Pacote 'darch'

20 de julho de 2016

Tipo Pacote

Pacote de títulos para arquiteturas profundas e máquinas restritas de Boltzmann

Versão 0.12.0

Data 2016-07-19

Autor Martin Drees [aut, cre, cph],
Johannes Rueckert [ctb],
Christoph M. Friedrich [ctb],
Geoffrey Hinton [cph],
Ruslan Salakhutdinov [cph],
Carl Edward Rasmussen [cph],

Manutenção Martin Drees <mdrees@stud.fh-dortmund.de>

Descrição O pacote darch é construído com base no código da GE
Hinton e RR Salakhutdinov (disponível no Código Matlab para a crença profunda
Redes). Este pacote é para gerar redes neurais com muitas camadas (profundas
Arquitecturas) e treiná-los com o método introduzido pelas publicações
"Um algoritmo de aprendizado rápido para redes de crenças profundas" (GE Hinton, S. Osindero,
YW Teh (2006) <DOI: 10.1162 / neco.2006.18.7.1527>) e "Reduzir a
Dimensionalidade de dados com redes neurais "(GE Hinton, RR
Salakhutdinov (2006) <DOI: 10.1126 / science.1127647>). Este método inclui
Pré-treinamento com o método de divergência contrastiva publicado por GE Hinton
(2002) <DOI: 10.1162 / 089976602760128018> e um ajuste fino com conhecidos comuns
Algoritmos de treinamento como retropropagação ou gradientes conjugados.
Além disso, o ajuste fino supervisionado pode ser aprimorado
Duas técnicas desenvolvidas recentemente para melhorar o ajuste fino das
Aprendendo.

Licença GPL (> = 2) | Arquivo LICENSE

URL https://github.com/maddin79/darch

BugReports https://github.com/maddin79/darch/issues

Depende de R (> = 3.0.0)

Importa estatísticas, métodos, ggplot2, reshape2, futile.logger (> = 1.4.1),
Caret, Rcpp (& gt; = 0,12.3)

Ligando para Rcpp

Sugere foreach, doRNG, NeuralNetTools, gputools, testthat, plyr (> = 1.8.3.9000)

Agrupar 'RcppExports.R' 'autosave.R' 'net.Class.R' 'darch.Class.R' 'Backpropagation.R' 'benchmark.R' 'bootstrap.R' 'caret.R' 'Compat.R' 'config.R' 'darch.Add.R' 'darch.Getter.R' 'Dataset.R' 'darch.Learn.R' 'darch.R' 'darch.Setter.R' 'DarchUnitFunctions.R' 'dropout.R' 'errorFunctions.R' 'Rbm.Class.R' 'generateRBMs.R' 'generateWeightsFunctions.R' 'LoadDArch.R' 'log.R' 'makeStartEndPoints.R' 'minimize.R' 'MinimizeAutoencoder.R' 'minimizeClassifier.R' 'mnist.R' 'Momentum.R' 'net.Getter.R' 'newDArch.R' 'params.R' 'plot.R' 'Predict.R' 'print.R' 'rbm.Learn.R' 'rbm.Reset.R' 'RbmUnitFunctions.R' 'rbmUpdate.R' 'rpropagation.R' 'RunDArch.R' 'saveDArch.R' 'test.R' 'util.R' 'WeightUpdateFunctions.R'

RoxygenNote 5.0.1

NecessidadesCompilação sim

Repositório CRAN

Data / publicação 2016-07-20 00:33:10

R documentados:

Backpropagation Função de aprendizagem backpropagation

Descrição

Esta função fornece o algoritmo backpropagation para arquiteturas profundas.

Uso

```
Backpropagation (darch, trainData, targetData,
  Bp.learnRate = getParameter (". Bp.learnRate", rep (1, times =
  Comprimento (darch @ camadas))),
  Bp.learnRateScale = getParameter (".bp.learnRateScale"),
  NesterovMomentum = getParameter (".darch.nesterovMomentum"),
  Dropout = getParameter (".darch.dropout", rep (0, times = comprimento (darch @ layers)
  + 1), darch), dropConnect = getParameter (".darch.dropout.dropConnect"),
  MatMult = getParameter (". MatMult"), debugMode = getParameter (". Debug", F)
  ...)
```

Argumentos

Darch	Uma instância da classe DArch .
TrainData	Os dados de treinamento (entradas).
TargetData	Os dados de destino (saídas).
Bp.learnRate	Taxas de aprendizado para backpropagation, comprimento é um ou o mesmo que o num-Matrizes de peso quando se utilizam taxas de aprendizagem diferentes para cada camada.
Bp.learnRateScale	A taxa de aprendizado é multiplicada por esse valor após cada época.
NesterovMomentum	Ver parâmetro darch.nesterovMomentum de darch .
cair fora	Veja o parâmetro darch.dropout de darch .
DropConnect	Consulte darch.dropout.dropConnect parâmetro de darch .
MatMult	Função de multiplicação de matriz, parâmetro interno.
modo de depuração	Se o modo de depuração está ativado, parâmetro interno.
...	Outros parâmetros.

Detalhes

O único backpropagation-specific, user-relevantes parâmetros são bp.learnRate e bp.learnRateScale;
Eles podem ser passados para a função darch ao ativar backpropagation como o ajuste fino func-
Ção. Bp.learnRate define a taxa de aprendizado de backpropagation e pode ser especificado como um único
Escalar ou como um vetor com uma entrada para cada matriz de peso, permitindo taxas de aprendizagem por camada.
Bp.learnRateScale é um escalar único que contém um fator de escala para a (s) taxa (s) de aprendizado

Será aplicado após cada época.

Backpropagation suporta dropout e usa a função de atualização de peso como definido através do arquivo darch.weightUpdateFunction
Parâmetro de darch .

Valor

A arquitetura profunda treinada

Referências

Rumelhart, D., GE Hinton, RJ Williams, Aprendendo representações por backpropagating erros,
Nature 323, S. 533-536, DOI: 10.1038 / 323533a0, 1986.

Veja também

Darch

Outras funções de ajuste fino: minimizeAutoencoder , minimizeClassifier , rpropagation

Exemplos

```
## Não correr:
Dados (iris)
Modelo <- darch (Espécie ~., Iris, darch.fineTuneFunction = "backpropagation")

## End (Não executado)
```

CrossEntropyError Função de erro de entropia cruzada

Descrição

A função calcula o erro de entropia cruzada a partir dos parâmetros originais e de estimativa.

Uso

```
CrossEntropyError (original, estimativa)
```

Argumentos

original	A matriz de dados original.
estimativa	A matriz de dados calculada.

Detalhes

Esta função pode ser usada para o parâmetro darch.errorFunction da função darch , mas é
Somente uma função de erro válida se usada em combinação com a função de ativação softmaxUnit! Isto é
Não é um valor válido para o parâmetro rbm.errorFunction.

Valor

Uma lista com o nome da função de erro na primeira entrada eo valor de erro na segunda entrada.

Referências

Rubinstein, RY, Kroese, DP (2004). O Método da Entropia Cruzada: Uma Abordagem Unificada
Otimização Binacional, Simulação de Monte Carlo e Aprendizagem Automática, Springer-Verlag, New
Iorque.

Veja também

Outras funções de erro: mseError , rmseError

Exemplos

```
## Não correr:
Dados (iris)
Modelo <- darch (Espécie ~., Iris, darch.errorFunction = "crossEntropyError")

## End (Não executado)
```

Darch Ajustar uma rede neural profunda

Descrição

Ajustar uma rede neural profunda com pré-treinamento opcional e um de vários algoritmos de afinação fina.

Uso

```
Darch (x, ...)

## Método S3 padrão:
Darch (x, y, layers = 10, ..., autosave = F,
  Autosave.epochs = round (darch.numEpochs / 20),
  Autosave.dir = "./darch.autosave", autosave.trim = F, bp.learnRate = 1,
  Bp.learnRateScale = 1, bootstrap = F, bootstrap.unique = T,
  Bootstrap.num = 0, cg.length = 2, cg.switchLayers = 1, darch = NULL,
  Darch.batchSize = 1, darch.dither = F, darch.dropout = 0,
  Darch.dropout.dropConnect = F, darch.dropout.momentMatching = 0,
  Darch.dropout.oneMaskPerEpoch = F, darch.elu.alpha = 1,
  Darch.errorFunction = if (darch.isClass) crossEntropyError mais mseError,
```

```
  Darch.finalMomentum = 0.9, darch.fineTuneFunction = backpropagation,
  Darch.initialMomentum = 0,5, darch.isClass = T,
  Darch.maxout.poolSize = 2, darch.maxout.unitFunction = linearUnit,
  Darch.momentumRampLength = 1, darch.nesterovMomentum = T,
  Darch.numEpochs = 100, darch.returnBestModel = T,
  Darch.returnBestModel.validationErrorFactor = 1 - exp (-1),
  Darch.stopClassErr = -Inf, darch.stopErr = -Inf,
  Darch.stopValidClassErr = -Inf, darch.stopValidErr = -Inf,
  Darch.trainLayers = T, darch.unitFunction = sigmoidUnit,
  Darch.weightDecay = 0,
  Darch.weightUpdateFunction = weightDecayWeightUpdate, dataSet = NULL,
  DataSetValid = NULL,
  GenerateWeightsFunction = generateWeightsGlorotUniform, gputools = F,
  Gputools.deviceId = 0, logLevel = NULL, normalizeWeights = F,
  NormalizeWeightsBound = 15, paramsList = list (),
  PreProc.factorToNumeric = F, preProc.factorToNumeric.targets = F,
  PreProc.fullRank = T, preProc.fullRank.targets = F,
  PreProc.orderedToFactor.targets = T, preProc.params = F,
  PreProc.targets = F, rbm.allData = F, rbm.batchSize = 1,
  Rbm.consecutive = T, rbm.errorFunction = mseError,
  Rbm.finalMomentum = 0.9, rbm.initialMomentum = 0.5, rbm.lastLayer = 0,
  Rbm.learnRate = 1, rbm.learnRateScale = 1, rbm.momentumRampLength = 1,
  Rbm.numCD = 1, rbm.numEpochs = 0, rbm.unitFunction = sigmoidUnitRbm,
  Rbm.updateFunction = rbmUpdate, rbm.weightDecay = 2e-04, retainData = F,
  Rprop.decFact = 0,5, rprop.incFact = 1,2, rprop.initDelta = 1/80,
  Rprop.maxDelta = 50, rprop.method = "iRprop +", rprop.minDelta = 1e-06,
  Seed = NULL, shuffleTrainData = T, weights.max = 0,1,
  Weights.mean = 0, weights.min = -0,1, weights.sd = 0,01,
  XValid = NULL, yValid = NULL)

## método S3 para a fórmula de classe
Darch (x, dados, camadas, ..., xValid = NULL, dataSet = NULL,
  DataSetValid = NULL, logLevel = NULL, paramsList = list (),
  Darch = NULL)

## Método S3 para a classe DataSet
Darch (x, ...)
```

Argumentos

X	Matriz de dados de entrada ou data.frame (darch.default) ou fórmula (darch.formula) Ou DataSet (darch.DataSet).
...	Parâmetros adicionais.
Y	Matriz de dados de destino ou data.frame , Se x é uma matriz de dados de entrada ou data.frame .
camadas	Vector contendo um inteiro para o número de neurônios de cada camada. Padrões Para c (a, 10, b), onde a é o número de colunas nos dados de treinamento eb a Número de colunas nos alvos. Se este tem o comprimento 1, é usado como o número Dos neurônios na camada oculta, não como o número de camadas!

salvamento automático	Lógical indicando se deseja ativar automaticamente salvando a instância DArch Para um arquivo durante o ajuste fino.
Autosave.epochs	Depois de quantas épocas deve auto-salvar acontecer, por padrão, após cada 5 1, A rede só será salva uma vez quando você fino-tuning é feito.
Autosave.dir	Diretório para os arquivos de salvamento automático, os nomes dos arquivos serão, por exemplo, autosave_010.net para A instância DArch após 10 epochs
Autosave.trim	Se deseja cortar a rede antes de salvá-la. Isso removerá o conjunto de dados e

A camada de pesos, resultando em uma rede que não é mais utilizável para previsões
Ou treinamento. Útil quando apenas estatísticas e configurações precisam ser armazenadas.

Bp.learnRate Taxas de aprendizado para backpropagation, comprimento é um ou o mesmo que o num-
Matrizes de peso quando se utilizam taxas de aprendizagem diferentes para cada camada.

Bp.learnRateScale

A taxa de aprendizado é multiplicada por esse valor após cada época.

Bootstrap Logical indicando se usar bootstrapping para criar um treinamento e validação
Dados de treinamento dados.

Bootstrap.unique

Logical indicando se devem ser tomadas somente amostras exclusivas para o treinamento (VERDADEIRO,
Padrão) ou tirar todas as amostras tiradas (FALSE), o que resultará em um treinamento maior
Conjunto com duplicatas. Nota: Isto é ignorado se bootstrap.num for maior que 0.

Bootstrap.num Se este for maior que 0, bootstrapping irá desenhar este número de amostras de treino
Sem substituição.

Cg.length Números de pesquisa de linha

Cg.switchLayers

Indica quando treinar a rede completa em vez de apenas as duas camadas superiores

Darch Instância DArch existente para a qual o treinamento deve ser retomado. Nota: Quando en-
Abling pré-treinamento, resultados anteriores de treinamento que se perdeu, ver explicação para pa-
Rameter rbm.numEpochs.

Darch.batchSize

Tamanho do lote, ou seja, o número de amostras de treinamento que são apresentadas à rede
Antes que sejam executadas atualizações de peso, para ajuste fino.

Darch.dither Se aplicar dither a colunas numéricas nos dados de entrada de treinamento.

Darch.dropout Taxas de desistência. Se este for um vetor, ele será tratado como as taxas de abandono para cada
Camada individual. Se um elemento estiver faltando, o dropout de entrada será definido como 0.
Ao ativar o darch.dropout.dropConnect, esse vetor precisa de um
Elemento (uma matriz de elemento por peso entre duas camadas em oposição a uma
Elemento por camada excluindo a última camada).

Darch.dropout.dropConnect

Se usar DropConnect em vez de dropout para as camadas ocultas. Usará
Darch.dropout como as taxas de abandono.

Darch.dropout.momentMatching

Quantas iterações a realizar durante a correspondência momentânea para inferência de abandono,
0 para desativar a correspondência momentânea.

Darch.dropout.oneMaskPerEpoch

Seja para gerar uma nova máscara para cada lote (FALSE, padrão) ou para cada
Época (TRUE).

Darch.elu.alfa

Parâmetro alfa para a função de unidade linear exponencial. Ver exponentialLinearUnit .

Darch.errorFunction

Função de erro durante o ajuste fino. Possíveis funções de erro incluem mseError , RmseError ,
E crossEntropyError .

Darch.finalMomentum

Momento final durante o ajuste fino.

Darch.fineTuneFunction

Função de ajuste fino. Os valores possíveis incluem backpropagation (padrão), rpropagation ,
MinimizeClassifier e minimizeAutoencoder (não supervisionado).

Darch.initialMomentum

Momento inicial durante o ajuste fino.

Darch.isClass Se a saída deve ser tratada como rótulos de classe durante o ajuste fino e classifi-
Devem ser impressas.

Darch.maxout.poolSize

Tamanho da piscina para unidades de maxout, quando utilizar a função de ativação de maxout. Vejo
MaxoutUnit .

Darch.maxout.unitFunction

Função de unidade interna usada por maxout. Consulte darch.unitFunction para a unidade possível
funções.

Darch.momentumRampLength

Depois de quantas épocas, em relação ao número total de épocas treinadas,
O momento alcança darch.finalMomentum? Um valor de 1 indica que o

Darch.finalMomentum deve ser alcançado na época final, um valor de 0,5 em-
Indica que darch.finalMomentum deve ser alcançado após metade do trem-
Está completo. Note que isto irá conduzir a solavancos na rampa de impulso se
A formação é retomada com os mesmos parâmetros para darch.initialMomentum e
Darch.finalMomentum. Defina darch.momentumRampLength como 0 para evitar isso

Problema ao retomar o treinamento.

Darch.nesterovMomentum

Se usar Nesterov Accelerated Momentum . (NAG) para a descida gradiente
Com base em algoritmos de ajuste fino.

Darch.numEpochs

Número de épocas de afinação.

Darch.returnBestModel

Lógico, indicando se o melhor modelo deve ser retornado no final do treinamento,
Do último.

Darch.returnBestModel.validationErrorFactor

Ao avaliar modelos com dados de validação, quão alta deve ser a validação
Erro, em comparação com o erro de formação? Este é um valor entre 0 e
1. Por padrão, esse valor é 1 - exp (-1). O fator de erro de treinamento eo
O fator de erro de validação sempre adicionará a 1, então se você passar 1 aqui, o treinamento
Erro será ignorado, e se você passar 0 aqui, o erro de validação será ignorado.

Darch.stopClassErr

Quando o erro de classificação é inferior ou igual a este valor, o treinamento é
Parado (0..100).

Darch.stopErr Quando o valor da função de erro é inferior ou igual a este valor, o treinamento
está parado.

Darch.stopValidClassErr

Quando o erro de classificação nos dados de validação for inferior ou igual a este
Valor, o treinamento é interrompido (0..100).

Darch.stopValidErr

Quando o valor da função de erro nos dados de validação for inferior ou igual
Para este valor, o treinamento é interrompido.

Darch.trainLayers

TRUE para treinar todas as camadas ou uma máscara contendo TRUE para todas as camadas que
Deve ser treinado e FALSO para todas as camadas que não devem ser treinadas (nenhuma entrada
Para a camada de entrada).

Darch.unitFunction

Função de camada ou vetor de funções de camada de comprimento número de camadas -
1. Note que a primeira entrada significa a função de camada entre as camadas 1 e
2, ou seja, a saída da camada 2. A camada 1 não tem uma função de camada, uma vez que
Os valores de entrada são usados diretamente. Possíveis funções de unidade incluem linearUnit ,
SigmoidUnit , tanhUnit , rectifiedLinearUnit , softplusUnit , softmaxUnit ,

E maxoutUnit .

Darch.weightDecay

Fator de decaimento de peso, o padrão é 0. Todos os pesos serão multiplicados por (1 - darch.weightDecay)
Antes de cada atualização de peso.

Darch.weightUpdateFunction

Função de atualização de peso ou vetor de funções de atualização de peso, muito semelhante a
Darch.unitFunction. Possíveis funções de atualização de peso incluem weightDecayWeightUpdate

E maxoutWeightUpdate Observe que maxoutWeightUpdate deve ser usado no
Camada após a função de ativação maxout!

Conjunto de dados A instância DataSet , passada de darch.DataSet (), pode ser especificada manualmente.

DataSetValid DataSet contendo dados de validação.

GenerateWeightsFunction

Função de geração de peso ou vetor de funções de geração de camada de comprimento
Número de camadas - 1. Possíveis funções de geração de peso incluem generateWeightsUniform

(Padrão), generateWeightsNormal , generateWeightsGlorotNormal , generateWeightsGlorotUniform ,
GenerateWeightsHeNormal , E generateWeightsHeUniform .

Gputools Logical indicando se deve usar gputools para a multiplicação matricial, se disponível
capaz.

Gputools.deviceId

Inteiro especificando o dispositivo a ser usado para a multiplicação de matriz GPU. Veja chooseGpu .

LogLevel Futile.logger log nível. Usa o nível de log definido atualmente por padrão,

É futile.logger :: flog.info se não foi alterado. Outros níveis disponíveis
Incluem, de menor a mais detalhado: FATAL, ERROR, WARN, DEBUG e TRACE.

NormalizeWeights
Logical indicando se a normalização de pesos (norma L2 = normalizeWeightsBound).

Página 10

NormalizeWeightsBound
Limite superior da norma L2 dos vetores de peso entrantes. Usado somente se normalizeWeights é verdade.

ParamsList Lista de parâmetros, pode incluir e substitui parâmetros especificados listados acima. Primário para conveniência ou para uso em scripts.

PreProc.factorToNumeric
Se todos os fatores devem ser convertidos para numéricos.

PreProc.factorToNumeric.targets
Se todos os fatores devem ser convertidos para numéricos nos dados de destino.

PreProc.fullRank
Se usar a codificação de classificação completa. Consulte preProcess para obter detalhes.

PreProc.fullRank.targets
Se usar a codificação de classificação completa para dados de destino. Consulte preProcess para obter detalhes.

PreProc.orderedToFactor.targets
Se fatores ordenados nos dados de destino devem ser convertidos em dados não Fatores. Nota: Os fatores ordenados são convertidos para numeric por dummyVars e não Mais utilizável para tarefas de classificação.

PreProc.params Lista de parâmetros a passar para a função preProcess para os dados de entrada ou FALSE
Para desativar o pré-processamento de dados de entrada.

PreProc.targets
Se os dados de destino devem ser centralizados e dimensionados. Ao contrário do preProc.params, este É apenas um lógico transformar pré-processamento de dados alvo ligado ou desligado, uma vez que este pré-O processamento deve ser revertido ao prever novos dados. Mais útil para Missão. Nota: Isso irá enviesar o erro de rede bruta.

Rbm.allData Logical indicando se deve usar dados de treinamento e validação para pré-treinamento. Nota: Isto também se aplica ao usar bootstrapping.

Rbm.batchSize Tamanho do lote de pré-treino.

Rbm.consecutive
Logical indicando se deve treinar os RBMs um de cada vez para rbm.numEpochs (TRUE, padrão) ou alternadamente treinando cada RBM para uma época em um Tempo (FALSE).

Rbm.errorFunction
Função de erro durante o pré-treino. Isso é usado apenas para estimar o erro RBM E não afeta o próprio treinamento. Possíveis funções de erro incluem mseError E rmseError .

Rbm.finalMomentum
Momento final durante o pré-treino.

Rbm.initialMomentum
Momento inicial durante o pré-treino.

Rbm.lastLayer Numérico indicando em que camada para parar o pré-treinamento. Valores possíveis
Clude 0, significando que todas as camadas são treinadas; Inteiros positivos, significando parar Treinamento após o RBM onde rbm.lastLayer forma a camada visível; negativo Inteiros, o que significa parar o treinamento no rbm.lastLayer RBMs do topo RBM.

Rbm.learnRate Taxa de aprendizagem durante a pré-formação.

Página 11

Rbm.learnRateScale
: As taxas de aprendizado serão multiplicadas com esse valor após cada época.

Rbm.momentumRampLength
: Depois de quantas épocas, em relação a rbm.numEpochs, o momento
Alcance rbm.finalMomentum? Um valor de 1 indica que o rbm.finalMomentum
Deve ser atingido na época final, um valor de 0,5 indica que rbm.finalMomentum
Deve ser atingido após a metade do treinamento está completo.

Rbm.numCD
: Número de etapas completas para as quais a divergência contrastiva é realizada. Aumentando
Isso vai diminuir o treinamento para baixo consideravelmente.

Rbm.numEpochs
: Número de épocas de pré-formação. Nota: Ao passar um valor diferente de 0 aqui
E também passando uma instância DArch existente através do parâmetro darch, os pesos
Da rede será completamente redefinido! A pré-formação é essencialmente uma
Inicialização de peso avançada e não faz sentido executar o pré-treinamento
Uma rede previamente treinada.

Rbm.unitFunction
: Função da unidade durante o pré-treino. Possíveis funções incluem sigmoidUnitRbm
(Padrão), tanhUnitRbm , E linearUnitRbm .

Rbm.updateFunction
: Função de atualização durante o pré-treinamento. Atualmente, darch fornece apenas rbmUpdate .

Rbm.weightDecay
: Decomposição de peso pré-treino. Os pesos serão multiplicados por (1 - rbm.weightDecay)
Antes de cada atualização de peso.

RetainData
: Logical indicando se deve armazenar os dados de treinamento na instância DArch após
Treinamento ou ao salvá-lo em disco.

Rprop.decFact
: Fator decrescente para o treinamento. O padrão é 0.6.

Rprop.incFact
: Fator crescente para o treinamento Padrão é 1.2.

Rprop.initDelta
: Valor de inicialização para a atualização. O padrão é 0,0125.

Rprop.maxDelta
: Limite superior para o tamanho da etapa. O padrão é 50

Rprop.method
: O método para o treinamento. O padrão é "iRprop +"

Rprop.minDelta
: Limite inferior para tamanho de passo. O padrão é 0.000001

semente
: Permite a especificação de uma semente que será definida via set.seed . Usado no
Contexto de darchBench .

ShuffleTrainData
: Logical indicando se deseja shuffle dados de treinamento antes de cada época.

Pesos.max
: Max para a função runif .

Pesos significa
: Parâmetro médio para a função rnorm .

Pesos.min
: Min para a função runif .

Pesos
: Sd para a função rnorm .

XValid
: Matriz de dados de entrada de validação ou data.frame .

Yvalid
: Matriz de dados de destino de validação ou data.frame , Se x é uma matriz de dados ou data.frame .

dados
: Data.frame contendo o dataset, se x for uma fórmula .

Detalhes

O pacote darch implementa Deep Architecture Networks e máquinas Boltzmann restritas.

A criação deste pacote é motivada pelos artigos de G. Hinton et. Al. A partir de 2006 (ver
Referências para detalhes) e do código-fonte MATLAB desenvolvido neste contexto. Este pacote
Oferece a possibilidade de gerar redes de arquitetura profunda (darch) como as redes de crenças profundas
De Hinton et. As arquiteturas profundas podem então ser treinadas com a divergência contrastiva
método. Após este pré-treinamento pode ser ajustado com vários métodos de aprendizagem como backprop-
Agilidade, retropropagação resiliente e gradientes conjugados, bem como técnicas mais recentes como
Abandono e maxout.

Consulte https://github.com/maddin79/darch para obter mais informações, documentação e lançamentos.

Pacote:	Darch
Tipo:	Pacote
Versão:	0.10.0
Encontro:	2015-11-12
Licença:	GPL-2 ou posterior
LazyLoad:	sim

Valor

Instância do DArch ajustada

Autor (es)

Martin Drees <mdrees@stud.fh-dortmund.de> e colaboradores.

Referências

Hinton, GE, S. Osindero, YW Teh, Um algoritmo de aprendizado rápido para redes de crenças profundas, Putation 18 (7), S. 1527-1554, DOI: 10.1162 / neco.2006.18.7.1527 2006.

Hinton, GE, RR Salakhutdinov, Reduzir a dimensionalidade dos dados com redes neurais, Science 313 (5786), S. 504-507, DOI: 10.1126 / science.1127647, 2006.

Hinton, Geoffrey E. et ai. (2012). "Melhorar as redes neurais impedindo a coadaptação do recurso Detectores ". Em: Ortopedia Clínica e Pesquisa Relacionada abs / 1207.0580. URL: http://arxiv.org/abs/1207.0580.

Goodfellow, Ian J. et ai. (2013). "Maxout Networks". Em: Actas da 30ª Conferência Internacional Conferência sobre Aprendizagem Automática, ICML 2013, Atlanta, GA, EUA, 16-21 de Junho de 2013, pp. 1319-1327. URL: http://jmlr.org/proceedings/papers/v28/goodfellow13.html.

Drees, Martin (2013). "Implementierung und Analyse von tiefen Arquitekturen in R". Alemão. Tese de mestrado. Fachhochschule Dortmund.

Rueckert, Johannes (2015). Msgstr "Estendendo a biblioteca do Darch para arquiteturas profundas". Tese de projeto. Fachhochschule Dortmund. URL: http://static.saviola.de/publications/rueckert_2015.pdf.

Veja também

Outras funções de interface darch: darchBench , darchTest , plot.DArch , predict.DArch , print.DArch

Exemplos

```
## Não correr:
Dados (iris)
Modelo <- darch (Espécie ~., Íris)
Impressão (modelo)
Previsões <- predict (model, newdata = iris, type = "class")
Cat (paste ("Classificações incorretas:", sum (predictions! = Iris [, 5])))

TrainData <- matriz (c (0,0,0,1,1,0,1,1), ncol = 2, byrow = TRUE)
TrainTargets <- matrix (c (0,1,1,0), nrow = 4)
Model2 <- darch (trainData, trainTargets, layers = c (2, 10, 1),
  Darch.numEpochs = 500, darch.stopClassErr = 0, retainData = T)
E <- darchTest (modelo2)
Cat (paste0 ("Classificações incorretas em todos os exemplos:", e [3], "(",
  E [2], "%) \ n"))

Parcela (modelo2)

## End (Não executado)

#
# Mais exemplos podem ser encontrados em
# Https://github.com/maddin79/darch/tree/v0.12.0/examples
```

Darchbank Benchmarking wrapper para darch

Descrição

Função de benchmarking simples que envolve a função darch para usuários que não podem ou não Deseja usar o pacote de cenários para benchmarking. Esta função requer que o pacote foreach Trabalho, e irá realizar benchmarks paralelos se um backend apropriado foi registrado previamente.

Uso

DarchBench (..., bench.times = 1, bench.save = F,
Bench.dir = "./darch.benchmark", bench.continue = T, bench.delete = F,
Bench.seeds = NULL, output.capture = bench.save, logLevel = NULL)

Argumentos

...	Parâmetros para a função darch
Hora de banco	Quantas execuções de benchmark para executar
Banco de areia	Se deseja salvar resultados de benchmarking em um diretório
Banco de dados	Caminho (relativo ou absoluto) incluindo o diretório onde os resultados de benchmark são salvos Se bench.save for verdadeiro

Bench.continue	Se o benchmark deve ser continuado de uma execução anterior. Se VERDADEIRO, existente Os resultados de benchmark são procurados no diretório dado em bench.dir e novos Os resultados são acrescentados. Se tanto este como o bench.continue forem FALSE ea dire- Tório dado em bench.dir já existe, o treinamento será abortado com um erro.
Bancada	Se deve excluir o conteúdo de bench.dir se bench.continue for FALSE. Cau- : Isto irá tentar apagar TODOS os arquivos no diretório dado, use no seu próprio risco!
Sementes de bancada	Vetor de sementes, um para cada corrida. Será passado para darch .
Output.capture	Se para capturar saída R em arquivos .Rout no diretório especificado. Isto é o Única forma de obter acesso à saída R, uma vez que o loop foreach não Nada para o console. Será ignorado se bench.save for FALSE.
LogLevel	Futile.logger log nível. Usa o nível de log definido atualmente por padrão, É futile.logger :: flog.info se não foi alterado. Outros níveis disponíveis Incluem, de menor a mais detalhado: FATAL, ERROR, WARN, DEBUG e TRACE.

Valor

Lista de instâncias DArch; Os resultados de cada chamada para darch.

Veja também

Outras funções de interface darch: darchTest , darch , plot.DArch , predict.DArch , print.DArch

Exemplos

```
## Não correr:
Dados (iris)
ModelList <- darchBench (Espécies ~., Iris, c (0, 50, 0),
 PreProc.params = list (method = c ("center", "scale")),
 Darch.unitFunction = c ("sigmoidUnit", "softmaxUnit"),
 Darch.numEpochs = 30, bench.times = 10, bench.save = T)

## End (Não executado)
```

DarchModelInfo

Cria um modelo de cursor personalizado para darch.

Descrição

Esta função cria uma descrição de modelo de circunferência para permitir o treinamento de instâncias de DArch com o trem
função. Consulte a documentação sobre modelos de caret personalizados para obter mais informações e
Como criar params válidos e valores de grade.

Uso

DarchModelInfo (params = NULL, grid = NULL)

Argumentos

Params	Data.frame de parâmetros ou NULL para usar um padrão simples (bp.learnRate).
grade	Função que processa um data.frame contendo uma grade de parâmetro combi-Nações ou NULL para usar um padrão simples.

Valor

Um modelo caret válido que pode ser passado para treinar .

Veja também

Caret modelos personalizados

Exemplos

```
## Não correr:
Dados (iris)
Tc <- trainControl (method = "boot", number = 5, allowParallel = F,
  VerboseIter = T)

Parâmetros <- data.frame (parâmetro = c ("layers", "bp.learnRate", "darch.unitFunction"),
  Classe = c ("caractere", "numérico", "caractere"),
  Label = c ("Estrutura da rede", "Taxa de aprendizagem", "unitFunction"))

Grid <- function (x, y, len = NULL, search = "grid")
{
  Df <- expand.grid (layers = c ("c (0,20,0)", "c (0,10,10,0)", "c (0,10,5,5,0)") ,
    Bp.learnRate = c (1,2,5,10))

  Df [["darch.unitFunction"]] <- rep (c ("c (tanhUnit, softmaxUnit)",
    "C (tanhUnit, tanhunit, softmaxUnit)",
    "C (tanhUnit, tanhUnit, tanhUnit, softmaxUnit)"), 4)

  Df
}

CaretModel <- train (Espécie ~., Data = iris, tuneLength = 12, trControl = tc,
  Method = darchModelInfo (parameters, grid), preProc = c ("center", "scale"),
  Darch.numEpochs = 15, darch.batchSize = 6, testing = T, ...)

## End (Não executado)
```

DarchTest Rede de classificação de teste.

Descrição

Propaga-propagar dados dados através da rede neural profunda e retornar precisão de classificação Usando os rótulos dados.

Uso

```
DarchTest (darch, newdata = NULL, targets = T)
```

Argumentos

Darch	Exemplo DArch .
Novos dados	Novos dados para usar, NULL para usar dados de treinamento.

Alvos	Etiquetas para os dados, NULL para usar etiquetas de treinamento (só é possível quando os dados são NULL também).

Detalhes

Esta é principalmente uma função de conveniência semelhante ao predict.Adrch com desempenho de classificação
Em vez da saída da rede, e retorna uma lista de indicadores de precisão (rede
Erro, percentagem de classificações incorrectas e número absoluto de classificações incorrectas).

Valor

Vetor contendo saída de função de erro, porcentagem de classificações incorretas e número absoluto
De classificações incorretas.

Veja também

Outras funções de interface darch: darchBench , darch , plot.DArch , predict.DArch , print.DArch

Exemplos

```
## Não correr:
Dados (iris)
Modelo <- darch (Species ~., Iris, retainData = T)
ClassificationStats <- darchTest (modelo)

## End (Não executado)
```

ExponentialLinearUnit Função de unidade linear exponencial (ELU) com derivadas unitárias.

Descrição

A função calcula a ativação das unidades e retorna uma lista, na qual a primeira entrada é
A ativação linear exponencial das unidades ea segunda entrada é a derivada da transferência
função.

Uso

```
ExponentialLinearUnit (input, alpha = getParameter (".darch.elu.alpha", 1, ...),
  ...)
```


Argumentos

entrada	Entrada para a função de ativação.
alfa	Hiperparâmetro ELU.
...	Parâmetros adicionais.

Valor

Uma lista com a ativação ELU na primeira entrada ea derivada da ativação na segunda entrada
entrada.

Referências

Clevert, Djork-Arne, Thomas Unterthiner e Sepp Hochreiter (2015). "Profundidade rápida e precisa
Aprendizagem em Rede por Unidades Lineares Exponenciais (ELUs) ". Em: CoRR abs / 1511.07289. URL:
Http://arxiv.org/abs/1511.07289

Veja também

Outras funções da unidade de darch: linearUnit , maxoutUnit , rectifiedLinearUnit , sigmoidUnit , softmaxUnit , SoftplusUnit , tanhUnit

Exemplos

```
## Não correr:
Dados (iris)
```

```
Modelo <- darch (Species ~., Iris, darch.unitFunction = "exponentialLinearUnit",
  Darch.elu.alpha = 2)

## End (Não executado)
```

GenerateWeightsGlorotNormal

Inicialização do peso normal Glorot

Descrição

Esta função é utilizada para gerar pesos aleatórios e vieses usando a inicialização de peso normal de Glorot. Como descrito em Glorot & Bengio, AISTATS 2010.

Uso

```
GenerateWeightsGlorotNormal (numUnits1, numUnits2,
  Weights.mean = getParameter (". Weights.mean", 0, ...), ...)
```


Argumentos

NumUnits1	Número de unidades na camada inferior.
NumUnits2	Número de unidades na camada superior.
Pesos significa	Parâmetro médio para a função rnorm .
...	Parâmetros adicionais, usados para resolução de parâmetros e passados para generateWeightsNormal .

Valor

Matriz de pesos.

Referências

Glorot, Xavier e Yoshua Bengio (2010). "Entender a dificuldade de formação profunda feed-
Redes de neurônios avançados "In: Conferência internacional sobre inteligência artificial e estatística, pp. 249-256

Veja também

Outras funções de geração de peso: generateWeightsGlorotUniform , generateWeightsHeNormal , GenerateWeightsHeUniform , generateWeightsNormal , generateWeightsUniform

Exemplos

```
## Não correr:
Dados (iris)
Model <- darch (Species ~., Iris, generateWeightsFunction = "generateWeightsGlorotNormal",
  Weights.mean = .1)

## End (Não executado)
```

GenerateWeightsGlorotUniform

Inicialização do peso uniforme Glorot

Descrição

Esta função é utilizada para gerar pesos aleatórios e vieses usando Glorot uniforme peso inicial-
Como descrito em Glorot & Bengio, AISTATS 2010.

Uso

```
GenerateWeightsGlorotUniform (numUnits1, numUnits2, ...)
```

Argumentos

NumUnits1

numUnits2	Número de unidades na camada inferior. Número de unidades na camada superior.
...	Os parâmetros adicionais, utilizados para resolução de parâmetros e passados para generateWeightsUniform .

Valor

matriz de peso.

Referências

Glorot, Xavier e Yoshua Bengio (2010). "Compreender a dificuldade de formação feedback profunda redes neurais para a frente". In: Conferência internacional sobre inteligência e estatísticas artificial, pp. 249-256

Veja também

Outras funções de geração de peso: generateWeightsGlorotNormal , generateWeightsHeNormal , generateWeightsHeUniform , generateWeightsNormal , generateWeightsUniform

Exemplos

```
## Não correr:
de dados (íris)
modelo <- Darch (. Espécies ~, íris, generateWeightsFunction = "generateWeightsGlorotUniform")

## End (não executados)
```

generateWeightsHeNormal

Ele inicialização peso normal

Descrição

Esta função é usada para gerar pesos aleatórios e preconceitos usando Ele inicialização peso normal como descrito em He et al., http://arxiv.org/abs/1502.01852 .

Uso

```
generateWeightsHeNormal (numUnits1, numUnits2,
  weights.mean = getParameter ( "weights.mean", 0, ...), ...)
```

Argumentos

numUnits1	Número de unidades na camada inferior.
numUnits2	Número de unidades na camada superior.
weights.mean	significa parâmetro para o rnorm função.
...	Os parâmetros adicionais, utilizados para resolução de parâmetros e passados para generateWeightsNormal .

Valor

matriz de peso.

Referências

Ele, Kaiming, Xiangyu Zhang, Shaoqing Ren, e Jian Sun (2015). "Mergulhar profundamente em Rectifiers: Superando o desempenho no nível humano no IMAGEnet Classificação". In: Corr abs / 1.502,01852. URL: http://arxiv.org/abs/1502.01852

Veja também

Outras funções de geração de peso: generateWeightsGlorotNormal , generateWeightsGlorotUniform , generateWeightsHeUniform , generateWeightsNormal , generateWeightsUniform

Exemplos

```
## Não correr:
de dados (íris)
modelo <-. Darch (Espécies ~, íris, generateWeightsFunction = "generateWeightsHeNormal",
  weights.mean = 0,1)

## End (não executados)
```

generateWeightsHeUniform

Ele inicialização peso uniforme

Descrição

Esta função é usada para gerar pesos aleatórios e preconceitos usando Ele inicialização peso uniforme como descrito em He et al., http://arxiv.org/abs/1502.01852 .

Uso

```
generateWeightsHeUniform (numUnits1, numUnits2, ...)
```

Argumentos

numUnits1	Número de unidades na camada inferior.
numUnits2	Número de unidades na camada superior.
...	Os parâmetros adicionais, utilizados para resolução de parâmetros e passados para generateWeightsUniform .

Valor

matriz de peso.

Referências

Ele, Kaiming, Xiangyu Zhang, Shaoqing Ren, e Jian Sun (2015). "Mergulhar profundamente em Rectifiers: Superando o desempenho no nível humano no IMAGEnet Classificação". In: Corr abs / 1.502,01852. URL: http://arxiv.org/abs/1502.01852

Veja também

Outras funções de geração de peso: generateWeightsGlorotNormal , generateWeightsGlorotUniform , generateWeightsHeNormal , generateWeightsNormal , generateWeightsUniform

Exemplos

```
## Não correr:
de dados (íris)
modelo <- Darch (. Espécies ~, íris, generateWeightsFunction = "generateWeightsHeUniform")

## End (não executados)
```

generateWeightsNormal gera uma matriz de peso usando rnorm .

Descrição

Esta função é o método padrão para a geração de pesos para os casos de Líquido .Ele usa rnorm para faça isso.

Uso

```
generateWeightsNormal (numUnits1, numUnits2,
   weights.mean = getParameter ( "weights.mean", 0, ...),
   weights.sd = getParameter ( "weights.sd", 0,01, ...), ...)
```

Argumentos

numUnits1	Número de unidades na camada inferior.
numUnits2	Número de unidades na camada superior.
weights.mean	significa parâmetro para o rnorm função.
weights.sd	parâmetro sd ao rnorm função.
...	Os parâmetros adicionais, usados para a resolução de parâmetro.

Valor

matriz de peso.

Veja também

Outras funções de geração de peso: generateWeightsGlorotNormal , generateWeightsGlorotUniform , generateWeightsHeNormal , generateWeightsHeUniform , generateWeightsUniform

Exemplos

```
## Não correr:
de dados (íris)
modelo <-. Darch (Espécies ~, íris, generateWeightsFunction = "generateWeightsNormal",
  weights.mean = 0,1, weights.sd = 0,05)

## End (não executados)
```

generateWeightsUniform

Gera uma matriz de peso usando runif

Descrição

Esta função é usada para gerar pesos aleatórios e preconceitos usando runif .

Uso

```
generateWeightsUniform (numUnits1, numUnits2,
   weights.min = getParameter ( "weights.min", -0.1, ...),
   weights.max = getParameter ( "weights.max", 0,1, ...), ...)
```

Argumentos

numUnits1	Número de unidades na camada inferior.
numUnits2	Número de unidades na camada superior.
weights.min	min parâmetro para o runif função.
weights.max	max parâmetro para o runif função.
...	Os parâmetros adicionais, usados para a resolução de parâmetro.

Valor

matriz de peso.

Veja também

Outras funções de geração de peso: generateWeightsGlorotNormal , generateWeightsGlorotUniform , generateWeightsHeNormal , generateWeightsHeUniform , generateWeightsNormal

Exemplos

```
## Não correr:
de dados (íris)
modelo <- Darch (Espécies ~, íris, generateWeightsFunction = "generateWeightsUniform",.
  weights.min = -.1, weights.max = 0,5)

## End (não executados)
```


linearUnit função de unidade linear com derivados de unidade.

Descrição

A função calcula a activação das unidades e retorna uma lista, em que a primeira entrada é a activação linear das unidades e a segunda entrada é a derivada da função de transferência.

Uso

```
linearUnit (entrada, ...)
```

Argumentos

entrada	Entrada para a função de activação.
...	Os parâmetros adicionais, não usado.

Valor

Uma lista com a activação linear em que a primeira entrada e a derivada da activação no segundo entrada.

Veja também

Outras funções da unidade Darch: exponentialLinearUnit , maxoutUnit , rectifiedLinearUnit , sigmoidUnit , softmaxUnit , softplusUnit , tanhUnit

Exemplos

```
## Não correr:
de dados (íris)
modelo <- Darch (Espécies ~, íris, darch.unitFunction = "linearUnit".)

## End (não executados)
```

maxoutUnit função unidade MAXOUT / LWTA

Descrição

A função calcula a activação das unidades e retorna uma lista, em que a primeira entrada é a resultam através da função de transferência MAXOUT e a segunda entrada é a derivada da transferência função.

Uso

```
maxoutUnit (entrada, ..., poolSize = getParameter ( "darch.maxout.poolSize", 2,
  ...), unitFunc = getParameter ( "darch.maxout.unitFunction", linearUnit,
  ...), dropoutMask = vector ())
```

Argumentos

entrada	Entrada para a função de activação.
...	Os parâmetros adicionais, passada para função da unidade de interior.
poolSize	O tamanho de cada piscina MAXOUT.
unitFunc	função de unidade interna para MAXOUT.
dropoutMask	Vector que contém a máscara de abandono.

Detalhes

MAXOUT define as ativações de todos os neurônios, mas o que tem a maior ativação dentro de uma piscina para 0. Se este for utilizado sem maxoutWeightUpdate , Torna-se o vencedor do local, leva tudo algoritmo, como a única diferença entre os dois é que os pesos de saída são compartilhados por MAXOUT.

Valor

Uma lista com a activação MAXOUT na primeira entrada e a derivada da função de transferência na segunda entrada.

Referências

Srivastava, Rupesh Kumar, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, e Juergen Schmidhuber (2013). "Competir para Calcular". In: Avanços na Neural Information Processing Sistemas 26. Ed. por CJC Burges, L. Bottou, M. Welling, Z. Ghahramani, e KQ Weinberger. Curran Associates, Inc., pp. 2310-2318. URL: http://papers.nips.cc/paper/5059-compete--To compute.pdf

Goodfellow, Ian J., Davi Warde-Farley, Mehdi Mirza, Aaron C. Courville, e Yoshua Bengio (2013). "MAXOUT Networks". In: Anais da 30ª Conferência Internacional sobre a máquina Aprender, ICML 2013, Atlanta, GA, EUA, 16-21 junho de 2013, pp. 1319-1327. URL: http://jmlr.org/proceedings/papers/v28/goodfellow13.h

Veja também

Outras funções da unidade Darch: exponentialLinearUnit , linearUnit , rectifiedLinearUnit , sigmoidUnit , softmaxUnit , softplusUnit , tanhUnit

Exemplos

```
## Não correr:
de dados (íris)
# LWTA:
modelo <-. Darch (Espécies ~, íris, c (0, 50, 0),
 darch.unitFunction = c ( "maxoutUnit", "softmaxUnit"),
 darch.maxout.poolSize = 5, darch.maxout.unitFunction = "sigmoidUnit")
```



```
# MAXOUT:
modelo <-. Darch (Espécies ~, íris, c (0, 50, 0),
 darch.unitFunction = c ( "maxoutUnit", "softmaxUnit"),
 darch.maxout.poolSize = 5, darch.maxout.unitFunction = "sigmoidUnit",
 darch.weightUpdateFunction = c ( "weightDecayWeightUpdate", "maxoutWeightUpdate"))

## End (não executados)
```

maxoutWeightUpdate Actualiza o peso em camadas MAXOUT

Descrição

Em camadas MAXOUT, apenas os pesos de unidades activas são alterados, adicionalmente, todos os pesos dentro de uma piscina deve ser o mesmo.

Uso

```
maxoutWeightUpdate (Darch, layerIndex, weightsInc, biasesInc, ...,
  weightDecay = getParameter ( "darch.weightDecay", 0, Darch),
  poolSize = getParameter ( "darch.maxout.poolSize", 2, Darch))
```

Argumentos

Darch	Darch exemplo.
layerIndex	índice de camada dentro da rede.
weightsInc	Matriz contendo agendado atualizações peso do algoritmo de ajuste fino.
biasesInc	atualizações de peso viés.
...	Os parâmetros adicionais, não usado.
weightDecay	Os pesos são multiplicados por (1 - weightDecay) antes de cada atualização. corresponde ao parâmetro darch.weightDecay de darch.default .
poolSize	Tamanho de piscinas MAXOUT, veja o parâmetro darch.maxout.poolSize de Darch .

Valor

Os pesos atualizados.

Referências

Goodfellow, Ian J., Davi Warde-Farley, Mehdi Mirza, Aaron C. Courville, e Yoshua Bengio (2013). "MAXOUT Networks". In: Anais da 30ª Conferência Internacional sobre a máquina Aprender, ICML 2013, Atlanta, GA, EUA, 16-21 junho de 2013, pp. 1319-1327. URL: http://jmlr.org/proceedings/papers/v28/goodfellow13.h

Veja também

Outras funções de atualização de peso: weightDecayWeightUpdate

Exemplos

```
## Não correr:
de dados (íris)
modelo <-. Darch (Espécies ~, íris, c (0, 50, 0),
 darch.unitFunction = c ( "maxoutUnit", "softmaxUnit"),
 darch.maxout.poolSize = 5, darch.maxout.unitFunction = "sigmoidUnit",
 darch.weightUpdateFunction = c ( "weightDecayWeightUpdate", "maxoutWeightUpdate"))

## End (não executados)
```

minimizeAutoencoder gradiente conjugado para uma rede autoencoder

Descrição

Esta função treina um Darch rede autoencoder com o método de gradiente conjugado.

Uso

```
minimizeAutoencoder (Darch, trainData, targetData,
  cg.length = getParameter ( "cg.length"),
  abandono = getParameter ( "darch.dropout"),
  dropConnect = getParameter ( "darch.dropout.dropConnect"),
  matMult = getParameter ( "matMult "), DebugMode = getParameter (". debug"),
  ...)
```

Argumentos

Darch	Um exemplo da classe Darch .
trainData	A matriz de dados de treinamento.
targetData	Os rótulos para os dados de treinamento.
cg.length	Números de pesquisa linha
cair fora	Veja darch.dropout parâmetro de Darch .
dropConnect	Veja darch.dropout.dropConnect parâmetro de Darch .
matMult	função de multiplicação de matrizes, parâmetro interno.
modo de depuração	Se modo de depuração está habilitado, o parâmetro interno.
...	Outros parâmetros.

Detalhes

Esta função é construído sobre a base do código de G. Hinton et. Al. (http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html - última visita 2016/04/30) para o ajuste fino de redes de crenças profundas. O código original está localizado no arquivos 'backpropclassify.m', 'CG_MNIST.m' e 'CG_CLASSIFY_INIT.m'. Ele implementa a multa ajuste para uma rede classificação com backpropagation usando uma tradução direta do minimizar função de C. Rassmussen (disponível em http://www.gatsby.ucl.ac.uk/~edward/code/minimize/ - última visite 2016/04/30) para R.

minimizeAutoencoder suporta abandono, mas não usa a função de atualização de peso como definido via o parâmetro darch.weightUpdateFunction de Darch , De modo que a decadência de peso, força etc, são não suportado.

Valor

O treinado Darch objeto.

Veja também

Darch , fineTuneDArch

Outras funções de ajuste fino: backpropagation , minimizeClassifier , rpropagation

Exemplos

```
## Não correr:
de dados (íris)
modelo <-. Darch (Espécie ~, íris, c (6,10,2,10,6), darch.isClass = F,
 preProc.params = lista (método c = ( "centro", "escala")),
 darch.numEpochs = 20, darch.batchSize = 6, darch.unitFunction = tanhUnit
 darch.fineTuneFunction = "minimizeAutoencoder")

## End (não executados)
```

minimizeClassifier gradiente conjugado para uma rede de classificação

Descrição

Esta função treina um Darch rede classificador com o método de gradiente conjugado.

Uso

```
minimizeClassifier (Darch, trainData, targetData,
  cg.length = getParameter ( "cg.length"),
  cg.switchLayers = getParameter ( "cg.length"),
  abandono = getParameter ( "darch.dropout"),
  dropConnect = getParameter ( "darch.dropout.dropConnect"),
  matMult = getParameter ( "matMult "), DebugMode = getParameter (". debug"),
  ...)
```

Argumentos

Darch	Um exemplo da classe Darch .

trainData	A matriz de dados de treinamento.
targetData	Os rótulos para os dados de treinamento.
cg.length	Números de pesquisa linha

cg.switchLayers	Indica quando para treinar a rede completo em vez de apenas as duas camadas superiores
cair fora	Veja darch.dropout parâmetro de Darch .
dropConnect	Veja darch.dropout.dropConnect parâmetro de Darch .
matMult	função de multiplicação de matrizes, parâmetro interno.
modo de depuração	Se modo de depuração está habilitado, o parâmetro interno.
...	Outros parâmetros.

Detalhes

Esta função é construir sobre a base do código de G. Hinton et. Al. (http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html - última visita 2016/04/30) para o ajuste fino de redes de crenças profundas. O código original está localizado no arquivos 'backpropclassify.m', 'CG_MNIST.m' e 'CG_CLASSIFY_INIT.m'. Ele implementa a multa ajuste para uma rede classificação com backpropagation usando uma tradução direta do minimizar função de C. Rassmussen (disponível em http://www.gatsby.ucl.ac.uk/~edward/code/minimize/ - última visite 2016/04/30) para R. Os cg.switchLayers parâmetro é para a alternar entre dois formação tipo. Como no código original, as duas camadas superiores pode ser treinado sozinho até que época é igual a epochSwitch. Depois de toda a rede serão treinados.

minimizeClassifier suporta abandono, mas não usa a função de atualização de peso como definido via o parâmetro darch.weightUpdateFunction de Darch , De modo que a decadência de peso, força etc, são não suportado.

Valor

O treinado Darch objeto.

Veja também

Darch , fineTuneDArch

Outras funções de ajuste fino: backpropagation , minimizeAutoencoder , rpropagation

Exemplos

```
## Não correr:
de dados (íris)
modelo <-. Darch (Espécies ~, íris,
 preProc.params = lista (método c = ( "centro", "escala")),
 darch.unitFunction = c ( "sigmoidUnit", "softmaxUnit"),
 darch.fineTuneFunction = "minimizeClassifier",
 cg.length = 3, cg.switchLayers = 5)

## End (não executados)
```

mseError Significa função erro quadrático

Descrição

A função calcula a média de erro ao quadrado (MSE) do parâme- original e estimativa
Ters.

Uso

```
mseError (original, estimativa)
```

Argumentos

original	A matriz de dados original.
estimativa	A matriz de dados calculados.

Detalhes

Esta função é um valor válido para ambos Darch parâmetros rbm.errorFunction e darch.errorFunction.

Valor

A lista com o nome da função de erro na primeira entrada e o valor de erro na segunda entrada.

Veja também

Outras funções de erro: crossEntropyError , rmseError

Exemplos

```
## Não correr:
de dados (íris)
modelo <-. Darch (Espécies ~, íris, rbm.errorFunction = "mseError",
 darch.errorFunction = "mseError")

## End (não executados)
```


plot.DArch Traçar Darch estatísticas ou estrutura.

Descrição

Esta função proporciona parcelas diferentes, dependendo do tipo de parâmetro:

Uso

```
## método S3 para Darch classe
plot (x, y = "raw", ..., type = y)
```

Argumentos

X	Darch exemplo.

Y	Veja tipo.
...	Os parâmetros adicionais, passado para funções de plotagem.
tipo	Que tipo de plano para criar, um dos cru, classe, tempo, momento e net.

Detalhes

- cru. Traça o erro de rede matéria (por exemplo, MSE), este é o padrão
- classe. Lotes do erro de classificação
- Tempo. Lotes os tempos necessários para cada época
- impulso. Lotes a rampa impulso
- net. Chamadas plotnet para traçar a rede

Valor

O traçado.

Veja também

Outras funções de interface Darch: darchBench , darchTest , Darch , predict.DArch , print.DArch

Exemplos

```
## Não correr:
de dados (íris)
modelo <- Darch (. Espécies ~, íris)
trama (modelo)
plot (modelo, "classe")
plot (modelo, "tempo")
plot (modelo, "momentum")
plot (modelo, "net")

## End (não executados)
```


predict.DArch Adiante-se propagam dados.

Descrição

Forward-propagam dado dados através da rede neural profunda.

Uso

```
## método S3 para Darch classe
prever (object, ..., newdata = NULL, type = "raw",
  inputLayer = 1, outputLayer = 0)
```

Argumentos

objeto	Darch instância
...	Outros parâmetros, se newdata é NULL, o primeiro parâmetro sem nome será usado para newdata vez.
newdata	Novos dados para prever, NULL para retornar mais recente saída da rede
tipo	tipo de saída, um dos seguintes: cru, bin, classe ou personagem. cru retorna a camada saída, bin retorna uma para cada camada de saída> 0,5, 0, caso contrário, e retorna classe 1 para a unidade de saída com a maior activação, de outro modo 0. Além disso, quando usando a classe, rótulos de classe são retornados quando disponível. personagem é o mesmo que classe, excepto o uso de vetores de caracteres em vez de fatores.
inputLayer	número Layer (> 0). Os dados apresentados na newdata será alimentado para esta camada. Nota que os números da contagem absoluta de da camada de entrada, ou seja, para uma rede com três camadas, uma indicaria a camada de entrada.
outputLayer	camada número (se> 0) ou deslocamento (se <= 0) em relação à última camada. A saída da camada dado é devolvido. Note que os números absolutos contar a partir da entrada camada, ou seja, para uma rede com três camadas, uma indicaria a camada de entrada.

Valor

Vetor ou matriz de redes saídas, tipo de saída, dependendo do parâmetro tipo.

Veja também

Outras funções de interface Darch: darchBench , darchTest , Darch , plot.DArch , print.DArch

Exemplos

```
## Não correr:
de dados (íris)
modelo <- Darch (. Espécies ~, íris, retainData = T)
predizer (modelo)

## End (não executados)
```


print.DArch Imprimir Darch detalhes.

Descrição

Imprimir informações detalhadas sobre um Darch instância.

Uso

```
## método S3 para Darch classe
impressão (x, ...)
```

Argumentos

X	Darch instância
...	Outros parâmetros, não utilizado.

Detalhes

Informações impressas incluem Darch parâmetros e um resumo das estatísticas de treinamento.

Veja também

Outras funções de interface Darch: darchBench , darchTest , Darch , plot.DArch , predict.DArch

Exemplos

```
## Não correr:
de dados (íris)
modelo <- Darch (. Espécies ~, íris)
print (modelo)

## End (não executados)
```

provideMNIST Fornece definidos na pasta dado dados MNIST.

Descrição

Esta função irá, se necessário e permitido, baixar o conjunto de dados MNIST comprimido e salvá-lo a arquivos .rdata usando readMNIST . Se os arquivos compactados MNIST estão disponíveis, que irá converter los em arquivos RDATA carregável de dentro R. Se os arquivos RDATA já estão disponíveis, nada ser feito.

Uso

```
provideMNIST (pasta = "dados /", baixar = F)
```

Argumentos

pasta	Nome da pasta, incluindo um barra final.
Download	Lógico que indica se o download é permitido.

Valor

valor booleano que indica sucesso ou fracasso.

Exemplos

```
## Não correr:
provideMNIST ( "mnist /", baixar = T)

## End (não executados)
```

rectifiedLinearUnit Rectificado função unidade linear com derivados de unidade.

Descrição

A função calcula a activação das unidades e retorna uma lista, em que a primeira entrada é a activação linear rectificado das unidades e a segunda entrada é a derivada da função de transferência.

Uso

```
rectifiedLinearUnit (entrada, ...)
```

Argumentos

entrada	Entrada para a função de activação.
...	Os parâmetros adicionais, não usado.

Valor

Uma lista com a activação linear rectificado na primeira entrada e a derivada da activação no segunda entrada.

Referências

Glorot, Xavier, Antoine Bordes, e Yoshua Bengio (2011). "Deep Sparse Retificador Neural NET-funciona". In: Proceedings da XIV Conferência Internacional sobre Inteligência Artificial e Estatísticas (AISTATS-11). Ed. por Geoffrey J. Gordon e David B. Dunson. Vol. 15. Jornal de Pesquisa Aprendizado de Máquina - Workshop e actas de conferências, pp 315-323.. URL: http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf

Veja também

Outras funções da unidade Darch: exponentialLinearUnit , linearUnit , maxoutUnit , sigmoidUnit , softmaxUnit , softplusUnit , tanhUnit

Exemplos

Não correr:
de dados (íris)
modelo <- Darch (Espécies ~, íris, darch.unitFunction = "rectifiedLinearUnit".)

End (não executados)

rmseError função de erro root mean square

Descrição

A função calcula o erro de raiz quadrada média (RMSE) do pa- original e estimativa
parâme-.

Uso

rmseError (original, estimativa)

Argumentos

original	A matriz de dados original.
estimativa	A matriz de dados calculados.

Detalhes

Esta função é um valor válido para ambos Darch parâmetros rbm.errorFunction e darch.errorFunction.

Valor

A lista com o nome da função de erro na primeira entrada e o valor de erro na segunda entrada.

Veja também

Outras funções de erro: crossEntropyError , mseError

Exemplos

```
## Não correr:
de dados (íris)
modelo <-. Darch (Espécies ~, íris, rbm.errorFunction = "rmseError",
 darch.errorFunction = "rmseError")

## End (não executados)
```


rpropagation treinamento backpropagation resiliente para arquiteturas de profundidade.

Descrição

A função treina uma arquitetura profunda com o algoritmo backpropagation resistente. É capaz de usar quatro diferentes tipos de formação (ver detalhes). Para detalhes do algoritmo backpropagation resilientes veja as referências.

Uso

```
rpropagation (Darch, trainData, targetData,
  rprop.method = getParameter ( "rprop.method"),
  rprop.decFact = getParameter ( "rprop.decFact"),
  rprop.incFact = getParameter ( "rprop.incFact"),
  rprop.initDelta = getParameter ( "rprop.initDelta"),
  rprop.minDelta = getParameter ( "rprop.minDelta"),
  rprop.maxDelta = getParameter ( "rprop.maxDelta"),
  nesterovMomentum = getParameter ( "darch.nesterovMomentum"),
  abandono = getParameter ( "darch.dropout"),
  dropConnect = getParameter ( "darch.dropout.dropConnect"),
  função erro = getParameter ( "darch.errorFunction"),
  matMult = getParameter ( "matMult "), getParameter DebugMode = (". depuração", F,
  Darch), ...)
```

Argumentos

Darch	A arquitetura profunda para treinar
trainData	Os dados de treinamento
targetData	A saída esperada para os dados de treinamento
rprop.method	O método para a formação. O padrão é "iRprop +"
rprop.decFact	Diminuir fator para a formação. O padrão é de 0,6.
rprop.incFact	Aumentar fator para o padrão de treinamento é de 1,2.
rprop.initDelta	valor de inicialização para a atualização. O padrão é 0,0125.
rprop.minDelta	um limite inferior para o tamanho do passo. O padrão é 0.000001
rprop.maxDelta	Limite superior para o tamanho do passo. O padrão é 50
nesterovMomentum	Veja darch.nesterovMomentum parâmetro de Darch .
cair fora	Veja darch.dropout parâmetro de Darch .
dropConnect	Veja darch.dropout.dropConnect parâmetro de Darch .
função erro	Veja darch.errorFunction parâmetro de Darch .
matMult	função de multiplicação de matrizes, parâmetro interno.
modo de depuração	Se modo de depuração está habilitado, o parâmetro interno.
...	Outros parâmetros.

Detalhes

Rprop suporta abandono e usa a função de atualização de peso como definido através da darch.weightUpdateFunction parâmetro de Darch .

O código para o cálculo da mudança de peso é uma tradução do código MATLAB do Rprop Optimization Toolbox implementado por R. Calandra (ver Referências).

Os métodos de treinamento possíveis (parâmetro rprop.method) são os seguintes (ver Referências para detalhes):

Rprop +: Rprop com Peso-Backtracking
Rprop-: Rprop sem peso-Backtracking
iRprop +: Melhor Rprop com Peso-Backtracking
iRprop-: Rprop melhorada sem peso-Backtracking

Valor

Darch - A arquitetura profunda treinados

Referências

M. Riedmiller, H. Braun. Um método adaptativo direto para a aprendizagem backpropagation mais rápido: O algoritmo Rprop. Em Proceedings da IEEE Conferência Internacional sobre Redes Neurais, pp 586-591. IEEE Press, 1993.

C. Igel, M. Huesken. Melhorar o algoritmo de aprendizagem Rprop, Proceedings of the Second Inter- Simpósio cional em Computação Neural, NC 2000, o ICSC Academic Press, Canada / Suíça, pp. 115-121., 2000.

Kohavi, R., A Study of Cross-Validação e Bootstrap para a Estimativa da Precisão e Modelo Se- lection, Proceedings of the 14th Int. Joint Conference on Artificial Intelligence 2, S. 1137-1143,

Morgan Kaufmann, Morgan Kaufmann Publishers Inc., San Francisco, CA, EUA, 1995.

Veja também

Darch

Outras funções de ajuste fino: backpropagation , minimizeAutoencoder , minimizeClassifier

Exemplos

```
## Não correr:
de dados (íris)
modelo <-. Darch (Espécies ~, íris, darch.fineTuneFunction = "rpropagation",
 preProc.params = lista (método c = ( "centro", "escala")),
 darch.unitFunction = c ( "softplusUnit", "softmaxUnit"),
 rprop.method = "iRprop +", rprop.decFact = 0,5, rprop.incFact = 1,2,
 rprop.initDelta = 1/100, rprop.minDelta = 1/1000000, rprop.maxDelta = 50)

## End (não executados)
```

sigmoidUnit função unidade sigmóide com derivados de unidade.

Descrição

A função calcula a activação e retorna uma lista que a primeira entrada é o resultado através do função de transferência sigmóide e a segunda entrada é a derivada da função de transferência.

Uso

```
sigmoidUnit (entrada, ...)
```

Argumentos

entrada	Entrada para a função de activação.
...	Os parâmetros adicionais, não usado.

Valor

Uma lista com a activação na primeira entrada e a derivada da função de transferência na segunda entrada.

Veja também

Outras funções da unidade Darch: exponentialLinearUnit , linearUnit , maxoutUnit , rectifiedLinearUnit , softmaxUnit , softplusUnit , tanhUnit

Exemplos

```
## Não correr:
de dados (íris)
modelo <- Darch (Espécies ~, íris, darch.unitFunction = "sigmoidUnit".)

## End (não executados)
```

softmaxUnit função unidade Softmax com derivados de unidade.

Descrição

A função calcula a activação das unidades e retorna uma lista, em que a primeira entrada é a resultam através da função de transferência softmax e a segunda entrada é a derivada da transferência função.

Uso

```
softmaxUnit (entrada, ...)
```

Argumentos

entrada	Entrada para a função de activação.
...	Os parâmetros adicionais, não usado.

Valor

Uma lista com a activação softmax na primeira entrada e a derivada da função de transferência na segunda entrada.

Referências

http://www.faqs.org/faqs/ai-faq/neural-nets/part2/section-12.html

Veja também

Outras funções da unidade Darch: exponentialLinearUnit , linearUnit , maxoutUnit , rectifiedLinearUnit , sigmoidUnit , softplusUnit , tanhUnit

Exemplos

```
## Não correr:
de dados (íris)
modelo <-. Darch (Espécies ~, íris,
 darch.unitFunction = c ( "sigmoidUnit", "softmaxUnit"))

## End (não executados)
```


softplusUnit função unidade Softplus com derivados de unidade.

Descrição

A função calcula a activação das unidades e retorna uma lista, em que a primeira entrada seja a activação softmax das unidades e a segunda entrada é a derivada da função de transferência. Softplus é uma versão suavizada da função de activação linear rectificado.

Uso

```
softplusUnit (entrada, ...)
```

Argumentos

entrada	Entrada para a função de activação.
...	Os parâmetros adicionais, não usado.

Valor

Uma lista com a activação Softplus na primeira entrada e a derivada da activação no segundo entrada.

Referências

Dugas, Charles, Yoshua Bengio, Francois Belisle, Claude Nadeau, e Rene Garcia (2001). "inserção incorreta porating segunda ordem Conhecimento funcional para uma melhor precificação das opções". In: Avanços na Neural Sistemas de Processamento de Informações, pp. 472-478.

Veja também

Outras funções da unidade Darch: exponentialLinearUnit , linearUnit , maxoutUnit , rectifiedLinearUnit , sigmoidUnit , softmaxUnit , tanhUnit

Exemplos

```
## Não correr:
de dados (íris)
modelo <- Darch (Espécies ~, íris, darch.unitFunction = "softplusUnit".)

## End (não executados)
```


tanhUnit	função contínua unidade de Tan-sigmóide.

Descrição

Calcula as ativações de unidade e retorna-los em uma lista.

Uso

```
tanhUnit (entrada, ...)
```

Argumentos

entrada	Entrada para a função de activação.
...	Os parâmetros adicionais, não usado.

Valor

Uma lista com a activação na primeira entrada e a derivada da função de transferência na segunda entrada.

Veja também

Outras funções da unidade Darch: exponentialLinearUnit , linearUnit , maxoutUnit , rectifiedLinearUnit , sigmoidUnit , softmaxUnit , softplusUnit

Exemplos

```
## Não correr:
de dados (íris)
modelo <- Darch (Espécies ~, íris, darch.unitFunction = "tanhUnit".)

## End (não executados)
```

weightDecayWeightUpdate

Atualiza o peso usando decadência peso.

Descrição

Multiplica os pesos de (1 - weightDecay) antes da aplicação das mudanças de peso programadas.

Uso

```
weightDecayWeightUpdate (Darch, layerIndex, weightsInc, biasesInc, ...,
  weightDecay = getParameter ( "darch.weightDecay", 0, Darch),
  debug = getParameter ( "debug", F, Darch))
```


Argumentos

Darch	Darch exemplo.
layerIndex	índice de camada dentro da rede.
weightsInc	Matriz contendo agendado atualizações peso do algoritmo de ajuste fino.
biasesInc	atualizações de peso viés.
...	Os parâmetros adicionais, não usado.
weightDecay	Os pesos são multiplicados por (1 - weightDecay) antes de cada atualização. corresponde ao parâmetro darch.weightDecay de darch.default .
depurar	bandeira depuração interna.

Valor

pesos atualizados

Veja também

Outras funções de atualização de peso: maxoutWeightUpdate

Exemplos

```
## Não correr:
modelo <-. Darch (Espécies ~, íris, c (0, 50, 0),
 darch.weightUpdateFunction = "weightDecayWeightUpdate")

## End (não executados)
```

Índice