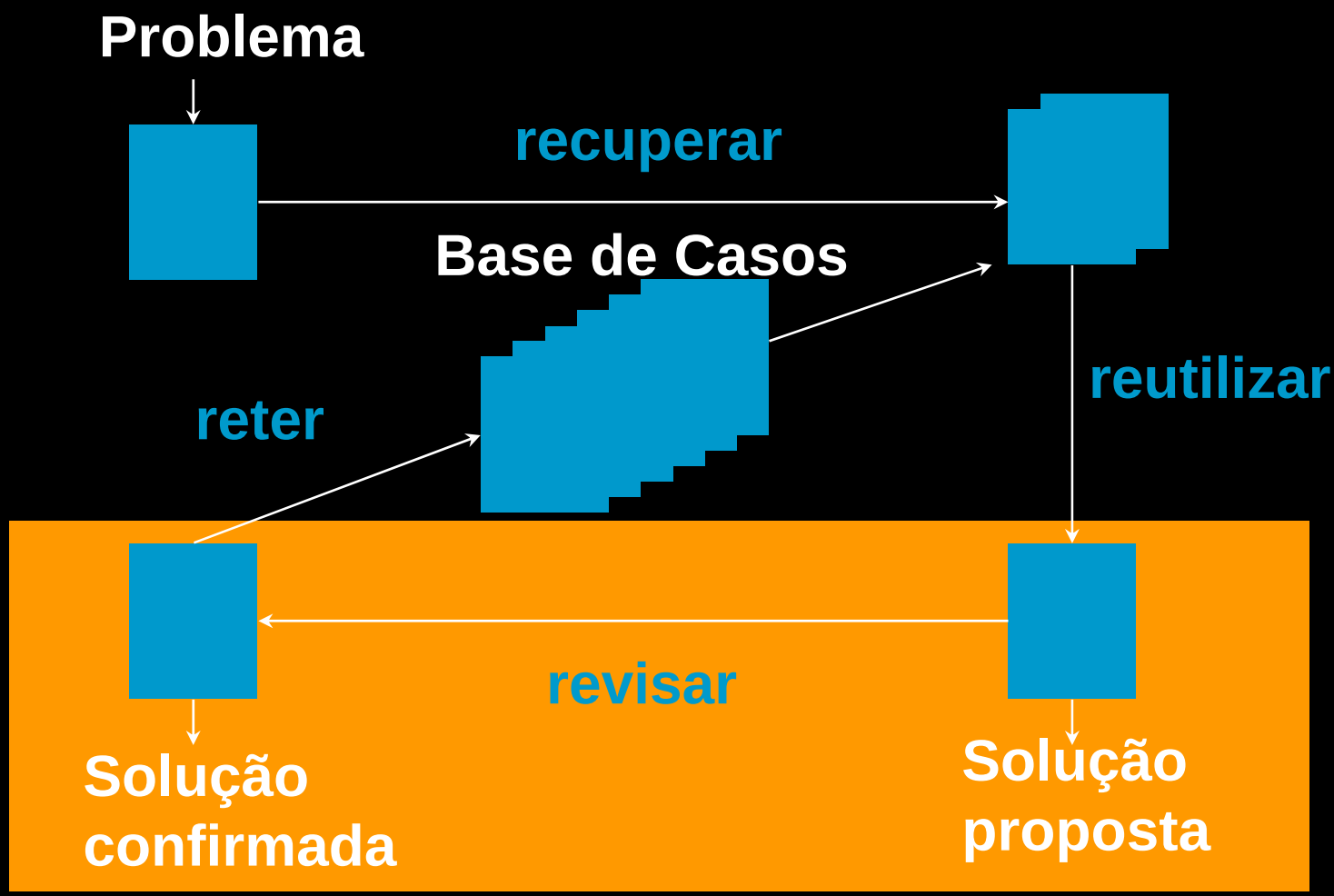
Raciocínio Baseado em Casos

5. Revisão e Aprendizagem

Prof. Aldo von Wangenheim
Disciplinas:
- Raciocínio Baseado em Casos - PPGCC/INE/UFSC
- Sistemas de Raciocínio e Gestão Baseados em Casos - EGC/UFSC

Ciclo de RBC – Revisão



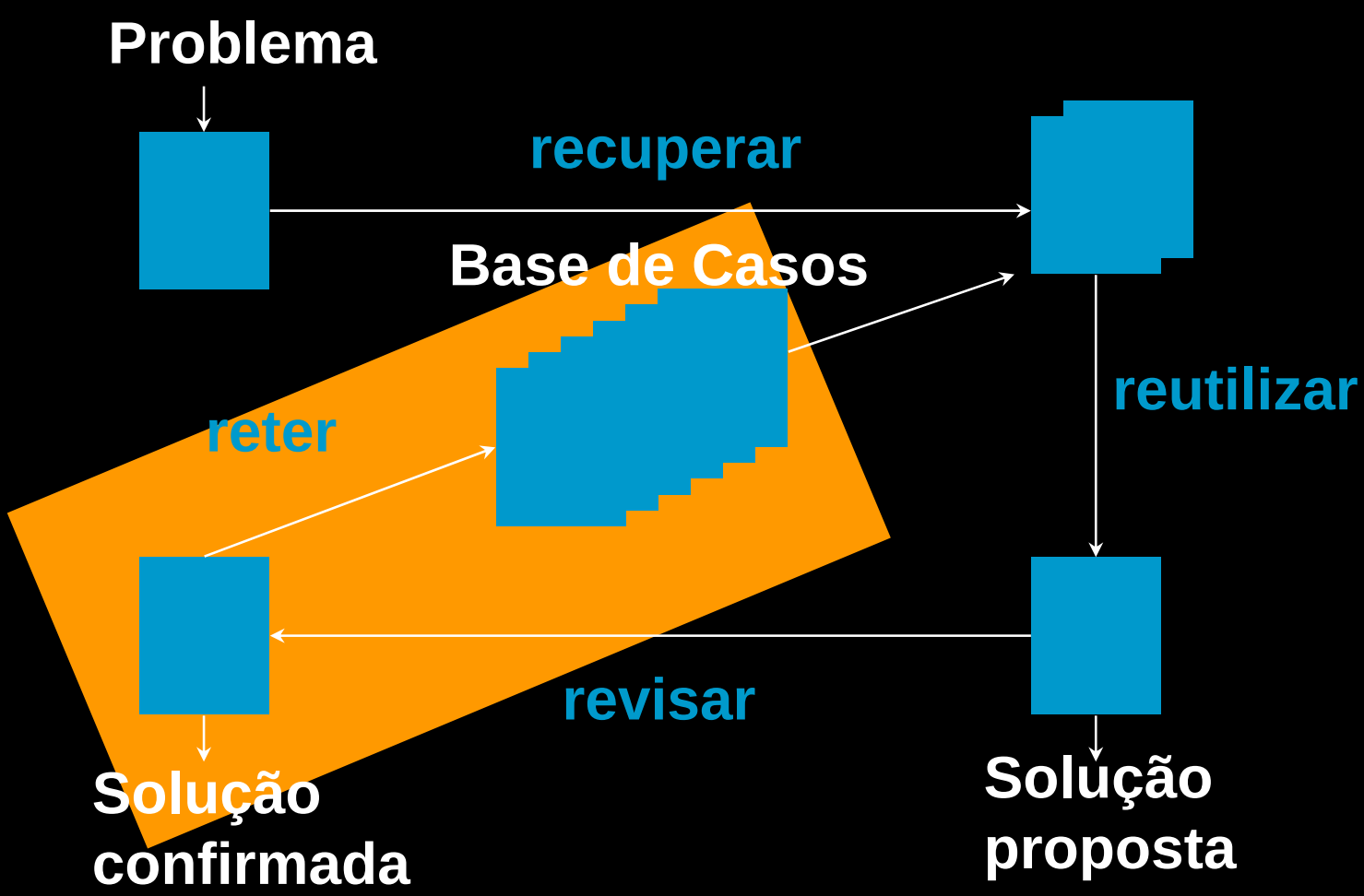
Revisão

- Revisão da aplicação da nova solução para resolver o problema atual
- A revisão consiste de duas tarefas:
  - avalie a solução gerada pelo reúso. Se for considerada como correta, aprenda com o sucesso e continue com a retenção do novo caso na base de casos.
  - Caso contrário, repare a solução para o caso, utilizando conhecimento específico sobre o domínio de aplicação ou informações fornecidas pelo usuário.
- Tamanho da revisão
  - sem melhoria
  - revisão da solução pela simulação
  - revisão da solução pela aplicação no mundo real
- Critérios da revisão
  - correção da solução
  - qualidade da solução
  - outros (p.ex. preferências do usuário)

Exemplos de revisão

- **Diagnóstico**:
  - novo problema: *não imprime texto preto*
  - solução sugerida: *trocar o cartucho de tinta preta*
  - usuário aplicou a solução à sua impressora em casa, mas o problema permanece: causa da falha: *causa real é falta de energia por problema na fonte de alimentação da impressora* e solução corrigida: *troca da fonte de alimentação*
- **Recuperação de informações**: agência virtual de viagens calculando a similaridade local para o destino de viagem exclusivamente na distância geográfica dos locais.
  - cliente quer ir ao Rio de Janeiro implicando que quer ir a uma grande cidade próxima a belas praias
  - sistema sugere um pacote de viagem para São Paulo (geograficamente mais perto do Rio)
  - cliente rejeita a oferta ⇨ modificação da medida de similaridade local
- **Planejamento**: planejamento de refeições como CHEF
  - substiuição de ingredientes da solução recuperados
  - falha observada, p.ex. verdura ficou crua
  - necessidade da adaptação também do procedimento de preparo, p.ex. cozinhar por 15 minutos ao invés de 5 minutos.

Ciclo de RBC - Reter novo caso



Retenção

Retenção de casos é o processo de incorporação, ao conhecimento já existente, daquilo que é útil de um novo episódio de solução de um problema.

- Objetivo é continuamente melhorar a performance do sistema RBC tornando-se um solucionador de problemas mais poderoso, com o passar do tempo e sua utilização.
  - Eficiência
  - Qualidade da solução sugerida
- Melhorar:
  - a base de casos, por meio de adição, modificação e deleção de casos
  - a medida de similaridade, p.ex., por meio do ajuste de pesos
  - a transformação da metodologia de solução, p.ex., por meio do ajuste das regras de adaptação de casos

Tipos de retenção

- Três tipos de retenção em sistemas RBC:
  - **Sem retenção de casos:** geralmente aplicado em domínios
  - A forma típica: **Retenção de soluções de problemas.** Assim que um novo problema é resolvido, a experiência é incorporado à base de casos como novo caso.
  - **Retenção de documentos.** Adquisição de novo conhecimento de forma assíncrona ao processo de solução de problemas, sempre que se encontrar disponível.
- Considerar o que?
  - Nova experiência (novo caso)
  - Performance do sistema:
    - Avaliação da similaridade
    - Importância dos atributos
  - Organização da base de casos (eficiência)
  - Adaptação da solução

Reter novos casos

- Razões para incluir novos casos na base:
  - melhorar competência do sistema (sem este caso, o sistema não pode achar uma solução correta)
  - melhorar eficiência (com este caso o sistema acha uma solução mais rápido, p.ex., menos esforço necessário para adaptação)
- Razões para não incluir um novo caso:
  - Aumenta esforço de recuperação
  - Aumenta necessidade de memória
- Razões para excluir casos da base:
  - redução do esforço de recuperação e memória
  - caso não é mais válido
  - caso é ultrapassado

Processo de retenção

- **Extração de conhecimento:** seleção da informação que deveria ser capturada
  - Fontes para novas experiências podem ser:
    - **Para sistemas de retenção de documentos**: documentos, manuais, descrições de produtos, jurisprudência, protocolos de patientes, etc.
    - **Para sistemas de retenção de soluções de problemas:** soluções de problemas, estruturas do caminho de solução, históricos de adaptação, etc.
- **Indexação de casos**: decidir que índices devem ser utilizados e como estruturar o espaço de busca.
  - Na verdade, um problema de aquisição de conhecimento:
    - Solução trivial: utilização de todos os atributos como índices
    - Métodos de aprendizagem para determinação de características relevantes
- **Integração na base de casos**:
  - Atualização dos *knowledge containers*

Aprendizagem

Aprendizado Baseado em Casos

- Durante o processo de aprendizado, é gerada por meio da entrada uma seqüência C_1,..., C_k de casos. Partindo-se de uma base de casos vazia CB = f e de uma medida de similaridade inicial sim_0, é gerada uma seqüência de tuplas (CB_1,sim_1), ...,(CB_k,sim_k) com $CB \subseteq \{C_1,..., C_k\}$.
- O objetivo do processo de aprendizado baseado em casos é, em seu extremo, descrever um conceito C exatamente por meio de uma tupla (CB_n,sim_n). Durante este processo, o conceito a ser aprendido é aproximado com a seqüência $C_1 = (CB_1,sim_1)$, ..., $C_k = (CB_k,sim_k)$ de conceitos.
- Um conceito C foi aprendido por um classificador comparador de casos, quando $\exists n\ \forall m\ C = (CB_n,sim_n) = (CB_m,sim_m)$, i.e., durante a entrada de mais casos C_i, $i \geq n$ a descrição do classificador (CB_n,sim_n) não se altera mais.

Algoritmos para o aprendizado baseado em casos - 1

- **Algoritmos de Aprendizado de Instâncias** (*Algoritmos-IBL):*
  - aprendem a categorizar um conjunto de classes de objetos de forma incremental com base em exemplos de instâncias dessas categorias
  - partem do princípio de que instâncias similares pertencem a categorias similares, e criam essas categorias em função das similaridades detectadas
- Cada caso é representado pelo mesmo conjunto de atributos que define um *espaço de instância* n-dimensional:
  - exatamente um desses atributos corresponde ao *atributo de categoria*
  - os outros atributos são *atributos preditores*
- Uma **categoria** é o conjunto de todos os casos em um espaço de instância que possui o mesmo valor para seu atributo de categoria, assumindo que:
  - existe exatamente um único atributo de categoria
  - as categorias são disjuntas
  - atributos preditores são definidos sobre conjuntos de valores completamente ordenados

Algoritmos para o aprendizado baseado em casos - 2

- Objetivo da aprendizagem:
  - Dado um conjunto de treinamento de exemplos classificados
  - ⇨ Construir uma descrição que predizerá corre s de exemplos futuros
- Dado:
  - Medida de similaridade: Similaridade (x,y) =
    n = nº de atributos,
    $f(x_i, y_i) = (x_i - y_i)^2$ (valores numéricos),
    $f(x_i, y_i) = (x_i \neq y_i)$ (booleano e simbólico)
  - Sequência de casos de treinamento $C_1, C_2, ..., C_n$
- Várias abordagens
  - **IBL1:** Incluir cada caso na base
  - **IBL2:** Incluir só casos que foram classificados com erro utilizando a base atual
  - **IBL3:** Incluir só casos que foram classificados com erro utilizando a base atual e remover casos „ruins“

Aprendizado Baseado em Instâncias

Durante o processo de aprendizado é gerada através da entrada uma seqüência $P_1,\ldots, P_k$ de padrões. Partindo-se de uma base de padrões vazia $CP = \phi$ e de uma medida de similaridade inicial sim_0 , é gerada uma seqüência de tuplas $(CP_1,sim_1), \ldots,(CP_k,sim_k)$ com $CP \subseteq \{C_1,\ldots, C_k\}$.

O objetivo do processo de aprendizado baseado em instâncias é, em seu extremo, descrever um conceito C exatamente através de uma tupla (CP_n,sim_n). Durante este processo, o conceito a ser aprendido é aproximado através da seqüência $C_1 = (CP_1,sim_1), \ldots, C_k = (CP_k,sim_k)$ de conceitos.

Um conceito C foi aprendido por um classificador comparador de padrões, quando $\exists n\ \forall m\ C = (CP_n,sim_n) = (CP_m,sim_m)$, i.e., durante a entrada de mais casos C_i, $i \geq n$ a descrição do classificador (CP_n,sim_n) não se altera mais.

Aprendizado de Máquina Baseado em Instâncias

- Algoritmos para o aprendizado baseado em padrões
  - Algoritmos de Aprendizado de Instâncias, Algoritmos-IBL.
  - Aprendem a categorizar um conjunto de classes de objetos de forma incremental com base em exemplos de instâncias dessas categorias.
  - Partem do princípio que instâncias similares pertencem a categorias similares.
- Única entrada: um conjunto de padrões de treinamento
- Saída é uma descrição de conceito
  - pode ser utilizada para realizar predições sobre valores de características esperados em padrões subseqüentemente apresentados.

Aprendizado de Máquina Baseado em Instâncias

- Cada padrão é representado pelo mesmo conjunto de EIs
  - define um espaço de instância n-dimensional.
  - exatamente uma dessas EIs corresponde ao atributo de categoria,
  - as outras EIs são atributos preditores.
- Categoria
  - é o conjunto de todos os casos em um espaço de instância que possue o mesmo valor para seu atributo de categoria.

Aprendizado de Máquina Baseado em Instâncias

- Resultado primário de algoritmos IBL
  - descrição conceitual DC, função que mapeia padrões a categorias:
    - dado um padrão, ela proverá uma classificação que é o valor predito para o atributo de categoria deste padrão.
  - Uma descrição conceitual baseada em instâncias inclue
    - um conjunto de casos armazenados e possivelmente,
    - informações sobre a sua performance classificatória no passado
  - Este conjunto de padrões pode mudar após o processamento de cada padrão de treinamento.
  - Conceitos: descritos de forma implícita
    - através da função de similaridade, da função de classificação e dos casos armazenados em uma base de padrões.

IBL - Instance-Based Learning: Como Algoritmos Aprendem Simbolicamente

- 3 componentes presentes em algoritmos-IBL:
  - **Função de similaridade**. Computa a similaridade entre uma instância de treinamento i e as instâncias em uma dada descrição conceitual. Retorna valores numéricos de similaridade.
  - **Função de classificação**. Recebe o resultado da função de similaridade e os registros de performance de classificação na descrição conceitual. Provê uma classificação para i.
  - **Atualizador do descritor conceitual**. Mantém registro da performance classificatória e decide quais instâncias incluir na descrição conceitual.

IBL - Instance-Based Learning: Como Algoritmos Aprendem Simbolicamente

- 5 dimensões para supervisionar a performance:
  - **Generalidade**. Classes de conceitos que podem ser aprendidos.
    - IBL é capaz de aprender quaisquer conceitos dados pela união de um número finito de hipercurvas fechadas de tamanho finito.
  - **Acurácia**. É a acurácia da classificação provida pela DC.
  - **Taxa de aprendizado**. É a velocidade com a qual a acurácia classificatória aumenta durante o aprendizado.
  - **Custos de incorporação**. Custos que decorrem da atualização da DC através da inclusão de uma instância única.
  - **Requisitos de armazenamento**. Tamanho da DC, definida como o número de instâncias que necessitam ser salvas para prover uma performance classificatória adequada.

IBL1

- Opção de Medida de Similaridade sugerida por David Aha:

$$sim(x,y) = -\sqrt{\sum_{i=1}^{n} f(x_i, y_i)}$$

IBL1 original

- Idêntico ao algoritmo nearest neighbour,
  - capaz de processar padrões de forma incremental e possue uma política simples para lidar com valores desconhecidos.
- As funções de similaridade e de classificação
  - provêem uma descrição conceitual extensional a partir do conjunto de padrões salvos.
    - pode-se determinar facilmente quais instâncias no espaço de instância serão classificadas por qual dos casos armazenados
- Performance boa
  - realiza número desnecessário de cálculos de similaridade

IBL1

{DC: Descritor Conceitual}

1. Inicialize DC:= 0.

2. PARA cada x ∈ conjunto de treinamento FAÇA

2.1. PARA cada y ∈ CD FAÇA
Sim[y] := sim(x,y)

2.2 y_{max} := algum y ∈ CD com MAX Sim[y]

2.3 SE classe(x) = classe(y_{max})
ENTÃO classificação := correta
SENÃO classificação:= incorreta

2.4 DC := DC ∪ {x}

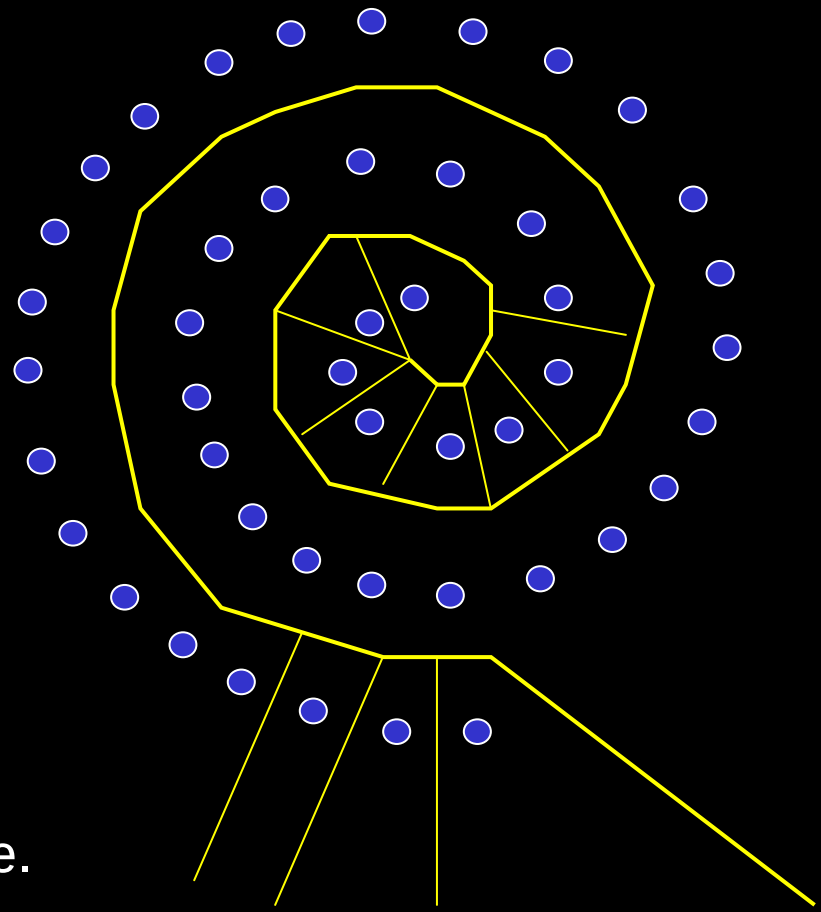
Algoritmo IBL1 na Nomenclatura-RBC:

1. Inicialize Base_de_casos:= 0.
2. PARA cada x $\in$ conjunto de treinamento FAÇA
2.1. PARA cada y $\in$ Base_de_casos FAÇA
Sim[y] := sim(x,y)
2.2 y_{max} := algum y $\in$ Base_de_casos com MAX Sim[y]
2.3 SE classe(x) = classe(y_{max})
ENTÃO classificação := correta
SENÃO classificação:= incorreta
2.4 Base_de_casos := Base_de_casos $\cup$ {x}

O que faz o Nearest Neighbour sobre um conjunto de dados intrincados no R^2 ?

Geração de Células em Torno de cada Padrão como Centróide -> Voronoi por força bruta.

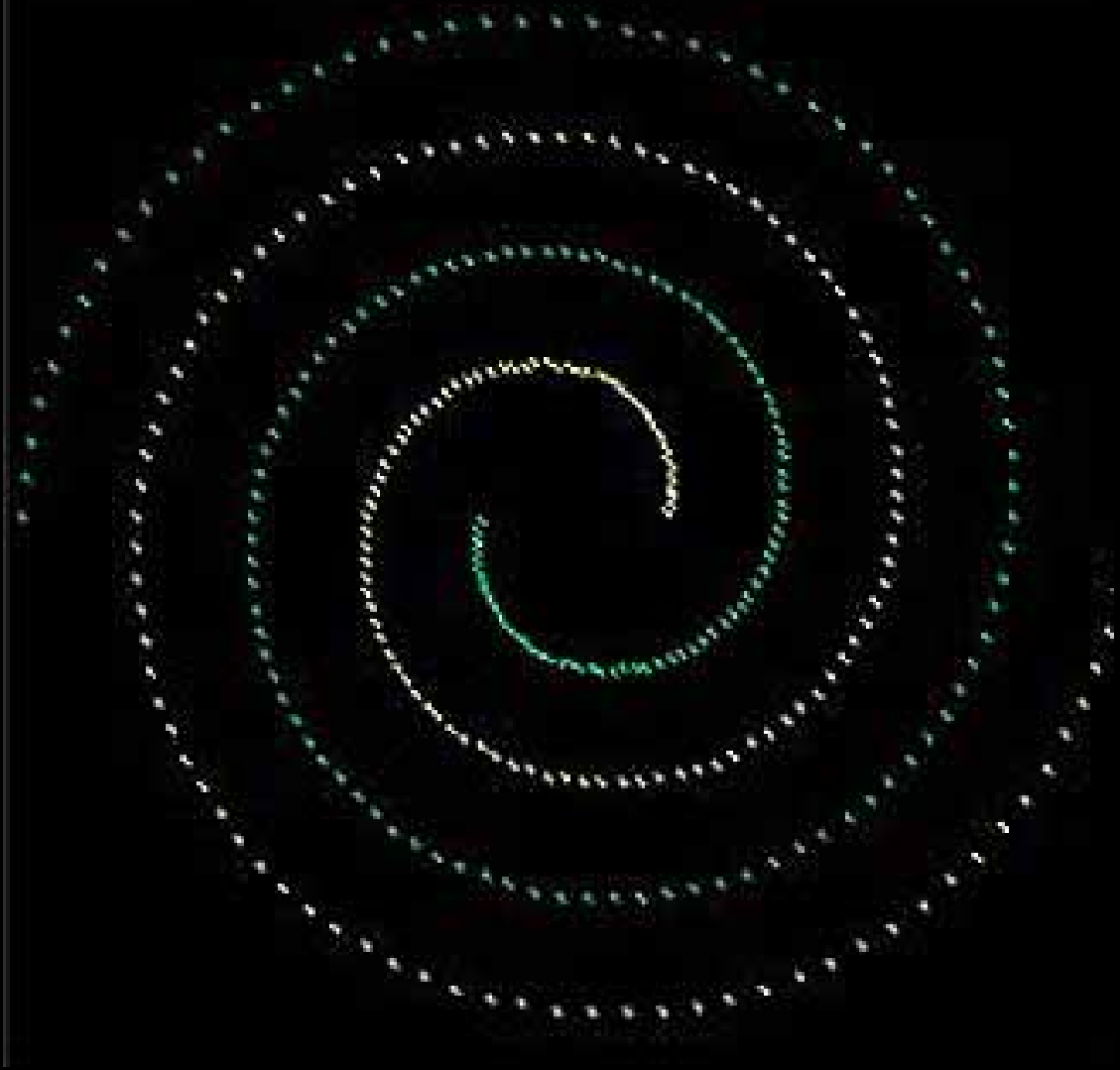
Cada ponto pode representar uma classe.



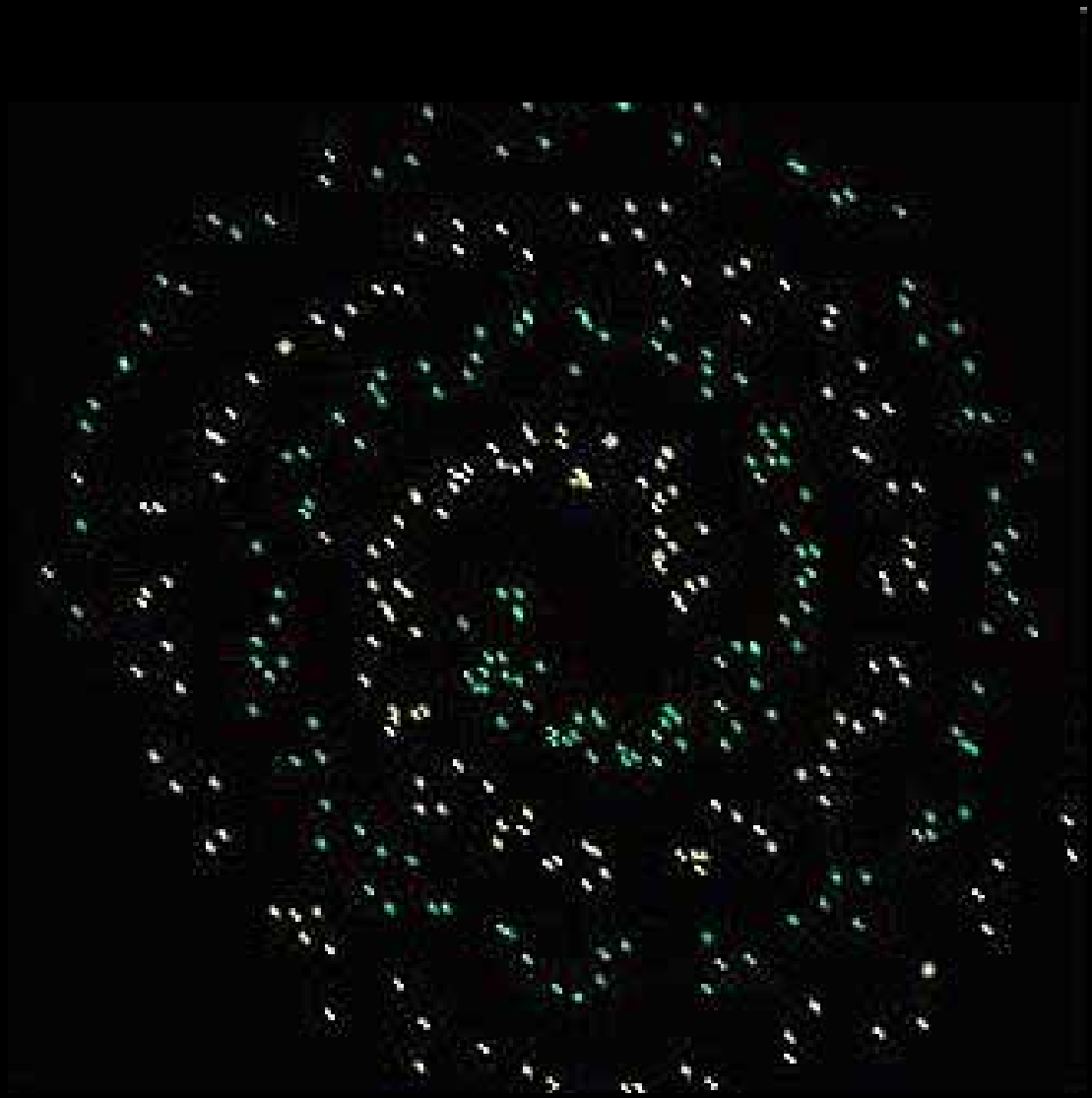
Espiral Dupla: Duas Classes Apenas

- Na espiral dupla, cada classe é representada por um conjunto de pontos, organizados em espiral
  - os dois conjuntos de dados não são linearmente separáveis.

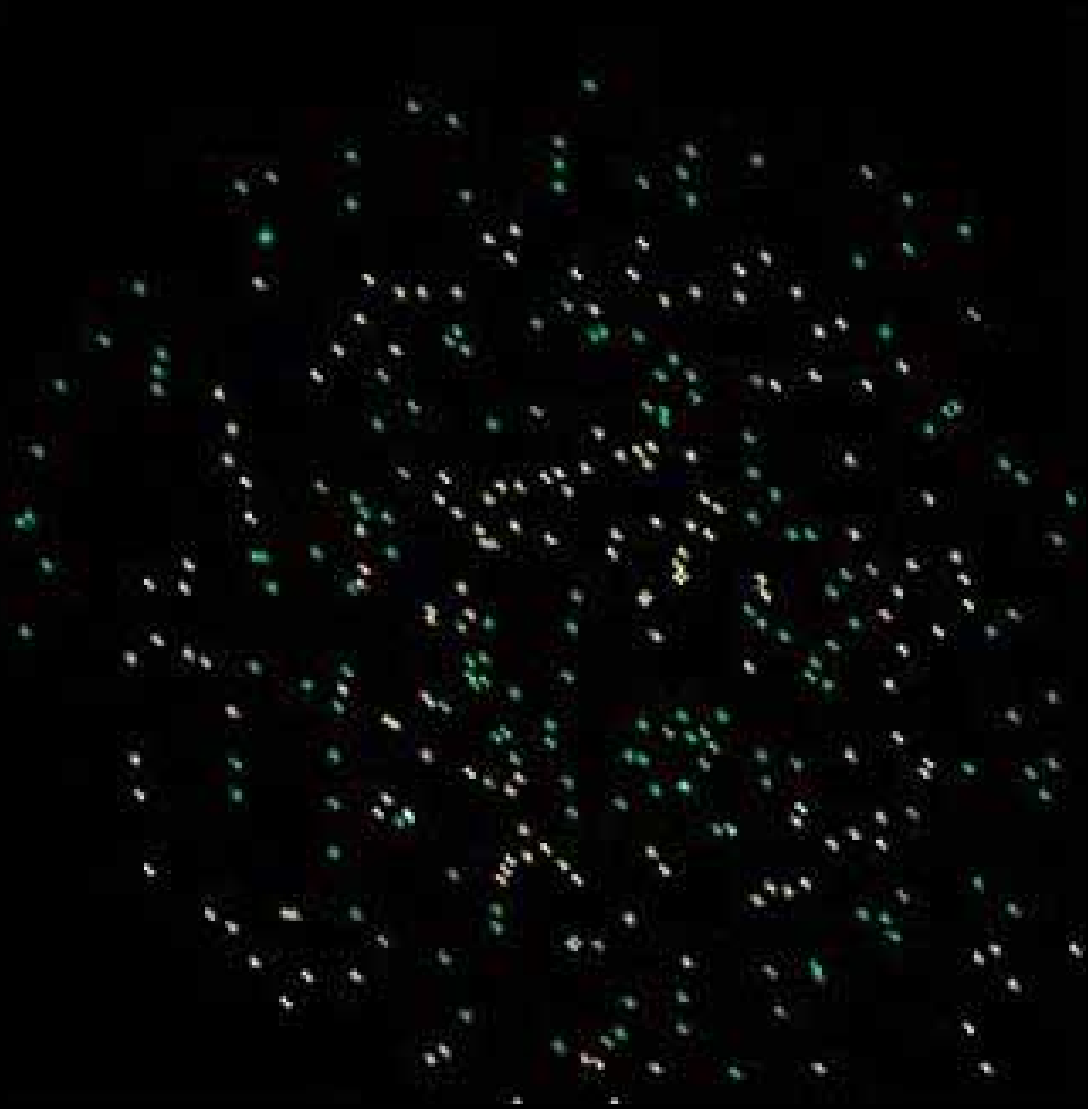
Espiral dupla sem ruído



Espiral dupla com pouco ruído



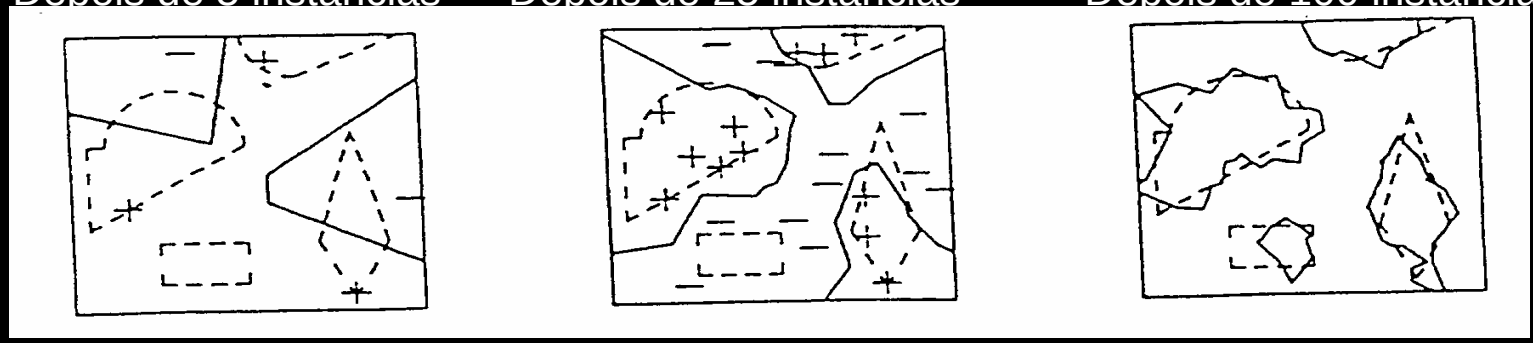
Espiral dupla com bastante ruído



IBL1: Performance

- Implementa a função de classificação de *nearest neighbor* que permite de determinar facilmente quais instâncias no espaço de instância serão classificadas por quais dos casos armazenados.
- IBL1 possui uma performance relativamente boa
- Exemplo: 100 instâncias de treinamento aleatórias e retiradas de uma distribuição uniforme e 4 conceitos alvos.

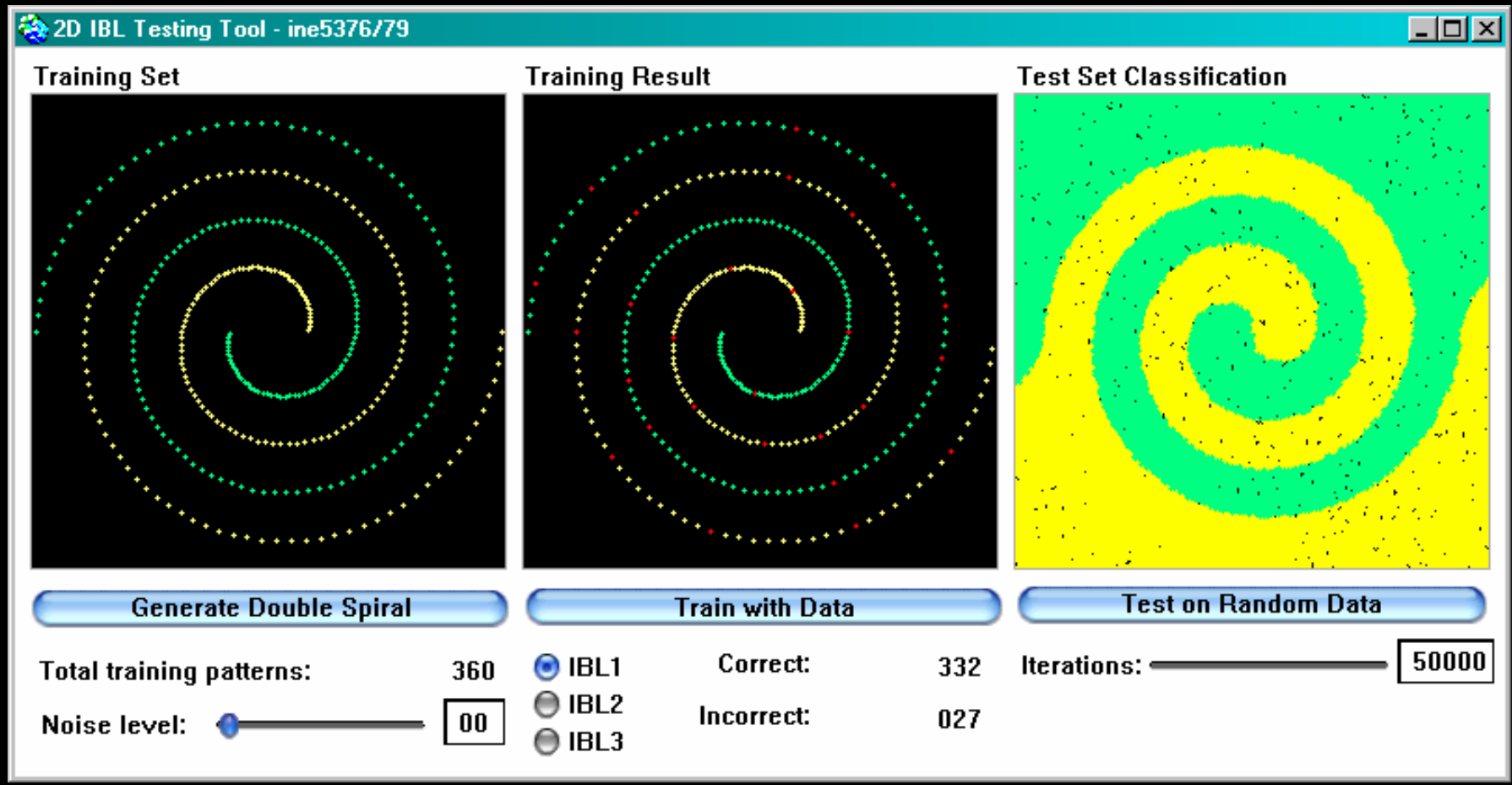
Depois de 5 instâncias Depois de 25 instâncias Depois de 100 instâncias



Base de casos = linha sólida
Conceito alvo = linha pontilhada

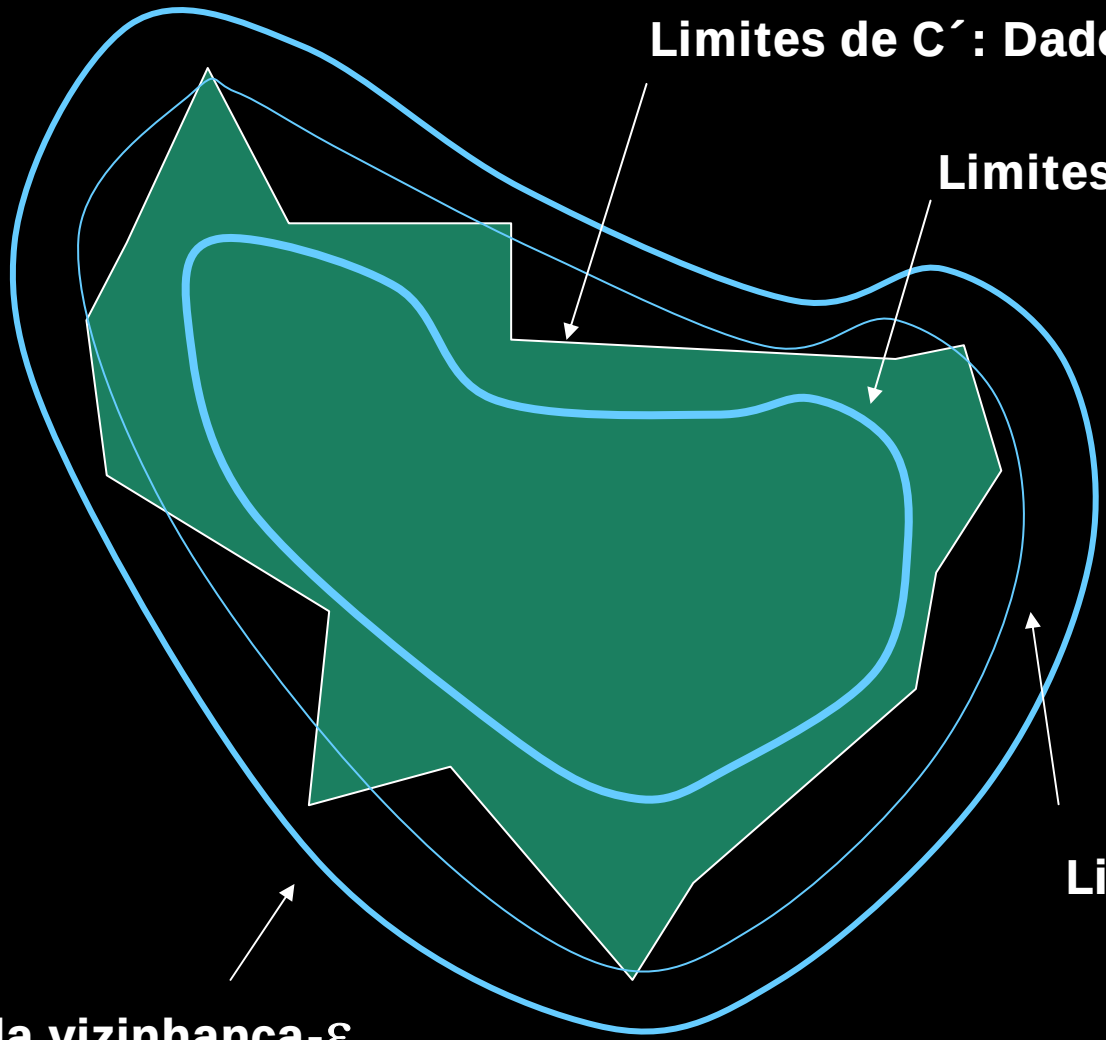
- Mas, realiza um número desnecessário de cálculos de similaridade durante a predição

Pausa: Demonstração IBL1 no R^2



IBL+ ?

- **esfera-ε** : A esfera-ε em torno de um ponto x em R^n é o conjunto de pontos dentro de uma distância ε de x: { y pertence a R^n | distância(x, y) < ε }. Em um espaço bidimensional é um círculo.
- **núcleo-ε** : O núcleo-ε de um conjunto C é constituído por todos os pontos de C tal que a esfera-ε em torno deles está contida em C.
- **vizinhança-ε** : A vizinhança-ε de C é definida como o conjunto de pontos que estão dentro de uma distância ε de um ponto qualquer em C.


Limites de C´: Dados observados
Limites do núcleo-ε
Limites reais de C
Limites da vizinhança-ε

IBL2

- **Idéia básica**: não são necessárias todas as instâncias para permitir uma boa descrição dos limites de um conceito.
  - necessitamos apenas dos conceitos na vizinhança dos limites do espaço deste conceito e de seu núcleo para ser utilizado como protótipo.
  - podemos representar um conceito C armazenando apenas as instâncias que se encontram no espaço os limites do núcleo-ε e da vizinhança-ε.
  - podemos portanto economizar muito espaço armazenando apenas estes padrões.

IBL2

- Este conjunto não é conhecido
  - é aproximado através das instâncias contidas em C´ e classificadas erroneamente por IBL no algoritmo IBL2.
  - IBL2 é idêntico a IBL1, porém ele salva apenas instâncias classificadas erroneamente.
- Raciocínio
  - o que importa é termos uma representação detalhada dos limites de um conceito
  - a maioria dos casos erroneamente classificados se encontram próximos às bordas deste caso no espaço de instância.
  - IBL2 reduz drasticamente as necessidades de armazenamento.

IBL2

- O maior problema de IBL2 é ruído
  - que rapidamente degrada sua performance,
  - podendo levar a uma acurácia bastante inferior à de IBL1.
- Ocorre porque IBL2 salva todos os exemplos de treinamento com ruído que classifica erroneamente e depois os utiliza.

IBL2

(DC: Descrição Conceitual)

1. Inicialize DC:= 0.

2. PARA cada x ∈ conjunto de treinamento FAÇA

2.1. PARA cada y ∈ DC FAÇA
Sim[y] := sim(x,y)

2.2. y_{max} := algum y ∈ DC com MAX Sim[y]

2.3. SE classe(x) = classe(y_{max})

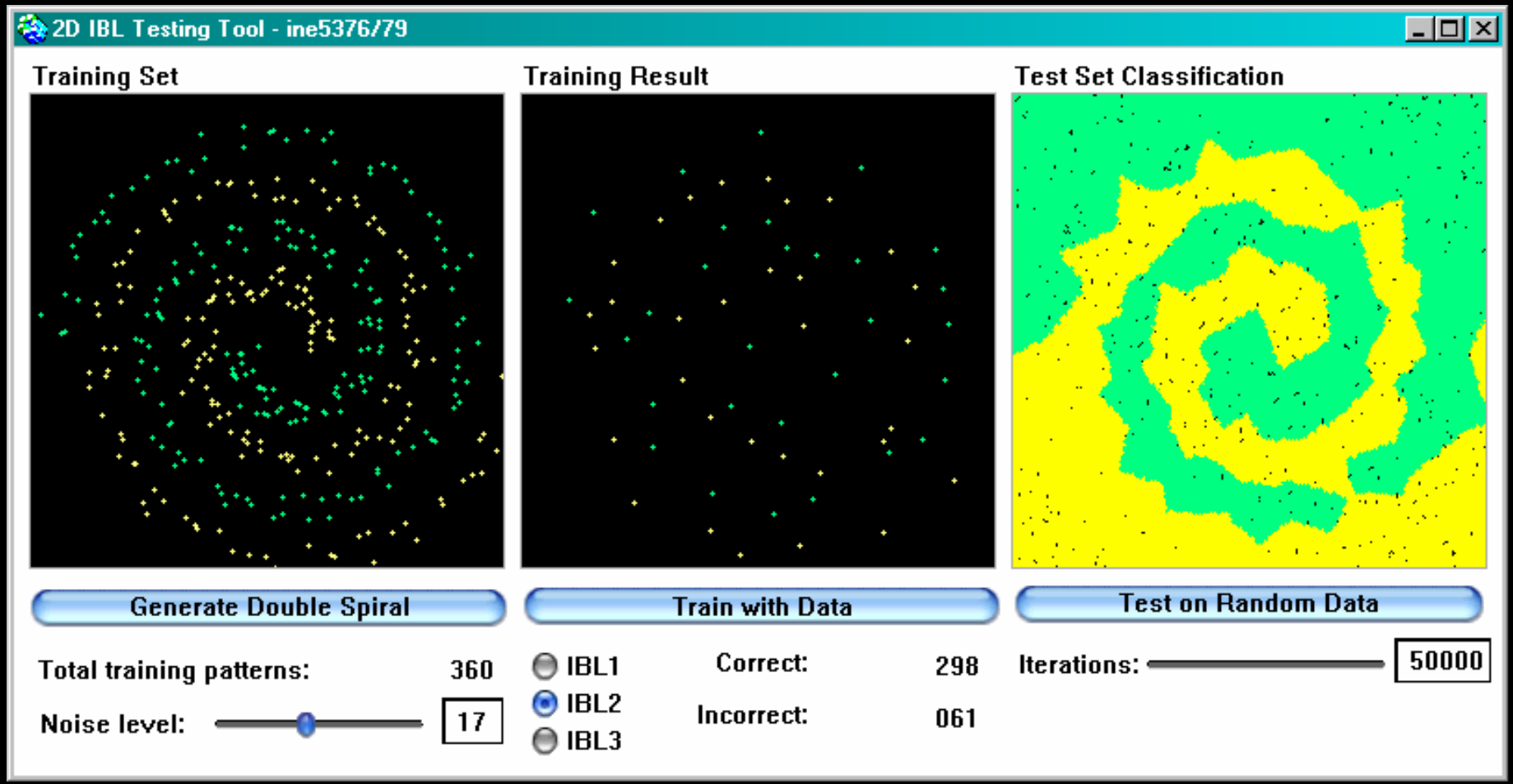
ENTÃO classificação := correta

SENÃO

2.3.1. classificação:= incorreta

2.3.2. DC := DC ∪ {x}

Pausa: Demonstração IBL2 no R^2



IBL3

- Extensão de IBL2 tolerante a ruídos
  - Emprega uma estratégia de coleta de evidências de “esperar para ver” para averiguar quais das instâncias salvas vão funcionar bem durante a classificação.
    - Sua função de similaridade é idêntica à de IBL2.
    - Função de classificação e o algoritmo de atualização diferem
  - Registra quais padrões são bons classificadores
    - registra a freqüência com a qual um padrão armazenado, quando escolhido como o mais similar ao padrão atual, correspondeu ao valor do atributo-meta do padrão atual.
  - Elimina da Base de Casos aqueles que são inúteis ou maus classificadores
    - IBL3 mantém um registro do número de tentativas corretas, associado a cada caso armazenado.

IBL3

- O *registro de classificação* espelha a performance classificatória de um determinado padrão
  - para cada novo padrão de treinamento apresentado, registros de classificação são atualizados para todos os padrões salvos que são tão similares ao padrão apresentado como o aceito como mais similar a este.
  - se nenhum dos padrões salvos é ainda suficientemente similar, é utilizada uma política que simula o comportamento do algoritmo quando pelo menos uma instância for aceitável
    - um número randômico ***r*** gerado na faixa [1,*n*], onde ***n*** é o número de padrões salvos e os registros classificatórios dos ***r*** padrões salvos mais similares são atualizados.

IBL3

- Emprega teste de significância para determinar quais padrões são bons classificadores e quais são supostos conterem ruído.
  - Os primeiros são então utilizados para a classificação de padrões subseqüentemente apresentados
  - Os outros são descartados da descrição conceitual.
- IBL3
  - aceita uma instância se a sua acurácia classificatória for significativamente superior do que a freqüência observada de sua classe e
  - remove instâncias da descrição conceitual se sua acurácia for significativamente inferior.

(DC: Descrição Conceitual)

1. Inicialize DC:= 0.

2. PARA cada x ∈ conjunto de treinamento FAÇA

2.1. PARA cada y ∈ DC FAÇA
Sim[y] := sim(x,y)

2.2. SE ∃ {y ∈ DC| aceitável(y)}

ENTÃO y_{max} := algum y aceitável ∈ CD
com MAX Sim[y]

SENÃO

2.2.1. i := valor randômico em [1,|CD|]

2.2.2. y_{max} := algum y ∈ DC : i-ésima
instância mais similar a x

2.3. SE classe(x) = classe(y_{max})

ENTÃO classificação := correta

SENÃO

2.3.1. classificação:= incorreta

2.3.2. DC := DC $\cup$ {x}

2.4. PARA cada y $\in$ DC FAÇA

SE Sim[y] $\geq$ Sim[y_{max}]

ENTÃO

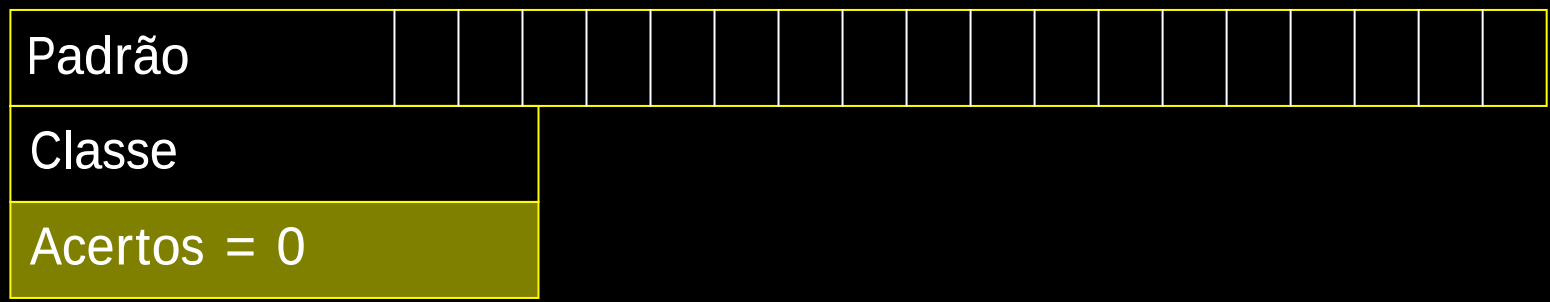
2.4.1. Atualize registro classificação y

2.4.2 SE registro y significantemente pobre

ENTÃO DC := DC - {y}

IBL3 - Comentários de Implementação

- Como implementar a estrutura de dados?
  - Inclua o registro de acertos na estrutura de dados do padrão armazenado na Base de Casos:



IBL3 - Comentários de Implementação

- Como implementar **`aceitável(y)`** ?
  - Aceitável é suficientemente similar. Significa que a distância `Sim[y]` calculada com `sim(x,y)` deve ser menor que algum valor α de tolerância.
    - Defina a sua tolerância como um parâmetro do algoritmo.
  - Você pode inclusive deixar o algoritmo melhor, fazendo esta tolerância ser variável. À medida de |BaseDeCasos| vai ficando maior, o α pode ir sendo reduzido gradualmente.
    - Assim no início do treinamento o algoritmo é extremamente tolerante e depois, à medida que a |BaseDeCasos| fica mais completa, vai ficando mais restritivo.

IBL3 - Comentários de Implementação

- Para quais elementos eu atualizo o registro de classificação ?
  - O algoritmo sugere todos cuja similaridade seja **maior ou igual** a Sim[y].
    - Como trabalhamos com um valor real, **igual** é extremamente difícil de acontecer.
    - A opção maior só vai contecer quando não houver nenhum aceitável e não é problema.
  - Defina uma função **`equivalente(Sim[y], Sim[y2])`**, baseada em um parâmetro β do algoritmo, que especifica uma tolerância para esta equivalência de similaridades.
    - Isto pode ser expresso em % e definido pelo usuário. Experimente começar com 5%.

IBL3 - Comentários de Implementação

- Quais elementos eu elimino da Base de Casos ?
  - O algoritmo sugere todos significantemente pobres.
    - Inicialmente acertos(y) será 0 para todos os padrões.
    - A taxa de acertos de um padrão cresce de acordo com a inclusão de novos padrões na DC.
    - Padrões recém-inseridos terão taxa de acertos = 0 por um bom tempo.
  - Utilize um contador adicional indicando quantas vezes um elemento participou em uma classificação.
    - Somente consideramos um elemento para eliminação se ele participou um número mínimo χ de vezes.
    - A taxa ε de acertos/participação deve ser o critério de eliminação.

IBL3 - Comentários de Implementação

- O que mais devo considerar na eliminação da Base de Casos ?
  - O número de elementos por classe pode ser bem diferente.
    - Em função disso, úm número de acertos significantemente pobre é um número que varia de classe para classe.
    - David Aha sugere que se calcule a taxa relativa de acertos de um elemento para tomar essa decisão:
      n° de acertos do elemento / n° de elementos na classe
    - Isto significa que o algoritmo necessita de uma lista adicional contendo o número de elementos de cada classe, atualizada sempre que um novo elemento é incluído.
    - Faça o limiar de eliminação χ ser essa taxa, caso seja ε maior que um limiar.

IBL4

- O IBL4 é uma extensão do IBL3 com o objetivo de reduzir a sensibilidade a atributos irrelevantes
  - Os algoritmos IBL apresentados até agora tratavam todos os atributos de um padrão com o mesmo grau de relevância;
  - Esse novo tratamento aumenta o nível de acurácia do método.
- Se difere do IBL3:
  - Função de Similaridade;
  - Função de Classificação;
  - Atualizador da Descrição do Conceito.

IBL4

- Função de Similaridade
  - A Função de Similaridade é dependente do conceito (classe);
  - Os atributos são associados a pesos, que determinam o grau de relevância para definição de um determinado conceito;
  - Todos os pesos dos atributos são atualizados após cada instância de treinamento.

$$similaridade(c, x, y) = -\sqrt{\sum_{i=1}^{n} peso_{c_i}^2 * f(x_i, y_i)}$$

Onde peso c_i é o peso do atributo i, no conceito c

IBL4

- Função de Classificação
  - As instâncias são classificadas com relação a um determinado conceito. Assim, cada descrição de um conceito agrupa todas as instâncias incorretas juntas, independente de suas outras classificações com relação a outros conceitos.

- Atualizador da Descrição do Conceito
  - Toda vez que uma instância *x* é classificada, o seu vizinho mais próximo *y*, com relação à descrição do conceito *c*, é usado para atualizar os pesos.

IBL4

DC = 0

PARA CADA x $\in$ conjunto_treinamento FAÇA

1. PARA CADA y E DC FAÇA
   Sim[y] = sim(x,y)

2. SE $\exists${y $\in$ DC | aceitável(y)} ENTAO
   y_{max} = algum aceitável y $\in$ DC com MAX Sim[y]
   SENAO
   i = valor aleatório $\in$ [1,|DC|]
   y_{max} = algum y $\in$ DC que é a i-ésima instância mais similar a x

3. SE classe(x) = classe(y_{max}) ENTAO
   classificacao = correta
   SENAO
   classificacao = incorreta
   DC = DC U {x}

IBL4

4. PARA CADA y $\in$ DC FAÇA
   SE Sim[y] >= Sim[y_{max}] ENTAO
   Atualiza o registro de classificação de y
   SE registro de classificação de y é significativamente pobre ENTAO
   DC = DC – {y}

5. PARA CADA atributo i FAÇA
   diferença = $|x_i - y_{max_i}|$
   SE classe(x) = classe(ymax) ENTAO
   peso_acumulado c_i = peso_acumulado c_i + (1 - λ) * (1 - diferença)
   SENAO
   peso_acumulado c_i = peso_acumulado c_i + (1 - λ) * (diferença)

   peso_normalizado c_i = peso_normalizado c_i + (1 - λ)
   peso c_i = max(((peso_acumulado c_i / peso_normalizado c_i) – 0,5), 0)

IBL5

- O IBL5 é uma variação do IBL4 com o objetivo de contornar o problema de padrões com atributos ausentes, não tratado pelos algoritmos anteriores
  - Durante o calculo de similaridade entre *x* e *y*, utiliza apenas os atributos existentes em ambas as instâncias;
  - Caso um atributo não exista em *x* ou *y* é ignorado pela função.
- Se difere do IBL4:
  - Função de Similaridade;
  - No Passo 5, os passos 5.1 e 5.4 só são processados se ambos_conhecidos(x_i,y_i) = 1.

IBL5

- Função de Similaridade

$$similaridade(c,x,y) = -\sqrt{\sum_{i=1}^{n} peso_c^2 * dif(x_i, y_i)^2}$$

$$onde,$$

$$dif(x_i, y_i) = \begin{cases} |x_i - y_i| & \text{se ambos\_conhecidos}(x_i,y_i) = 1 \\ 0 & \text{caso contrário} \end{cases}$$

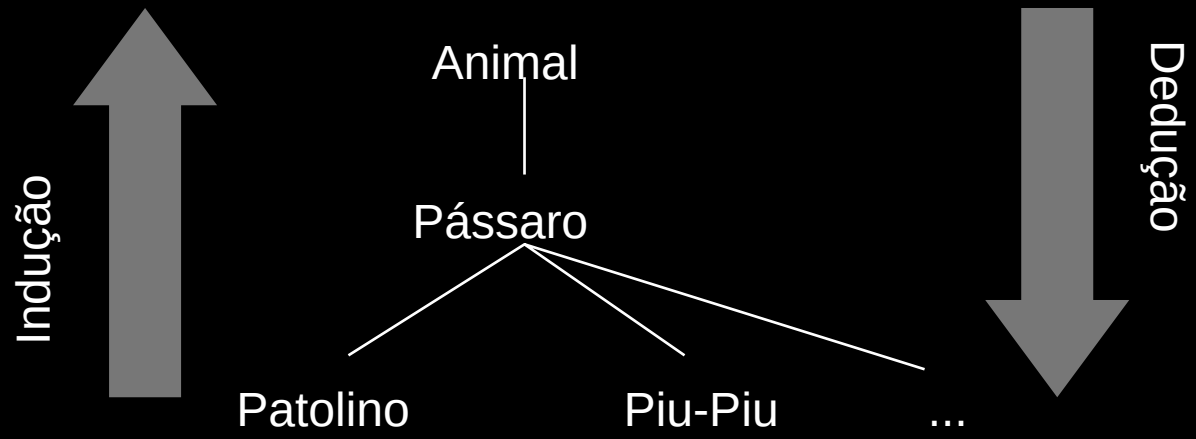
- ambos_conhecidos(x_i,y_i) = 1 se ambos os valores de atributo, tanto em *x* quanto em *y*, são conhecidos, e 0 caso contrário.

Aprendizado baseado em casos

- Métodos de aprendizado simbólicos: representação explícita do conceito aprendido, p.ex., por uma regra, árvore de decisão
- Métodos baseados em casos: representação implícita dos conceitos por meio da função de similaridade, da função de classificação e dos casos armazenados na base de casos.
- Vantagens:
  - Simplicidade
  - Redução do tamanho de armazenamento de casos
  - Usado para tarefas de classificação (classificar instâncias para conceitos não explicitamente definidos)
  - Baixos custos de modificação

Indução

Uso de exemplos para chegar em conclusões generalizadas



- Extrapolar de um conjunto dado de exemplos para fazer predições corretas sobre exemplos futuros.