritmo k-NN (45.45%) como pelas árvores de decisão (37.37%).

Na seção 2, apresentamos algumas abordagens para seleção de modelos. A seção 3 traz uma breve descrição do algoritmo MP assim como o estudo de caso abordado no trabalho. Na seção 4, apresentamos os experimentos realizados e resultados obtidos. Finalmente, na seção 5, tecemos algumas considerações finais.

2. Seleção de Modelos de Séries Temporais

Existem diversas técnicas que podem ser usadas para se prever uma série temporal, tais como a família de modelos de alisamento exponencial **?**, os modelos de Box e Jenkins **?** e diversos modelos de redes neurais **?**. Em geral, o usuário interessado na previsão de um conjunto de séries tem um número limitado de modelos disponíveis para escolher. Dado um conjunto de modelos candidatos, o usuário pode ou escolher um único modelo para todas as séries, ou escolher um modelo adequado para cada série a ser prevista. A primeira estratégia é chamada de método de seleção agregada enquanto que a segunda é chamada de método de seleção individual **?**. Embora métodos agregados sejam simples de implementar, existem evidências empíricas de que não existe um modelo que se comporte melhor que os outros para todas as séries temporais **?**. Esse fato motivou diversos autores a desenvolver métodos individuais, comumente associando características das séries com o modelo mais adequado.

Uma maneira de formalizar conhecimento útil para seleção de modelos é usando sistemas especialistas, tais como o sistema Rule-Based Forecasting **?**. Nesse trabalho, os autores implementaram um sistema especialista com 99 regras usadas para definir pesos para os seguintes modelos de previsão: random walk, regressão linear, alisamento exponencial linear de Holt e alisamento exponencial linear de Brown. A base de regras foi desenvolvida a partir das sugestões fornecidas por cinco especialistas humanos durante a análise de problemas reais. Os autores usaram, como antecedentes das regras, características das séries temporais tais como, descontinuidades no nível da série, tendência básica significativa, outliers, dentre outras. O conseqüente de cada regra modificava os pesos associados a cada modelo.

Embora os sistemas especialistas possam expressar conhecimento de uma maneira prática e reutilizável, o seu processo de aquisição de conhecimento depende de especialistas humanos, que são freqüentemente caros e pouco disponíveis **?**. Nesses casos, os algoritmos de aprendizado de máquina podem ser uma alternativa interessante. Esses algoritmos são capazes de adquirir conhecimento de forma automática a partir de dados, reduzindo a necessidade de especialistas e proporcionando um ganho potencial de desempenho **?**.

O uso de algoritmos de aprendizado para seleção de modelos foi proposto originalmente em **?** e adotado em diferentes outros trabalhos **?**, **?**, **?**. Em geral, esses trabalhos

tratam a seleção de modelos como um problema de classificação, e usam um algoritmo de aprendizado como classificador. Cada exemplo de treinamento consiste de uma série temporal descrita por um conjunto de características, associada a um atributo de classe que representa o melhor modelo candidato para prever a série. Um conjunto de tais exemplos serve de base para o treinamento do algoritmo de aprendizado.

3. A Abordagem de Meta-Protótipos

O trabalho descrito aqui usa um novo algoritmo de aprendizado, os Meta-Protótipos (MP) **?**, para o problema de seleção de modelos de séries temporais. Esse algoritmo foi proposto e desenvolvido dentro de uma nova área em descoberta de conhecimento, a Análise de Dados Simbólicos **?**, que fornece novas ferramentas para gerenciar informações representadas por diversos tipos de dados, tais como variáveis multivaloradas e intervalares. O algoritmo MP foi usado anteriormente em **?** para filtragem de informação, e os resultados obtidos foram bem sucedidos tanto em termos de precisão como de tempo de execução.

3.1. Descrição do algoritmo

Nós apresentamos aqui os detalhes de implementação do algoritmo MP para resolver o problema de seleção de modelos.

Seja Ω um conjunto de exemplos de treinamento, onde cada exemplo representa uma série temporal descrita por um conjunto de características Y_i e associada a uma $Classe \in c_1, ..., c_m$. Cada classe representa um modelo candidato para prever a série temporal.

O algoritmo MP cria para cada classe c_k uma descrição simbólica, chamada *meta-protótipo*. Essa descrição consiste de um vetor de variáveis modais $Y_{c_k,i}$ que representam uma distribuição de probabilidade, histograma, frequência ou pesos associados aos valores do atributo Y_i levando em consideração apenas os exemplos de Ω pertencentes à classe c_k. Dado um novo exemplo ω a ser classificado, uma função de *matching* mede a similaridade entre o novo exemplo e o meta-protótipo de cada classe. O novo exemplo é então associado à classe do meta-protótipo mais similar.

A definição das variáveis modais $Y_{c_k,i}$ depende do tipo do atributo (booleano ou numérico) do qual a variável agrega informação. Para atributos booleanos (assumindo valor 0 ou 1), definimos a variável $Y_{c_k,i}$ da seguinte maneira:

$$Y_{c_k,i} = [1(p^1_{c_k,i}), 0(p^0_{c_k,i})] \tag{1}$$

onde $p^1_{c_k,i}$ e $p^0_{c_k,i}$ representam a frequência de exemplos da classe c_k nos quais o atributo Y_i recebe valor 1 e valor 0, respectivamente. Dado o conjunto Ω, os pesos $p^1_{c_k,i}$ e $p^0_{c_k,i}$ são calculados como segue:

$$p^1_{c_k,i} = P(Y_i = 1|Classe = c_k, \Omega) = \frac{\#(Y_i = 1, Classe = c_k|\Omega)}{\#(Classe = c_k|\Omega)} \tag{2}$$

$$p^0_{c_k,i} = P(Y_i = 0|Classe = c_k, \Omega) = \frac{\#(Y_i = 0, Classe = c_k|\Omega)}{\#(Classe = c_k|\Omega)} \tag{3}$$

O matching de um exemplo ω com o meta-protótipo da classe c_k, levando em consideração o atributo Y_i, é definido como o peso associado ao valor do atributo Y_i em ω:

$$Match_i(\omega, c_k) = p_{c_k,i}^{Y_i(\omega)} \tag{4}$$

Como no caso booleano, as variáveis modais $Y_{c_k,i}$ para atributos numéricos são definidas como na equação 1. Contudo, no caso numérico, o peso $p_{c_k,i}^1$ representa a frequência de exemplos da classe c_k nos quais o valor do atributo Y_i é maior ou igual a um limiar pré-definido T_i, e o peso $p_{c_k,i}^0$ é a frequência de exemplos da classe nos quais o valor do atributo é menor que o limiar. Assim, a variável modal $Y_{c_k,i}$ descreve um novo atributo booleano Y_i' que recebe valor 1 se $Y_i \geq T_i$ e 0 caso contrário. Os pesos são calculados como segue:

$$p_{c_k,i}^1 = P(Y_i' = 1|Classe = c_k, \Omega) = \frac{\#(Y_i \geq T_i, Classe = c_k|\Omega)}{\#(Classe = c_k|\Omega)} \tag{5}$$

$$p_{c_k,i}^0 = P(Y_i' = 0|Classe = c_k, \Omega) = \frac{\#(Y_i < T_i, Classe = c_k|\Omega)}{\#(Classe = c_k|\Omega)} \tag{6}$$

O valor do limiar T_i foi definido de forma a reduzir a entropia do atributo Y_i em relação ao atributo classe. Nós testamos todos os valores th do atributo numérico Y_i observados nos exemplos de Ω, e escolhemos o valor que maximizasse o *ganho de informação*, que mede a redução na entropia.

$$T_i = argmax_{th \in Y_i(\Omega)} Ganho(Y_i, th, \Omega) \tag{7}$$

onde

$$\begin{aligned} Ganho(Y_i, th, \Omega) \quad = \quad & -P(Y_i \geq th|\Omega) * Ent(Classe|Y_i \geq th, \Omega) \\ & -P(Y_i < th|\Omega) * Ent(Classe|Y_i < th, \Omega) \end{aligned} \tag{8}$$

e

$$Ent(Classe|E, \Omega) = \sum_{c_k} -P(Classe = c_k|E, \Omega) log_2 P(Classe = c_k|E, \Omega) \tag{9}$$

O matching de um exemplo ω com o meta-protótipo da classe c_k, calculado no atributo Y_i, é definido como segue:

$$Match_i(\omega, c_k) = p_{c_k,i}^{Y_i'(\omega)} \tag{10}$$

O matching final do exemplo ω com o meta-protótipo da classe c_k é definido como a média das similaridades levando em consideração todos os atributos:

$$Match(\omega, c_k) = \frac{\sum_{i \in atributos} Match_i(\omega, c_k)}{\#\ of\ atributos} \tag{11}$$

Para classificar um novo exemplo, o algoritmo calcula o matching final para cada meta-protótipo, e retorna a classe que obtiver o maior valor.

3.2. Estudo de Caso

Neste trabalho, investigamos o uso do algoritmo MP na tarefa de selecionar entre um dos dois modelos de previsão: Alisamento Exponencial Simples (AES) **?** e a rede neural Time-Delay Neural Network (TDNN) **?**. Ambos os modelos foram usados para a previsão a curto prazo de séries sem tendência e sem sazonalidade. Essa tarefa já foi abordada em um trabalho anterior **?**, contudo usando árvores de decisão .

Para a geração de um conjunto de exemplos de treinamento, nós usamos 99 séries temporais disponíveis na *Time Series Data Library* **?**. Cada série foi descrita por um conjunto de 10 atributos usados comumente na literatura de análise de séries temporais (ver tabela 1).

Tabela 1. Atributos descritores das séries temporais.

Atributo	**Descrição**	**Tipo**
Y_1	Tamanho da série	Numérico
Y_2	Média das 5 primeiras autocorrelações	Numérico
Y_3	Coeficiente de variação	Numérico
Y_4	Valor absoluto do coeficiente de skewness	Numérico
Y_5	Valor absoluto do coeficiente de kurtosis	Numérico
Y_6	Presença de autocorrel. significativas	Booleano
Y_7	Significância da primeira autocorrel.	Booleano
Y_8	Significância da segunda autocorrel.	Booleano
Y_9	Significância da terceira autocorrel.	Booleano
Y_{10}	Teste de *turning points* para aleatoriedade	Booleano

Para cada série temporal, usamos os dois modelos para gerar as previsões a um passo dos últimos 30 pontos da série. Nós definimos o atributo classe para a série como sendo o modelo que obteve o menor erro absoluto médio de previsão nesses 30 pontos.

Na tabela 2, mostramos a distribuição de classes obtida para as 99 séries usadas para gerar o conjunto de exemplos. Também mostramos nessa tabela o erro de classificação obtido pelo classificador *inocente* (erro *default*) que sempre associa a um novo exemplo a classe com maior número de exemplos (no nosso caso, a classe $tdnn$). Esse classificador é útil somente nos casos onde não se tem conhecimento sobre como relacionar as características das séries aos modelos de previsão. Desta forma, o mínimo que se espera de um algoritmo de aprendizado é que o seu erro obtido seja menor que o erro default.

Tabela 2. Distribuição de classes

# de exemplos $tdnn$	52
# de exemplos aes	47
Erro default	47.47%

A tabela 3 mostra um exemplo dos meta-protótipos definidos para as classes aes e $tdnn$, assim como o processo de matching de uma nova instância ω com esses meta-protótipos. Na primeira e segunda coluna, podemos ver os meta-protótipos das classes aes e $tdnn$, respectivamente. Na terceira coluna, temos os limiares que foram usados para definir as variáveis modais das 5 primeiras características das séries. Na coluna seguinte, temos o novo exemplo a ser classificado. Na quinta coluna, temos os atributos booleanos Y_i' gerados a partir da discretização dos atributos numéricos. Em seguida, temos o cálculo do matching do novo exemplo com os meta-protótipos de aes e $tdnn$, respectivamente. O exemplo foi classificado como $tdnn$, uma vez que o seu matching (0.54) foi maior que o matching da classe aes (0.39).

Tabela 3: Matching da nova instância com os meta-protótipos das classes aes e $tdnn$.

Meta-Protótipo Classe aes ($Y_{aes,i}$)	Meta-Protótipo Classe $tdnn$ ($Y_{tdnn,i}$)	Limiares (T_i)	Exemplo (ω)	Variáveis $Y_i'(\omega)$	Match aes	Match $tdnn$
[1(0.2), 0(0.8)]	[1(0.4), 0(0.6)]	T_1=339	$Y_1(\omega)$=73	$Y_1'(\omega)$=0	0.8	0.6
[1(0.4), 0(0.6)]	[1(0.7), 0(0.3)]	T_2=5.36	$Y_2(\omega)$=8.2	$Y_2'(\omega)$=1	0.4	0.7
[1(0.3), 0(0.3)]	[1(0.4), 0(0.6)]	T_3=0.05	$Y_3(\omega)$=2.1	$Y_3'(\omega)$=1	0.3	0.4
[1(0.8), 0(0.2)]	[1(0.9), 0(0.1)]	T_4=0.44	$Y_4(\omega)$=0.1	$Y_4'(\omega)$=0	0.2	0.1
[1(0.5), 0(0.5)]	[1(0.4), 0(0.6)]	T_5=0.18	$Y_5(\omega)$=0.2	$Y_5'(\omega)$=1	0.5	0.4
[1(0.2), 0(0.8)]	[1(0.5), 0(0.5)]	-	$Y_6(\omega)$=1	-	0.2	0.5
[1(0.9), 0(0.1)]	[1(0.3), 0(0.7)]	-	$Y_7(\omega)$=0	-	0.1	0.7
[1(0.9), 0(0.1)]	[1(0.2), 0(0.8)]	-	$Y_8(\omega)$=0	-	0.1	0.8
[1(0.3), 0(0.7)]	[1(0.7), 0(0.3)]	-	$Y_9(\omega)$=1	-	0.3	0.7
[1(0.4), 0(0.6)]	[1(0.6), 0(0.4)]	-	$Y_{10}(\omega)$=0	-	0.6	0.4
				Média	**0.39**	**0.54**

4. Experimentos e Resultados

Nessa seção, apresentamos os experimentos e resultados obtidos pelo algoritmo MP na tarefa de seleção de modelos investigada no nosso trabalho.

Nós avaliamos o desempenho de classificação do algoritmo usando o procedimento *leave-one-out* **?**. Em cada passo, 98 exemplos são usados como conjunto de treinamento e o exemplo restante é usado para testar o algoritmo. Esse processo é repetido 99 vezes, usando em cada passo um exemplo diferente para teste. Usando o procedimento leave-one-out, o algoritmo MP obteve um número de 59 exemplos classificados corretamente de um conjunto de 99 exemplos de teste, o que representa um erro de 40.40% (ver tabela 4).

Para comparar o desempenho algoritmo MP, nós realizamos experimentos com o algoritmo IBk, que é a versão do algoritmo dos k-vizinhos mais próximos (*k-nearest neighbors, k-NN*) implementada no pacote Java WEKA **?**. Nós também comparamos os resultados do algoritmo MP com os resultados obtidos anteriormente em **?** usando o algoritmo J.48 implementado no pacote WEKA. Esse algoritmo é uma variação do algoritmo C4.5 proposto por Quinlan para indução de árvores de decisão **?**. Usando o procedimento leave-one-out, nós observamos que o erro de classificação obtido pelo algoritmo MP (40.40%) foi menor que o erro obtido pelo algoritmo IBk (45.45%) e maior que o erro obtido pelo algoritmo J48 (37.37%).

Tabela 4. Resultados do procedimento leave-one-out.

	MP	**J48**	**IBk**
Exemplos classificados corretamente	59	62	54
Exemplos classificados incorretamente	40	37	45
Erro de classificação	40.40%	37.37%	45.45%

Depois de analisar esses primeiros resultados, nós tentamos investigar a razão do baixo desempenho do algoritmo IBk na nossa tarefa. Embora o erro obtido pelo algoritmo IBk tenha sido menor que o erro default, essa diferença não foi significativa. Como destacado em **?**, uma das dificuldades dos algoritmos baseados em instâncias, tais como o algoritmo de k-NN, é que eles, em geral, usam todos os atributos descritores para medir similaridade entre instâncias. Desta forma, na presença de atributos irrelevantes, esses algoritmos podem apresentar um desempenho prejudicado. Um algoritmo de árvores de

decisões, por sua vez, tem um mecanismo interno de seleção de atributos que pode eventualmente eliminar atributos irrelevantes. De fato, nós observamos em nossos experimentos que alguns dos atributos das séries temporais não foram usados em algumas árvores induzidas pelo algoritmo J48. Baseado nessa evidência, acreditamos que o desempenho do algoritmo IBk poderia ser melhorado se um mecanismo de seleção de atributos tivesse sido usado.

Assim como o algoritmo IBk, o algoritmo MP usou todos os atributos para calcular uma medida de similaridade. Desta forma, podemos também esperar que o desempenho do algoritmo MP possa ser melhorado com seleção de atributos. Assim, a próxima etapa do nosso trabalho foi incorporar um mecanismo seleção de atributos ao algoritmo MP.

Existem duas abordagens gerais para seleção de atributos: *filters e wrappers* **?**. Na primeira, a relevância de cada atributo é medida através de estatísticas calculadas nos dados de treinamento, tais como entropia e correlação. A principal desvantagem dessa abordagem é que ela não leva em consideração o algoritmo de aprendizado que está sendo usado. Na segunda abordagem, uma busca é realizada no espaço de atributos. O algoritmo de aprendizado é executado usando diferentes subconjuntos de atributos, e o que obtiver os melhores resultados de treinamento é então selecionado. A principal desvantagem dessa abordagem é o tempo de execução, uma vez que o algoritmo é executado diversas vezes antes que um bom subconjunto de atributos seja selecionado. Se o algoritmo sendo usado for lento então o uso de wrappers se torna inviável. Apesar desta potencial desvantagem, decidimos usar a abordagem de wrappers uma vez que observamos nos experimentos anteriores que o algoritmo MP se apresentou eficiente no tempo. O treinamento rápido do algoritmo MP já havia sido constatado em **?**.

No nosso trabalho, aplicamos o algoritmo *Backward-Elimination*, que é uma das formas de wrappers mais usadas na literatura **?**. Nesse algoritmo, o ponto inicial de busca é o conjunto inteiro dos atributos. Em cada passo, novos pontos de busca são gerados eliminando um atributo diferente do ponto de busca atual. Todos esses novos subconjuntos são então avaliados, e que obtiver o melhor resultado no treinamento é então definido como o novo ponto de busca atual. Esse processo é repetido até que todos os atributos sejam eliminados.

Como podemos ver na tabela 5, o algoritmo MP com seleção de atributos obteve um erro de classificação de 34.34%, que foi menor que o erro obtido pelo algoritmo sem a seleção (40.40%). Comparado com os algoritmos IBk e J48, o algoritmo MP com seleção de atributos obteve o menor erro de classificação.

Observamos aqui que o tempo de execução do algoritmo com essa modificação aumentou em relação à implementação sem seleção de atributos. De fato, durante a seleção dos atributos, o algoritmo MP é executado para 55 subconjuntos de atributos. Entretanto, como a construção dos meta-protótipos requer pouco custo computacional, esse aumento no tempo de execução não foi tão drástico. O aumento no tempo de execução seria realmente drástico se a seleção de atributos tivesse sido realizada com o algoritmo IBk, que é bem menos eficiente no tempo que o algoritmo MP.

Tabela 5. Resultados da seleção de atributos aplicada ao algoritmo MP.

	Algoritmo MP (todos os atributos)	**Algoritmo MP (seleção de atributos)**
Exemplos classificados corretamente	59	65
Exemplos classificados incorretamente	40	34
Erro de classificação	40.40%	34.34%

5. Conclusão

Neste trabalho, nós aplicamos o algoritmo de Meta-Protótipos (MP) para o problema de seleção de modelos para previsão de séries temporais. Nos experimentos realizados, verificamos que o desempenho do algoritmo MP foi satisfatório quando comparado a outros algoritmos de aprendizado tradicionais. Nós observamos ainda que o algoritmo se mostrou sensível aos atributos usados no processo de treinamento. De fato, nós obtivemos uma melhoria significativa no desempenho de classificação quando um mecanismo de seleção de atributos foi aplicado ao algoritmo MP.

Como trabalhos futuros, nós planejamos realizar algumas modificações na implementação atual do algoritmo, tais como definir novas formas de construir as variáveis modais para atributos numéricos, e avaliar outras funções de matching. Além disso, outras características de séries temporais pode ser incluídas nos experimentos, assim como outros modelos de previsão. Outro ponto a investigar no futuro, é o valor do uso do algoritmo MP para fornecer rankings de modelos, em vez de retornar apenas o melhor modelo. Um ranking de modelos para uma dada série poderia ser gerado ordenando os valores de matching dos meta-protótipos com a série em questão.

References