

Estudo Prático para um Sistema de Previsão Autônoma de Chuva de Curto Período (Um Dia) Baseado em Lógica Fuzzy

Yuri Vasconcelos de Almeida Sá

UNESP Campus Sorocaba - Mestrando - Ciências Ambientais
Sorocaba/SP (e-mail: yuri@sa2.com.br).

Abstract: This work objective is a proposition of a Fuzzy System to estimate the possibility of rainfall on the next day (logistical process), using official government data of an automatic weather station located at Sorocaba/SP . This article formalize the theoretical basis for the creation of an autonomous offline device for local rainfall forecasting.

Resumo: Este trabalho tem como objetivo propor um sistema *fuzzy* para estimar a possibilidade de chuva no dia seguinte (processo logístico), utilizando dados oficiais de uma estação climática autônoma localizada no município de Sorocaba/SP. Este artigo formaliza a base teórica para a criação de um dispositivo autônomo, offline, de previsão de chuva local.

Keywords: Fuzzy Logic, Rainfall prediction, Machine Learning, Weather Automation, Weather Forecasting.

Palavras-chaves: Lógica Fuzzy, Previsão de chuva, Aprendizado de máquina, Automação Climática, Previsão do tempo

1. INTRODUÇÃO

O Brasil tem mais de 8 milhões de quilômetros quadrados de extensão (IBGE, 2012), o que coloca o país em 5o na lista de mais extensos do mundo, sendo o clima tropical predominante, fazendo com que chuvas sejam frequentes e volumosas. Embora a importância da chuva para os processos econômicos e sociais seja enorme, ela também pode trazer prejuízos para a atividade econômica, inviabilizar operações logísticas, retardar o comércio, dentre outras.

Este trabalho faz parte de um estudo amplo, para a construção de um sistema de estações autônomas, offline, de previsão de chuva utilizando dispositivos digitais autônomos, portanto a Lógica Fuzzy foi escolhida devido a facilidade de execução em plataformas digitais embarcadas.

Neste artigo apresenta um estudo prático de como implementar um sistema na linguagem de programação R. Devido a capacidade desta linguagem de produzir códigos estatísticos rápidos e pela velocidade de desenvolvimento.

2. FUNDAMENTAÇÃO TEÓRICA

O evento chuva pode ser previsto através de probabilidade (BERROCAL et al., 2008)(HAMILL, 2017), porém existe uma restrição a quais modelos e algoritmos podem ser executados de forma eficiente, tanto em plataformas digitais móveis quanto em computadores pessoais. Alguns modelos,

como redes neurais, exigem treino e iterações que são particularmente custosas do ponto de vista computacional e crescem exponencialmente em função do número de variáveis, tornando a operacionalização do modelo impossível (VANHOUCKE, 2011)

A lógica Fuzzy tem a capacidade de lidar com as imprecisões e relações de diversas variáveis de entrada e computá-las em variáveis de saída, através de funções de pertinência, podendo ser executadas com facilidade em computadores pessoais e sistemas embarcados (ALVES, 2018).

3. METODOLOGIA

3.1 Aquisição dos dados

Antes mesmo de introduzir o algoritmo ou sistema utilizado, temos que estabelecer os dados a serem analisados como fonte, neste caso, por aderência utilizo os dados da estação meteorológica automática código A713 operada pelo InMET/INPE/CPTEC, localizada no município de Sorocaba/SP, aberta em 21/08/2006, instalada na Latitude: -23.426035° e Longitude: -47.585552°.

Esta estação fornece dados de seus sensores em intervalos horários, contendo mínimo e máximo de todas as variáveis.

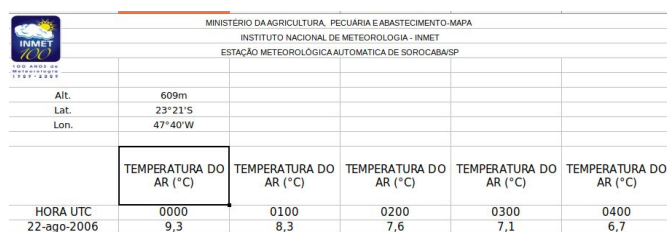
As variáveis (às vezes chamadas de parâmetros), nas estações automáticas no Brasil são: Temperatura do Ar, Umidade,

Temperatura do ponto de Orvalho, Pressão, Vento, Radiação e Chuva.

As quatro primeiras apresentam os dados instantâneos, Máximos e Mínimos de cada hora, enquanto Vento apresenta Velocidade, Direção e Rajada, Radiação apresenta somente o dado instantâneo e Chuva apresenta o acumulado da hora, somente.

Através da lei de acesso à informação (BRASIL, 2011) foram obtidos os dados desde sua instalação, em formato Excel.

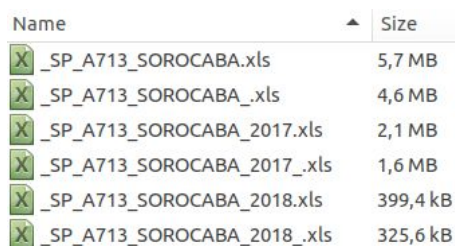
É importante observar a falta de padrão e método na apresentação dos dados, onde os dados eram organizados em um formato de dupla linha, a primeira contendo a variável e a hora da leitura na segunda (Figura 1). Impossibilitando a indexação dos dados através das linhas, exigindo que antes mesmo do sistema começar os dados deveriam ser organizados em um formato de Banco de Dados estruturado.



MINISTÉRIO DA AGRICULTURA, PECUÁRIA E ABASTECIMENTO - MAPA					
INSTITUTO NACIONAL DE METEOROLOGIA - INMET					
ESTATION METEOROLÓGICA AUTOMÁTICA DE SOROCABA/SP					
Alt.	609m				
Lat.	23°21'S				
Lon.	47°40'W				
	TEMPERATURA DO AR (°C)	TEMPERATURA DO AR (°C)	TEMPERATURA DO AR (°C)	TEMPERATURA DO AR (°C)	TEMPERATURA DO AR (°C)
HORA UTC	0000	0100	0200	0300	0400
22-ago-2006	9,3	8,3	7,6	7,1	6,7

Figura 1 - Planilha original enviada pelo INPE/CPTEC

Além da falta de organização dos dados contidos nos arquivos, também existia nos próprios arquivos, devido ao número excessivo de colunas que o método de armazenamento dos dados gerou, os arquivos enviados eram dois para cada período (Figura 2).



Name	Size
X _SP_A713_SOROCABA.xls	5,7 MB
X _SP_A713_SOROCABA.xls	4,6 MB
X _SP_A713_SOROCABA_2017.xls	2,1 MB
X _SP_A713_SOROCABA_2017.xls	1,6 MB
X _SP_A713_SOROCABA_2018.xls	399,4 kB
X _SP_A713_SOROCABA_2018.xls	325,6 kB

Figura 2 - Arquivos e seus detalhes enviados pelo INPE CPTEC

3.2 Preparação dos Dados

Por uma questão de agilidade e visualização, os arquivos foram concatenados através de um trabalho manual no software LibreOffice Calc, um manipulador de planilhas livre e de código aberto. Neste ponto do trabalho a estrutura continua a mesma, porém em um só arquivo em formato CSV.

Para que os dados sejam preparados para o sistema de classificação Fuzzy desenvolvido, foi feito um script R, em separado, cuja função é ler os dados concatenados manualmente em CSV e organizar em tuplas (X1..Xn,Y), passíveis de indexação por linha.

Os dados são lidos e colocados em Data Frame, de nome source_file (sem strings para fator).

Após ler os dados, as duas linhas que identificam as colunas por nome e hora são fundidos, criando nomes únicos para cada coluna, no formato VARIABELHORA, ex.: temp_ar1100 (temperatura do ar, às onze horas).

Uma tabela auxiliar é criada para indexar e identificar os parâmetros, contendo três colunas: Parâmetro, Hora, Índice. Sendo a primeira o nome do parâmetro, a segunda a hora e a terceira um índice numérico (ordem no arquivo) para localização.

Foi criada uma função cuja a entrada consiste do nome e da hora, e a resposta é o próprio índice da coluna, para construção da tabela definitiva através de laços for().

Para a criação da planilha definitiva, foi feita uma expansão da grade, calculando as horas em função dos dias e preenchida com NAs, já no formato de destino. Ou seja, cada hora de cada dia tem uma linha, e os parâmetros seguem nas colunas relacionadas.

Após criar o Data Frame, com o número desejado de linhas e colunas, já preenchidos os dias e horas, é criado um laço for que vai percorrer cada coluna após a segunda (a primeira é o DIA e a segunda a HORA), e preencher este dado pesquisando a coluna e linha utilizando uma função personalizada.

Os arquivos enviados pelo órgão apresentavam uma codificação em “linguagem natural”, que embora seja normatizada, é dependente da configuração do computador, pelo menos no formato das datas, que apresentava os meses em formato de string com 3 caracteres. A título de facilidade, foram disponibilizados os arquivos com a coluna DIA em português e inglês.

É importante observar que este processo é extremamente lento e dispendioso de recursos, tanto de CPU quanto de memória, a própria estrutura interna do R não favorece a utilização de dados unitários (único valor em variável), a impossibilidade de fazer esse procedimento em vetores cria este problema. Que é mitigado somente através de poder computacional.

Ao final do processo, todo o conteúdo das variáveis é convertido para numérico, para que não tenhamos problemas com fatores e strings na manipulação.

O resultado da preparação dos dados é um Data Frame chamado dias_horas, que contém os dados com a primeira linha o dia, a segunda a hora da leitura e um parâmetro por

coluna, já disponível para as outras análises do script principal

3.3 Ajuste da Entrada

O ajuste da entrada (*Feature Engineering*) é o processo de normalização, adequação, agrupamento, combinação e decomposição das variáveis, para que o modelo preditivo/classificador (GUYON, 2003) as compreenda melhor.

Algumas variáveis estão prontas para uso, sem nenhum tipo de tratamento, em sua maioria variáveis com limites fixos ou contidas em intervalos, porém, depende muito da natureza da variável, o único caso específico deste grupo de dados é a direção do vento, que apesar de estar limitada de 0 a 360 (graus), ela pode apresentar problemas (principalmente quando a direção envolve o Norte), pois 359° e 0° estão muito próximos na direção correta, porém estão em lados totalmente opostos numericamente, podendo fazer com que sua média vire para 180°, diametralmente oposto.

Para a variável Direção do Vento, foram criadas outras quatro variáveis, representadas pelos pontos cardeais, e nelas foram calculados os valores de Seno e Cosseno (0..1) de cada ponto, respeitando-se o valor negativo e positivo de cada, portanto, cada entrada é decomposta em uma combinação de quatro outras variáveis. Por exemplo, se o sensor aponta 45°, as quatro variáveis respondem N=0.5, L=0.5, S=0, O=0.

O projeto descreve um modelo novo, com pouco ou nenhum teste, então para termos mais confiabilidade no resultado, foram utilizadas somente linhas com 100% dos dados íntegros (diferente de nulo), portanto, excluiu-se as colunas referente a radiação, pois, de longe era o dado que mais apresentava falhas de medição, este sensor apresentou falha em 49344 medições, enquanto a média de falha era de 2258 em todos os outros sensores.

Neste ponto do trabalho, o dataset está em formato aceitável, organizado e coeso, no entanto, cada linha ainda representa uma hora de medição, representando 104040 linhas de dados. Muitos modelos de *Machine Learning*, principalmente os que utilizam algoritmos de redes neurais para previsão ou classificação de dados podem se valer deste volume de dados, porém modelos preditivos mais simples como este precisam de dados mais compactos e agregados, como o objetivo da previsão era somente um dia, o caminho natural para a condensação é então, o intervalo de um dia.

Foi criado então um dataset novo, contendo os dados agregados, em três funções de agregação diferentes: Soma, Média e Delta (máxima - mínima) de cada uma das variáveis. E este seria o dataset para estudo.

Por se tratar de um modelo preditivo, podemos até dizer em um problema supervisionado, precisamos de um resultado para comparação, por simplificação é mais fácil lidar com o resultado na mesma linha, então é preciso escolher o

resultado (no caso foi a soma da precipitação diária), e deslocá-la através da função `shift.column` do pacote `useful`, isso gerou uma coluna a mais, com a soma da precipitação do dia seguinte ao referido na linha, para uma classificação binária, foi utilizado um condicional `SE...ENTÃO` simples para o resultado ser 0 ou 1. (se `soma_precipitacao > 0` então 1, senão 0).

3.4 Seleção das variáveis de entrada

Com a agregação dos subconjuntos do dataset (soma, média e deltas) ficamos com uma quantidade enorme de variáveis (63 no total, além das colunas descrevendo o dia da coleta).

Uma das principais características de um sistema Fuzzy é que ele pode ser usado para o apoio a decisão, portanto a seleção de variáveis de entrada é feita com conhecimento de domínio, aproximando-se do que nós utilizamos em nosso processo cognitivo para tentar prever a condição de chuva no dia seguinte, as variáveis escolhidas foram:

médias: temperatura do ar, umidade relativa do ar, temperatura do ponto de orvalho, pressão atmosférica, velocidade do vento, precipitação, direção do vento (em quatro variáveis).

somas: precipitação

deltas: temperatura do ar, umidade relativa do ar, temperatura do ponto de orvalho, pressão atmosférica

3.5 A separação do *dataset* com base no resultado

3.5.1 Definindo resultado

O resultado foi aferido medindo a precipitação somada do dia, ou seja, os dias que tiveram uma soma superior a 0 mm/m2 de precipitação, é considerado o resultado 1, e para facilitar a manipulação dos dados, este resultado foi deslocado em uma linha utilizando a função `shift.column()`, do pacote `useful` (LANDER, 2018) assim, existe uma coluna a mais no dataset, esta coluna dá o resultado do dia seguinte.

Para facilitar ainda mais a manipulação dos dados, essa coluna foi convertida em binária (`ifelse()`) e posteriormente em fator (`factor()`).

3.5.2 Separando os resultados em *datasets* distintos

O sistema estudado é um classificador binário, portanto, o melhor método para atingir tal resultado com Fuzzy é avaliar a probabilidade de cada resultado possível (0 - sem chuva, 1 - com chuva, no dia seguinte).

Foi feita então uma separação do dataset (`subsetting`), separando as linhas pelo resultado, através da função `which()` no colchete de seleção `dataset[ln, col]`. Assim foram criados dois novos datasets para avaliação individual.

Cada dataset terá um sistema individualizado, com as próprias funções de pertinência e regras, e que o resultado de cada um seja a probabilidade de cada um acontecer, calculada pelo FIS.

3.5.3 Criação de sistemas Fuzzy (FIS) paralelos

Para que a classificação seja feita, dois sistemas fuzzy (FIS) são criados, e calculados independentemente, utilizando as mesmas variáveis, porém separadas pelo resultado. Conforme diagrama da Figura 3.

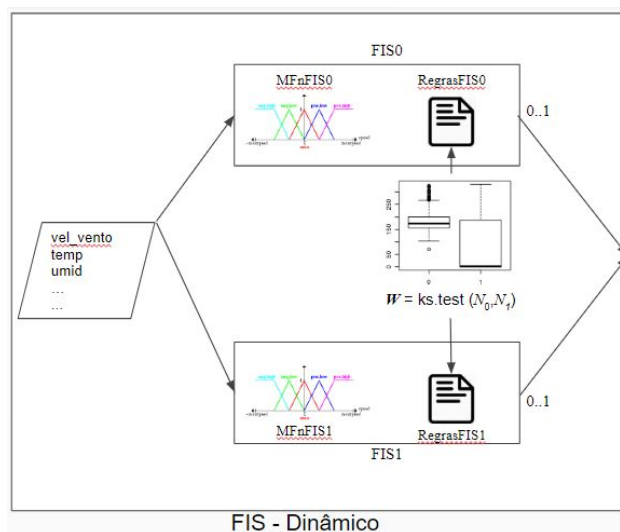


Figura 3 - Descreve o funcionamento dos dois sistemas fuzzy paralelos.

Ao final, cada consulta e execução dos sistemas fuzzy devem retornar dois resultados independentes e simultâneos.

Toda a implementação fuzzy foi feita no pacote FuzzyR (GARIBALDI, J, 2017), que contém todas as funções básicas do cálculo fuzzy baseado em regras e uma interface muito similar ao padrão da indústria, com arquivos de extensão .fis e importação das regras via matriz, tornando a experimentação rápida e prototipagem muito mais rápida e eficiente.

3.6 A formação das funções de pertinência

O sistema deve ser o mais dinâmico possível para que atenda o maior número de datasets e variáveis possível, para tanto, as funções de pertinência devem ser dinâmicas e auto-calculadas.

Dentre os muitos métodos possíveis, os mais populares utilizam redes neurais e algoritmos genéticos para definir as funções de pertinência, embora estejam bem estabelecidas na literatura, consomem muito tempo e recursos computacionais (ASANKA, 2017).

O método utilizado no trabalho, descrito na Figura 4 abaixo, define os limites através dos valores obtidos nos box-plots das colunas (Quartis e mediana), criando 3 estágios de probabilidade, sendo a mediana a maior probabilidade e quanto mais afastado, menor.

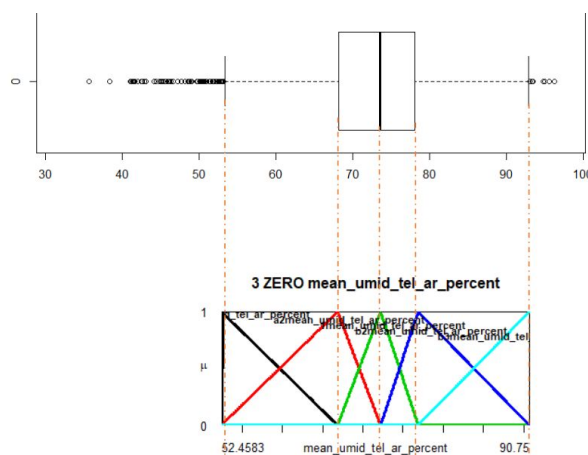


Figura 4 - Boxplots gerando funções de pertinência

3.7 - Criação das regras

Diferentemente de outros métodos de Inteligência Artificial que dependem de aprendizado (otimização), o Sistema Fuzzy não requer treino, porém sofre do tamanho da lista de regras, em um exemplo simples, no sistema proposto, com as variáveis escolhidas, teríamos 155, o que resultaria em absurdas 759375 regras, cada regra ocupa na formação do objeto Fuzzy 2kb de memória RAM, resultando em 1,53 GB somente de regras em um só objeto (o sistema utiliza dois objetos Fuzzy), como descrito na Figura 5. Isso torna a criação e a manipulação do sistema custosa e lenta, isso sem processar nenhum dado, cada vez que o sistema recebe uma entrada ele vai iterar sobre este volume gigantesco de dados e calcular o resultado fuzzy.

Uma nova estratégia foi divisada para superar este obstáculo, que mesmo alterando o resultado numérico do sistema Fuzzy, ainda dá a direção correta do cálculo, foram feitas então as regras incompletas do sistema, não relacionando as variáveis de entrada entre si, cada variável de entrada somente tem relação com o resultado.

Em uma relação com a Estatística, isso torna o sistema Fuzzy Univariado, sendo que o resultado final é um acumulado de todas as variáveis calculadas individualmente.

Para atingir essa tabela, o sistema cria um modelo preenchido com "0" e escreve em cima de cada regra somente, a partir de um modelo, inserido manualmente.

Utilizando este método, descrito na Figura 6 o mesmo sistema fica somente com 75 regras (150kb de regras dentro

do objeto FIS) deixando tudo muito mais ágil, pronto para um cálculo iterativo de todo o dataset.

	var1	var2	var3	res_row	1	1
[1,]	1	1	1	1	1	1
[2,]	2	1	1	1	1	1
[3,]	3	1	1	2	1	1
[4,]	4	1	1	1	1	1
[5,]	5	1	1	1	1	1
[6,]	1	2	1	1	1	1
[7,]	2	2	1	2	1	1
[8,]	3	2	1	2	1	1
[9,]	4	2	1	2	1	1
[10,]	5	2	1	1	1	1
[11,]	1	3	1	2	1	1
[12,]	2	3	1	2	1	1
[13,]	3	3	1	3	1	1
[14,]	4	3	1	2	1	1

Figura 5 - Sistema de regras completo, com todas as variáveis descritas no modelo (somente 3 variáveis de entrada, uma de saída com pesos e condicional fixos)

	[.1]	[.2]	[.3]	[.4]	[.5]	[.6]	[.7]	[.8]	[.9]	[.10]	[.11]	[.12]	[.13]	[.14]	[.15]	[.16]	[.17]	[.18]	
[1,]	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.33264819	1
[2,]	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0.33264819	1
[3,]	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0.33264819	1
[4,]	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0.33264819	1
[5,]	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.33264819	1
[6,]	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.90071001	1
[7,]	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0.90071001	1
[8,]	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0.90071001	1
[9,]	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0.90071001	1
[10,]	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.90071001	1

Figura 6 - Sistema de variáveis incompleto, somente relacionando cada variável ao resultado, com pesos dinâmicos e condicional fixo. 15 variáveis de entrada.

3.8 - A aplicação de pesos

O conceito de regras limitadas empregado neste trabalho facilita a aplicação de pesos a cada variável, portanto, é possível criar pesos diferentes para cada entrada no sistema, fazendo com que o resultado penda para as variáveis com maior peso.

O peso das regras de cada variável é calculado a partir da diferença da distribuição normal entre a mesma variável no dataset com resultado 0 e o dataset com resultado 1, utilizando o Teste Kolmogorov-Smirnov definida no R como `ks.test()`.

Para uma maior otimização dos pesos (maximização dos resultados), todos os pesos foram colocados em uma matriz e então normalizados, ou seja, a maior diferença gera um peso de valor 1, e a menor de valor 0 (o que não é calculado no sistema Fuzzy resultante), conforme exemplificado na Figura 7 abaixo.

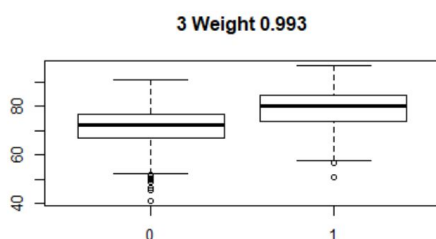


Figura 7 - Boxplots representando a distribuição normal da mesma variável nos datasets 0 e 1, quanto mais diferente, maior o peso (weight)

3.9 - Funções de resultado

Diversas funções de resultado foram criadas, porém, a raiz de todas elas é a função `fuz_sis()` que organiza e concatena o resultado, ela operacionaliza todo o modelo, e todas as outras funções são baseadas nela. Esta função retorna um dataframe simples, com a resposta bruta e numérica dos dois datasets utilizados, nomeados “FIS0” e “FIS1” no resultado. A partir daqui diversos cálculos podem ser efetuados e para facilitar o entendimento e alguns já estão no código citado.

A função `result_matrix()`, executa automaticamente a função `fuz_sis()`, concatena o valor real do resultado, entregando uma matriz que contém o resultado, de cada sistema fuzzy, o resultado real (coluna Benchmark) e duas avaliações binárias, se FIS0 e FIS1 são maiores que 50 (se $FIS > 50$ então 1, senão 0). Já a função `accuracy_fis()`, utiliza as mesmas entradas porém retorna uma matriz de confusão do resultado de `result_matrix()`, em função dos dados de teste (entrada da função), o pacote `Caret` (KUH, 2018) oferece a função de compilação e cálculo da Matriz confusão. A função `avaliacao_periodos()`, estabelece uma manipulação móvel (às vezes chamada “em janela”) dos dados, limitando o dataset de entrada aos últimos N períodos e utilizando a saída de somente uma linha de resultado (dia seguinte) da função `result_matrix()` em um laço `for()`, armazenando todos os resultados em uma matriz e retornando uma matriz confusão de todos os resultados armazenados. Esta é a função utilizada para os resultados descritos neste trabalho.

4 - APLICAÇÃO E RESULTADOS

Devido a natureza cíclica do objeto (análise de chuva é um evento por período), os experimentos e seus resultados também foram idealizados considerando períodos.

Uma vez que este trabalho é baseado em conhecimento de domínio, a escolha da duração dos períodos também foi feita utilizando este raciocínio.

Para se aproximar de uma utilização cotidiana, diária, somente a avaliação “em janela” foi avaliada, no período do último ano de medição (que engloba todas as estações do ano).

Os períodos de análise escolhidos foram, todos os dados até então (variável `nALLt0`), 90 dias (uma estação do ano, variável `n90t0`), 180 dias (duas estações do ano, variável `n180t0`) e 365 dias (um ano completo, quatro estações, variável `n365t0`, acrescido do dia atual).

Estes resultados foram aplicados com o período anterior, ou seja, no caso de 365 dias, ele precisa de 365 dias anteriores para avaliação.

	Acerto Positivo	Acerto Negativo
90 dias	67 (53,6%)	175 (72,6%)
180 dias	63 (59,4%)	190 (73,0%)
365 dias	72 (69,9%)	202 (76,8%)
Todo o período anterior	73 (61,8%)	188 (75,8%)

Tabela 1 - Compilação dos resultados dos experimentos executados (Acertos)

	Falso Positivo	Falso Negativo
90 dias	58 (46,4%)	66 (27,3%)
180 dias	43 (40,5%)	70 (26,9%)
365 dias	31 (30,1%)	61 (23,1%)
Todo o período anterior	45 (38,1%)	60 (24,1%)

Tabela 2 - Compilação dos resultados dos experimentos executados (Erros/Falsos)

	Precisão Geral
90 dias	66,1%
180 dias	69,1%
365 dias	74,8%
Todo o período anterior	71,3%

Tabela 3 - Compilação dos resultados dos experimentos executados (Precisão Geral)

5. CONCLUSÕES

Diante do resultado, fica claro que o sistema proposto tem capacidade de prever a chuva como um evento porém também evidencia sinais de sobre ajuste se observamos que a precisão cai quando se usa o período total disponível..

É necessário, porém, testes e ensaios para que as próprias variáveis sejam auto-selecionadas, com base estatística.

O sistema e seu script R, como está posto também apresenta um consumo excessivo de recursos computacionais, tornando sua utilização e teste demorados para uma aplicação prática, a

execução simples de cada uma das variáveis demora em média 5 minutos em um computador pessoal intermediário.

Embora o modelo seja mais preciso quando não chove (convencionado 0 ou, simplesmente “negativo”), ainda assim apresenta acertos de 70% no resultado positivo, tornando-o qualificado para desenvolvimentos futuros.

REFERÊNCIAS

- Alves, Aj. EFLL (Embedded Fuzzy Logic Library) is a standard library for Embedded Systems. Disponível em: <<https://github.com/zerokol/eFLL>>. Acesso em: 20 set. 2018.
- Asanka, PPG Dinesh; PERERA, Amal Shehan. DEFINING FUZZY MEMBERSHIP FUNCTION USING BOX PLOT. 2017.
- Berrocal, Veronica J., Rafter, Adriane, Tilmann, Gneiting, 2008. Probabilistic quantitative precipitation field forecasting using a two-stage. The Annals of Applied Statistics 2 (4), 1170?1193.
- BRASIL. Lei 12.527, de 18 de novembro de 2011. Regula o acesso a informações previsto no inciso XXXIII do art. 5º, no inciso II do § 3º do art. 37 e no § 2º do art. 216 da Constituição Federal; e dispositivos da Lei nº 8.159, de 8 de janeiro de 1991; e dá outras providências. Disponível em: . Acesso em: 22 set. 2018
- CPTEC/INPE Nota da Coordenação-Geral do CPTEC/INPE sobre a implementação operacional do modelo regional operacional do CPTEC - modelo regional WRF. 2018
- Garibaldi, J, CHAO C. FuzzyR: Fuzzy Logic Toolkit, 2017 for R. R package version 2.1. <https://CRAN.R-project.org/package=FuzzyR>
- Guyon, Isabelle; ELISSEEFF, André. An introduction to variable and feature selection. Journal of machine learning research, v. 3, n. Mar, p. 1157-1182, 2003.
- Kuhn, M. caret: Classification and Regression Training. R package version 5.16-24, 2018.
- Lander, J. et al.useful: A Collection of Handy, Useful Functions. R package version 1.2.6, 2018
- Vanhoucke, Vincent; SENIOR, Andrew; MAO, Mark Z. Improving the speed of neural networks on CPUs. In: Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop. 2011. p. 4.