Parâmetros em Algoritmos de Computação Natural

Computação Natural

Gisele L. Pappa

Como saber se meu algoritmo funciona?

- Algoritmos de computação natural normalmente são utilizados para resolver problemas NP ou NP-hard
- Não espere que os primeiros experimentos que você rodar deem certo
- O sucesso depende fortemente dos parâmetros
- Nem sempre o algoritmo vai funcionar
  - O problema pode ser muito difícil

Mudanças pequenas podem causar grandes impactos

- Existem muitos parâmetros que influenciam o sucesso dos algoritmos
- Mudando um ou dois deles pode não ter nenhum impacto no seu algoritmo, porém pode causar um grande impacto
  - Mudar um terminal pode mudar tudo
- O mesmo é válido para pequenas mudanças no código
  - Trocar um > por um >= pode trocar totalmente os vencedores de um torneio em algoritmos evolucionários

Mudanças grandes podem não causar nenhum impacto

- Se você mexe em todos os parâmetros e nada muda no algoritmo, o que isso significa?
  - Sua representação não é apropriada
  - A fitness não é apropriada
- Se a fitness é apropriada
  - Aumentar o tamanho da população, em algum momento, vai fazer com que a fitness melhore

O que fazer com convergência prematura?

- Não utilizar o operador de reprodução
- Utilizar um mecanismo de seleção mais fraco
  - Por exemplo, diminuir o número de indivíduos de um torneio
- Adicionar diferentes tipos de operadores de mutação
- Introduzir conceitos de espécies
- Utilizar fitness sharing

Como rodar experimentos ?

- Os experimentos não podem ser executados apenas uma vez
  - Em otimização, o comum são 30 vezes
  - Em aprendizagem, 5 ou 10 vezes
- Porém, eles devem ser reprodutíveis
  - Para isso, você deve conhecer a semente aleatória utilizada em cada execução

Como rodar experimentos ?

- Como começar?
  - Escolhendo um parâmetro para variar e deixando todos os outros fixos
  - Existem alguns valores padrão para cada parâmetro, que devem ser utilizados na primeira rodada
- Normalmente se começa variando o tamanho da população, e depois os operadores genéticos
- No caso de GP o tamanho máximo da árvore do indivíduos normalmente é variado antes dos operadores

Como rodar experimentos ?

- Parâmetros “padrão” variam para problemas de otimização versus aprendizagem
  - População :
    - entre 30 e 1000 indivíduos para aprendizagem
    - Entre 500 e 1000 indivíduos para otimização
  - Geração: 50 a 1000? Depende do custo da fitness
  - Operadores
    - Crossover de 90% em GP e variando de 60-90% em GA
    - Mutação variando de 1 a 10% em GA e 10% em GP

Como rodar experimentos ?

- Depois de “otimizar” um parâmetro, este se torna fixo e variamos os outros
- Normalmente, o tamanho da população, número de gerações e a taxa de mutação tem maior influência no algoritmo
- Em GP, o tamanho da árvore (que depende do número de terminais e operadores) também tem influência grande

Análise do Algoritmo

- Monitorar a fitness
- Monitorar a diversidade da população
- Monitorar o número de operações de cruzamento e mutação que geram indivíduos melhores que os pais

Análise da População

- Elemento mais importante a ser analisado
- Avaliação básica: plotar fitness versus geração
- Verificar também
  - Distribuição do tamanho dos indivíduos, se ele for variável
  - Distribuição dos valores de fitness durante cada geração
    - Pode dar dicas a respeito da estrutura do espaço de busca
    - Se existe uma variação muito grande entre os valores, podem ter regiões do espaço que não estão sendo descobertas

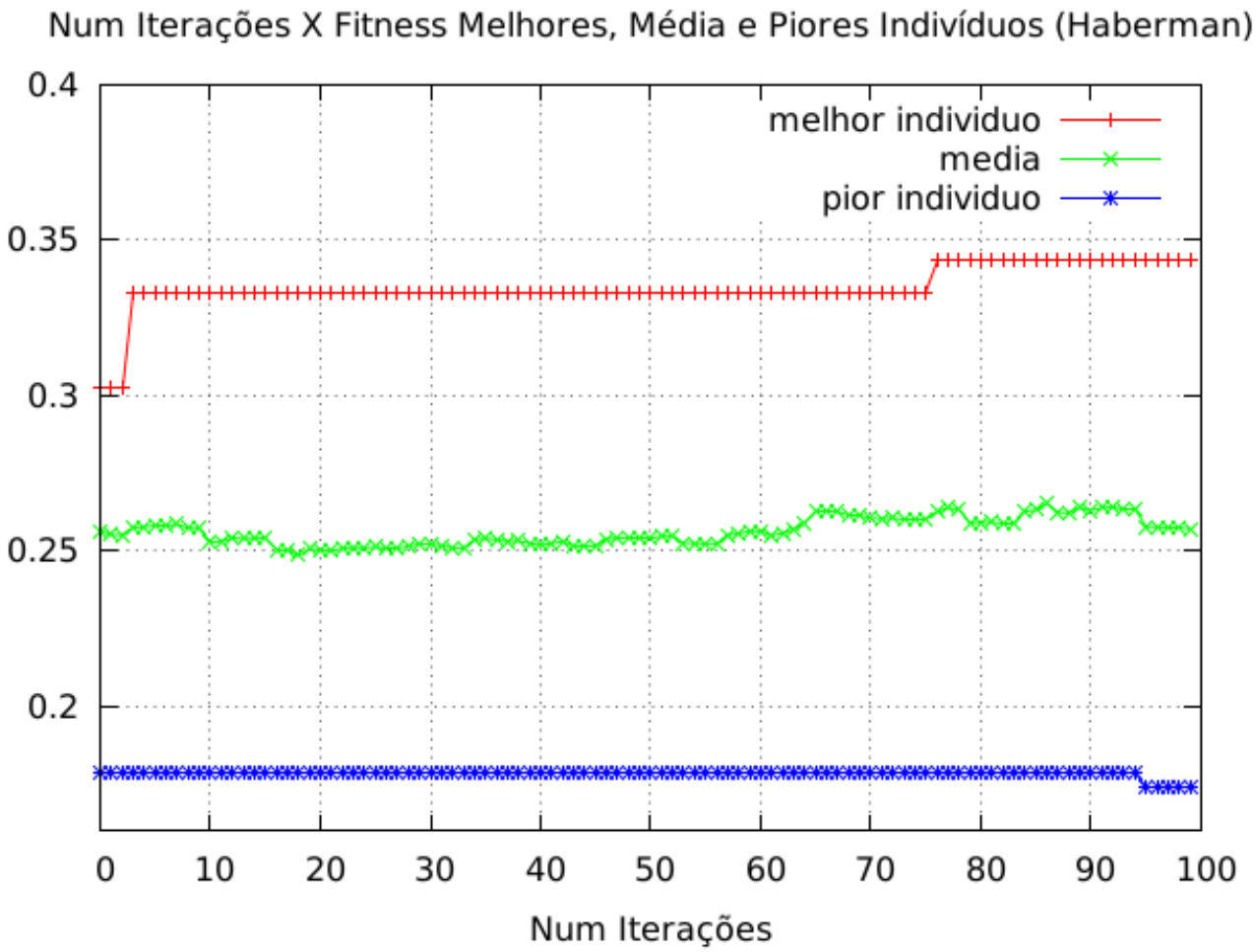
Análise da População

- Para isso, você deve guardar em um log o máximo de informações possíveis.
- Vizualizar os dados onde a fitness está sendo calculada também pode ser de grande ajuda

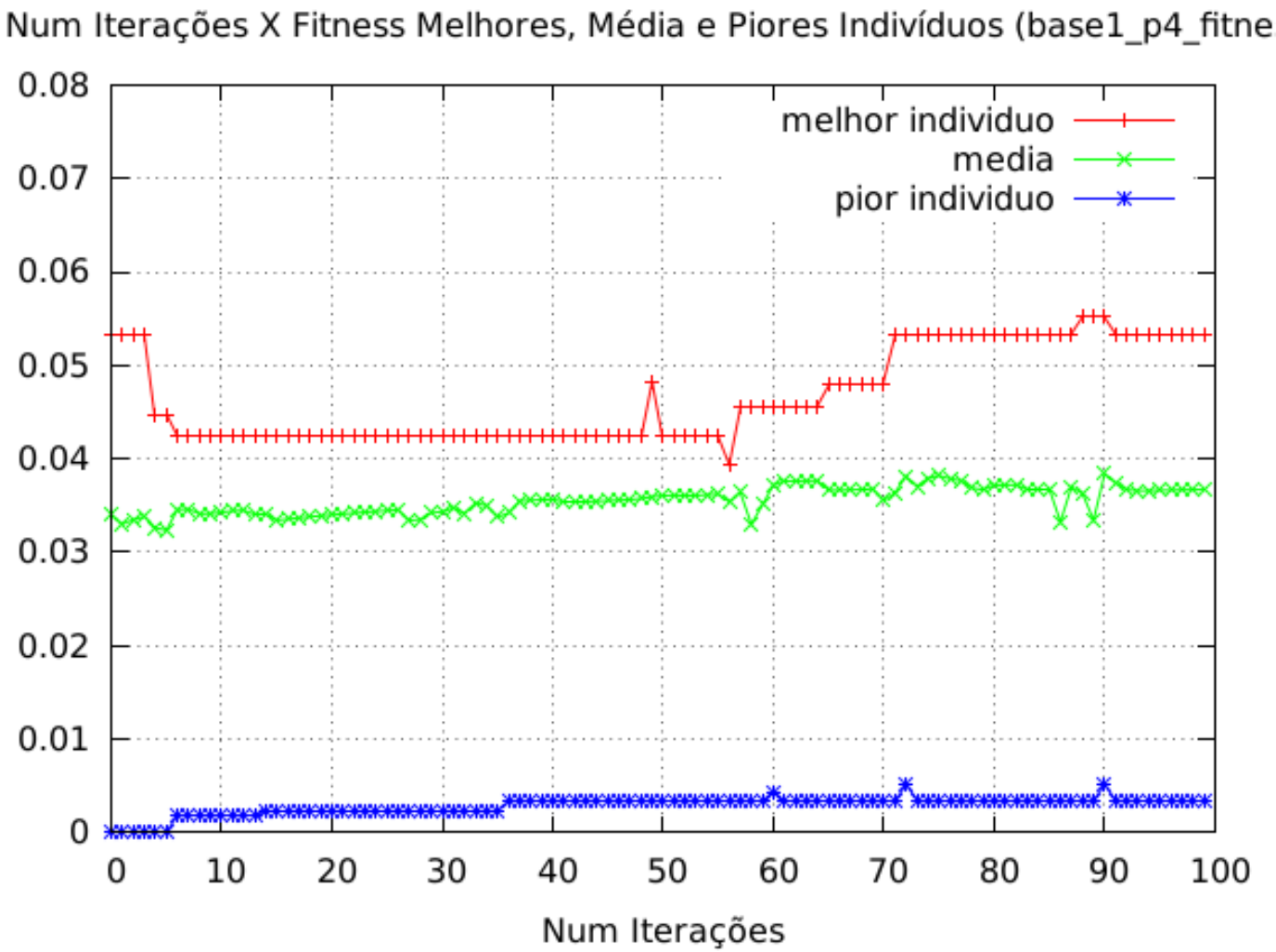
Diversidade

- Se mais de 90% dos indivíduos da população são iguais em um tempo muito curto, isso pode indicar um problema
- Ao mesmo tempo, muita diversidade também pode representar um problema
- Por isso, se possível, medir a distância de fenótipo entre dois indivíduos pode ser muito importante

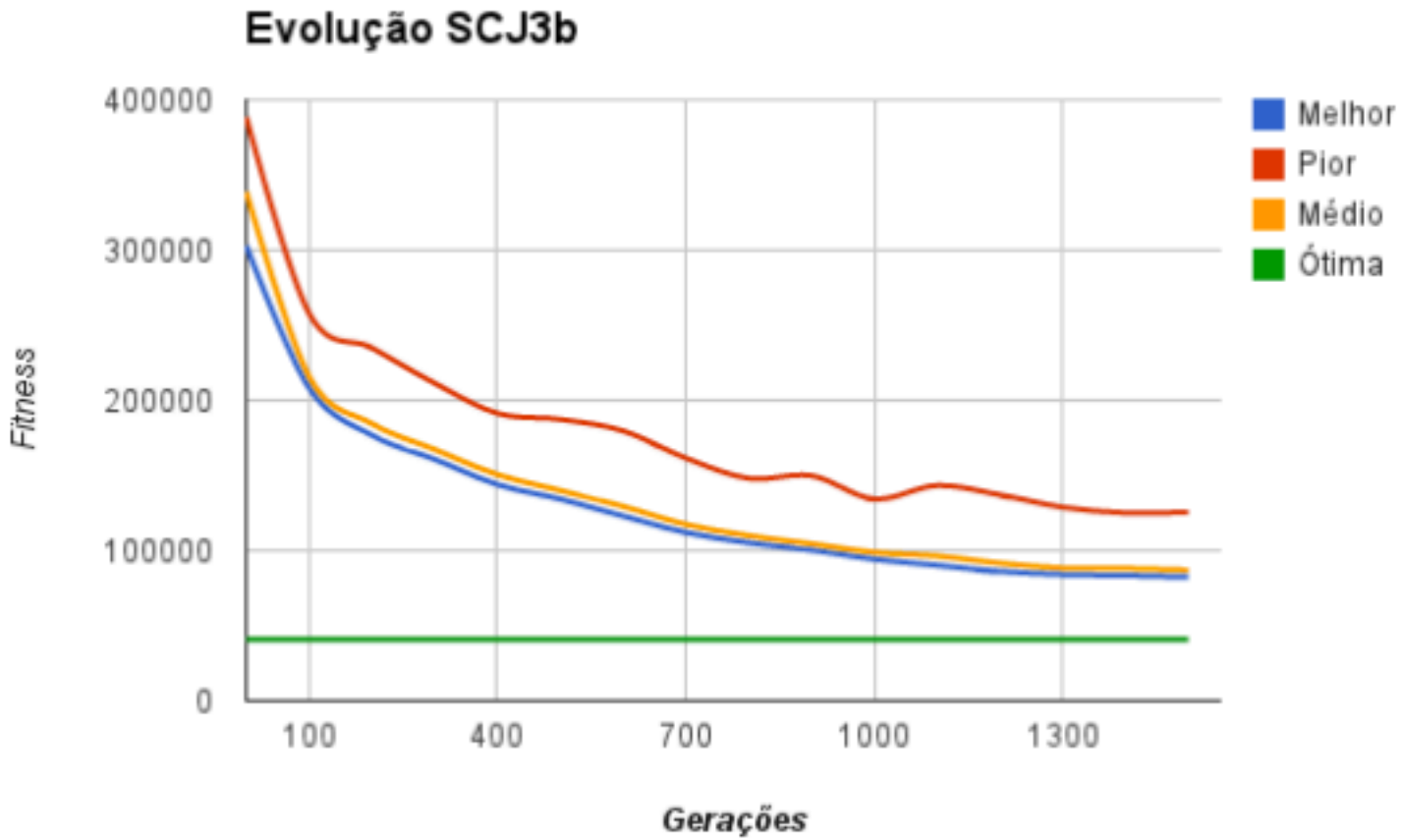
Exemplos


Num Iterações X Fitness Melhores, Média e Piores Indivíduos (Haberman)
melhor individuo
media
pior individuo
0.4
0.35
0.3
0.25
0.2
0
10
20
30
40
50
60
70
80
90
100
Num Iterações

Exemplos


Num Iterações X Fitness Melhores, Média e Piores Indivíduos (base1_p4_fitne
melhor individuo
media
pior individuo
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
10
20
30
40
50
60
70
80
90
100
Num Iterações

Exemplos


Evolução SCJ3b
400000
300000
200000
100000
0
Fitness
100
400
700
1000
1300
Gerações
Melhor
Pior
Médio
Ótima

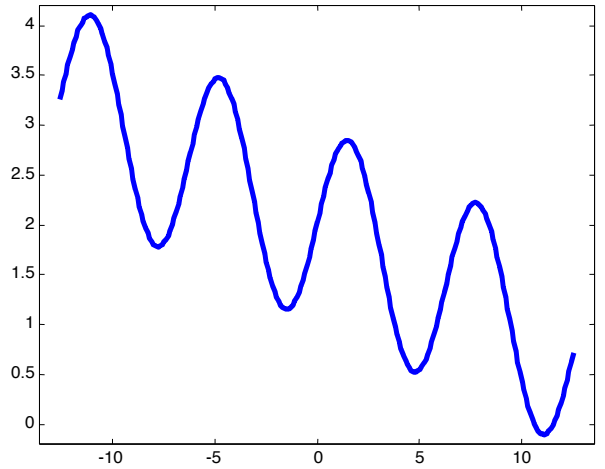
Otimização versus Aprendizagem

Trabalhando com dados

- A maioria dos métodos de computação natural trabalha com dados
- A metodologia de experimentação para otimização é muito diferente da de aprendizagem
- Em otimização o objetivo é, através de um conjunto de dados, encontrar a solução ótima para o problema em mãos
  - A fitness é calculada utilizando a base de dados completa

Exemplo

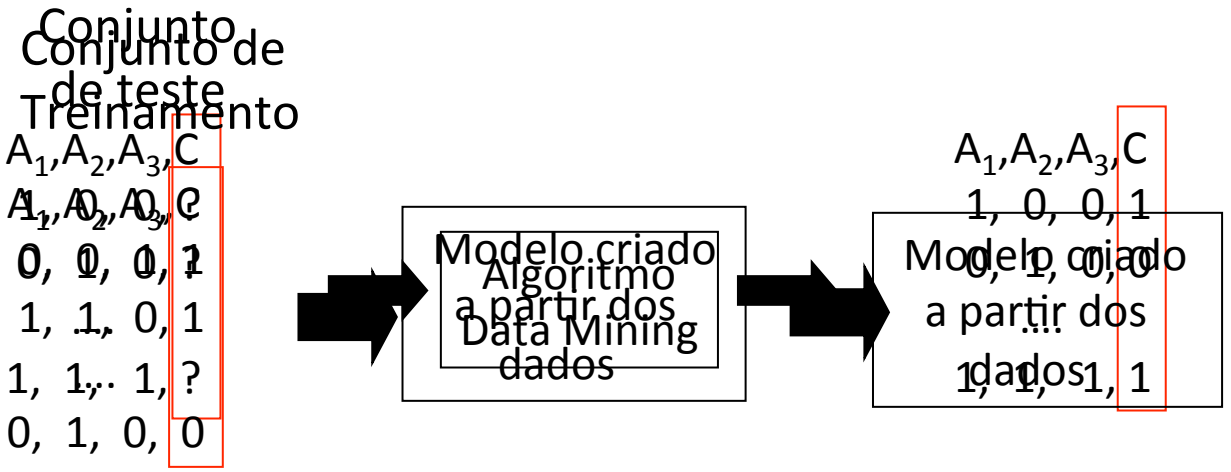
x	saída
-10	3.6
-8	2.1
-6	2.7
-4	2.6
.	.
.	.



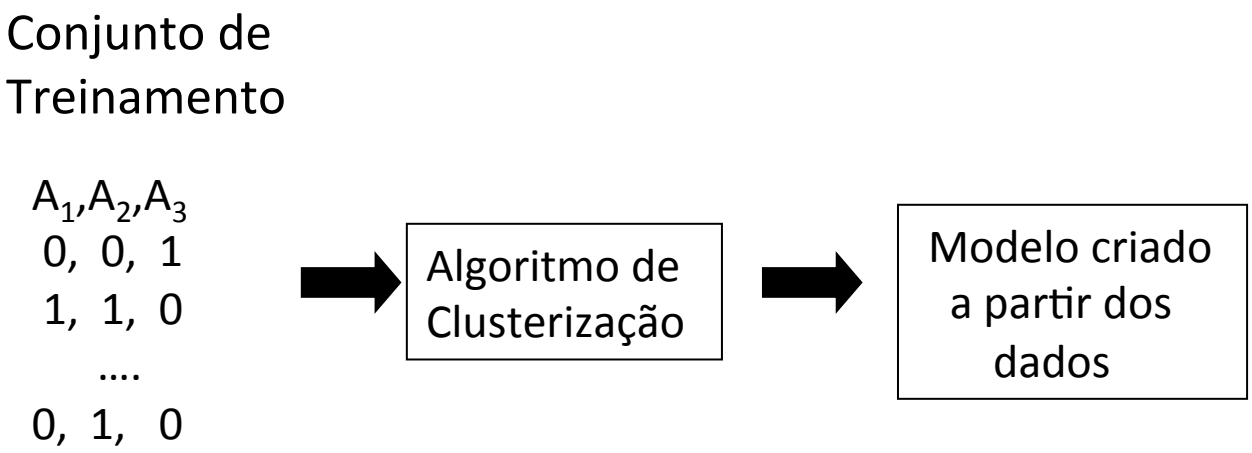
Trabalhando com Dados

- Aprendizagem
  - O objetivo é criar um modelo que depois vai ser utilizado num novo conjunto de dados
- O tipo do modelo criado depende do problema sendo resolvido
  - Mais comuns:
    - Classificação (Predição)
    - Agrupamento

Classificação



Clusterização



- Nesse caso, a quantidade de classes é desconhecida. Cabe ao algoritmo determinar e agrupar os dados de acordo com suas características

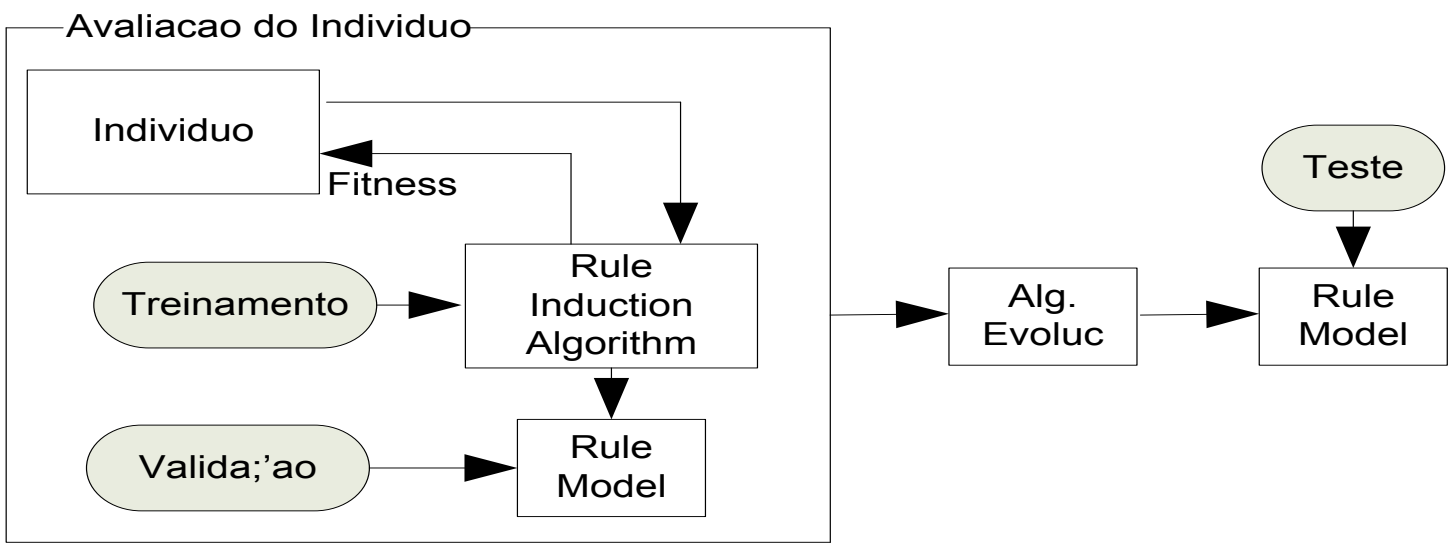
Trabalhando com Dados em Computação Natural

- Dados são divididos em 2 ou 3 partes (dependendo do problema)
  - Treinamento – utilizados para criar o modelo durante o cálculo da fitness
  - Validação – utilizado para avaliar o modelo criado, e determinar a medida de fitness
  - Teste – utilizado para testar o melhor indivíduo retornado

Aprendizado

- Em tarefas de aprendizagem, os dados devem ser divididos em pelo menos dois conjuntos:
  - Conjunto de treinamento, onde a fitness é calculada
  - Conjunto de teste, onde o melhor indivíduo retornado é avaliado

Exemplo


Avaliacao do Individuo
Individuo
Fitness
Treinamento
Rule Induction Algorithm
Valida;'ao
Rule Model
Alg. Evoluc
Teste
Rule Model

Validação Cruzada

- Garante Validação estatística dos resultados
- Como funciona?
  - Passo 1: os dados são divididos em *k* subconjuntos de mesmo tamanho
  - Passo 2: em cada instante um subconjunto é usado para teste e os demais para treinamento
- Isto é chamado *validação cruzada de fator k*
- Normalmente os subconjuntos são estratificados (divididas de forma que a distribuição das classes sejam mantidas) antes de realizar a validação cruzada
- Faz-se a média das estimativas de erro para obter o erro estimado geral

Validação Cruzada

- Método padrão para avaliação: validação cruzada estratificada com fator 10
- A realização de vários experimentos tem demonstrado que 10 é a melhor escolha
- A estratificação reduz a variância da estimativa
- Opção melhor: validação cruzada estratificada com fator 10 repetida (10 x)

Tuning automático de parâmetros

- Existem diversas ferramentas que testam diferentes combinações de parâmetros automaticamente
- Ferramenta muito utilizada atualmente: iRace
  - http://iridia.ulb.ac.be/irace/
- Baseado no algoritmo Iterated F-race (I/F-Race)
- Funciona em 2 fases:
  - Tuning
  - Teste

iRace (pacote R)

- Usa o teste estatístico não-paramétrico de Friedman para analisar a variância de várias configurações de parâmetros através de um ranking
- Cada parâmetro a ser otimizado segue uma distribuição normal (atributos numéricos) ou discreta (atributos categóricos)
- Ao longo das iterações, as distribuições são atualizadas de forma a aumentar a probabilidade de amostras parâmetros próximos aos melhores encontrados até o momento

Leitura Recomendada

- **Parameter Control in Evolutionary Algorithms,** Ágoston E. Eiben , Robert Hinterding , Agoston E. Eiben Robert Hinterding , Zbigniew Michalewicz, IEEE Transactions on Evolutionary Computation, 2000
- **A study of cross-validation and bootstrap for accuracy estimation and model selection**, R. Kohavi, Proc. of the 14th International Joint Conference on Artificial Intelligence 2 (12): 1137–1143, 1995
- The irace Package: Iterated Racing for Automatic Algorithm Configuration, Relatório técnico
  - http://iridia.ulb.ac.be/IridiaTrSeries/link/IridiaTr2011-004.pdf