**Capitulo 2 – Estado da Arte**

**(Descrição do capítulo)**

* 1. **Revisão da Literatura**

A combinação de opiniões especialistas é um tema estudado desde a metade do século XX. No início, os estudos eram voltados para aplicações como economia, democracia e decisões militares. Um dos primeiros modelos de aprendizagem de um sistema com múltiplos especialistas é a arquitetura *Pandemonium*, descrita por Oliver Selfridge em 1959. Posteriormente, alguns estudos importantes foram realizados sobre a combinação de especialistas usando diferentes termos e abordagens, mas a área de sistemas de múltiplos classificadores (*MCS – Multiple Classifier Systems*) ou *ensemble learning* tornou-se popular a partir da década de 1990 [1].

Hansen e Salamon (1990) mostraram a propriedade de redução da variância em um sistema de *ensemble,* e que o desempenho de generalização de uma rede neural pode ser melhorado usando um conjunto de redes neurais de configuração semelhante. Mas foi o trabalho de Schapire que colocou os sistemas de *ensemble* no centro das pesquisas de aprendizagem de máquina, quando ele provou que um classificador forte no sentido provavelmente aproximadamente correto (*PAC – Probably Approximately Correct*) pode ser gerado pela combinação de classificadores fracos através de um procedimento que ele o chamou de *boosting* (Schapire (1990)).

Esses trabalhos foram fundamentais para que as pesquisas em *ensemble learning* tenham expandido rapidamente e foram categorizados como sendo de seleção ou fusão, caso os classificadores sejam selecionados ou se suas saídas são combinadas, respectivamente. Lembrando que esses dois métodos, seleção e fusão, podem ser combinados.

Nos trabalhos de fusão a preocupação se restringe a duas etapas: i) procedimento de geração dos classificadores individuais ou ensemble. ii) estratégia empregada para combinação dos classificadores. Assim diversos métodos de geração de ensemble surgiram como: bagging (Breiman 1996), boosting (Schapire 1990), AdaBoost (Freud and Schapire 1996), Random Subspace Method (Ho 1998). Esses métodos de geração têm como objetivo construir um bom *ensemble,* onde os classificadores sejam precisos e diversos quanto possíveis. Para a segunda etapa, a combinação pode ser aplicada às classes de saída ou a um conjunto de valores contínuos para cada classe específica resultante da classificação de um especialista individual do ensemble (Kittler 1998). Neste último caso, as saídas são frequentemente normalizadas no intervalo [0, 1], e esses valores são interpretados como um suporte dado pelo classificador para cada classe. Tal interpretação permite a aplicação de várias regras algébricas de combinação (votação majoritária, máximo, mínimo, soma, produto e outras combinações de probabilidades a posteriori) (Kittler 1998, Kuncheva 2002, Roli 2002), integral fuzzy (Cho 1995), a fusão baseada em Dempster-Shafer (Rogova 1994) e mais recentemente, modelos de decisão (*decision templates*) (Kuncheva 2001).

Porém, vários estudos têm investigado bastante o uso da abordagem de seleção de classificador como uma opção ao invés da fusão, pois esta exige a independência de classificadores. Como resultado, vários métodos de seleção dinâmica (DCS e DES) têm sido propostos na literatura como visto a seguir.

Em 1997, Woods propôs em [2] uma abordagem para seleção dinâmica de classificador, o DCS-LA (Dynamic Classifier Selection by Local Accuracy), baseado no conceito “Local Accuracy Estimates”, em que a precisão local de cada classificador em relação a um padrão de teste é estimada e o classificador com a maior precisão local é selecionado. Woods ainda propôs dois métodos para estimar a precisão local (local accuracy): OLA (Overall Local Accuracy), e o LCA (Local Class Accuracy). Em que a OLA é uma simples precisão de classificação ao redor de uma amostra de teste para cada classificador, e a LCA é similar ao OLA, só que a precisão local é com respeito as classes de saída.

Já Giacinto propôs em [3] um framework para a seleção dinâmica de classificadores usando os métodos de seleção A Priori e A Posteriori e um algoritmo baseado em seu framework proposto. Esses métodos de seleção são baseados na estimativa da probabilidade da classificação correta em uma região local do espaço de características em torno do padrão de teste desconhecido. Portanto, assim como os métodos de Woods baseados na estimativa da precisão local, o classificador que tiver a maior probabilidade de classificação correta sobre a amostra de teste é selecionado, em que o método A Priori não usa a informação da classe atribuída pelo classificador ao padrão de teste, e A Posteriori usa a informação da classe.

Uma maneira de obter o máximo de desempenho de um EoC é através do conceito Oracle, em que temos um seletor ótimo ou ideal para um conjunto de classificadores, ou seja, o Oracle classifica corretamente um dado padrão se existe algum classificador do EoC que o classifica corretamente, assim o Oracle é o limite superior do desempenho de um EoC.

A partir do conceito Oracle, Ko [4] propôs a abordagem KNORA (K-nearest-oracles). O KNORA é similar aos conceitos OLA, LCA, A Priori e A Prosteriori em considerar a vizinhança do padrão de teste, mas difere destes pelo uso direto de sua propriedade tendo as amostras de treinamento na região com a qual encontra o melhor ensemble para uma determinada amostra. Para uma amostra de teste, o KNORA encontra os seus k vizinhos mais próximos no conjunto de validação, descobre quais os classificadores que classificam corretamente esses vizinhos no conjunto de validação e usa-os como um ensemble para classificar o padrão de teste fornecido. Ko ainda propôs quatro diferentes métodos para a abordagem KNORA: KNORA-E (KNORA-Eliminate), KNORA-U (KNORA-Union), e suas versões ponderadas, KNORA-E-W e KNORA-U-W. A partir desta abordagem, algumas variantes KNORA surgiram posteriormente.

Outro trabalho notável foi o de Dos Santos, que em [5] propôs uma estratégia dinâmica de OCS (Overproduce-And-Choose Strategy), que combina otimização e seleção dinâmica em uma fase de seleção de dois níveis. Em que no nível de otimização é realizado uma busca multi-objetiva através de algoritmos genéticos, e no nível de seleção dinâmica é baseada em medidas de confiança, ao invés da precisão do classificador nas regiões de competência como nos métodos de seleção tradicionais.

Por fim, existem outras abordagens que podem ser usadas para melhorar o desempenho de EoC, como as estratégias que visam melhorar a geração dos classificadores bases [6] ou aprimorar as regiões de competência através da eliminação de ruídos por meio de filtros de seleção [7].

* 1. **Geração de Ensemble**

A intuição diz que em um bom *ensemble* os classificadores devem ser precisos o quanto possível e diversos o quanto possível. Sabe-se que quanto mais precisos são os classificadores menores são as taxas de erros que eles apresentam para novas entradas. No entanto, a diversidade não é um conceito preciso. Definir este conceito e como isto influencia o desempenho não é trivial.

Um ponto de consenso é que, quando os classificadores cometem erros estatisticamente independentes, a combinação tem o potencial de aumentar o desempenho do sistema. Se o conhecimento sobre a diversidade de um ensemble for boa, isso torna possível escolher melhor o método de fusão. Por exemplo, sabe-se que simples fusões podem ser usadas para classificadores com uma simples complementação de padrões, mas que fusões complexas são necessárias para classificadores com um modelo de dependência complexa.

A ligação como às medidas de diversidade estão relacionadas com a precisão do ensemble é um assunto de investigação ainda. No entanto, estudos experimentais observaram os resultados esperados: quanto maior diversidade maior o desempenho. Eles também mostraram que as propriedades de um ensemble que são desejáveis (alta precisão e diversidade) para obter uma combinação de sucesso não são comuns na prática.

Um exemplo didático é mostrado na *fig 1* onde 3 classificadores (com 0.6 de precisão individual) tem que classificar 10 objetos desconhecidos. A figura mostra quatro casos: um caso estatisticamente independente, um caso com classificadores idênticos e dois casos de dependência de classificadores. Por este exemplo é claro que a independência contribui para o aumento do desempenho. Quando a dependência é assumida, podemos ter classificação correta ou incorreta da dependência, dando resultados muito diferentes. Avaliar a diversidade de um classificador também não é uma tarefa trivial, já que o conceito em si não é claro, mas existem medidas *pairwise* que podem ser usadas.

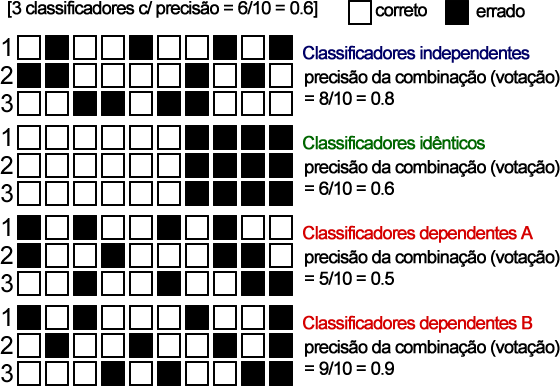


Fig 1. Exemplo de diversidade de ensemble. Quatro casos indicando o possível desempenho, mostrando que classificadores independentes aumenta potencialmente o desempenho individual, e dois casos de dependência de classificador com resultados diferentes.

Portanto, as diversas abordagens de geração de classificadores visam criar bons classificadores (precisos e diversos) para o ensemble a fim de terem um melhor desempenho em sua combinação. Alguns dos métodos mais populares são vistos a seguir.

* + 1. **Bagging**

O *Bagging* (Breiman 1996), um acrônimo para *Bootstrap AGGregatING*, é um dos primeiros, mais intuitivos, e talvez um dos mais simples algoritmos baseado em ensemble, com um bom desempenho *(ref. scholarpedia)*. O *bagging* é baseado na ideia que amostras de *bootstrap* do conjunto de treinamento original irão apresentar uma pequena mudança com respeito ao conjunto de treinamento original, mas uma diferença suficiente para produzir classificadores diversos (Moacir). Assim, para o *bootstrap* os diferentes subconjuntos de dados de treinamento são sorteados randomicamente - com reposição - do conjunto de dados de treinamento inteiro. Depois cada subconjunto de dados de treinamento é usado para treinar um classificador base diferente. E por fim os classificadores individuais são então combinados pela média ou votação majoritária de suas decisões. O pseudocódigo do algoritmo pode ser visto na figura 2 abaixo:

|  |
| --- |
| **Algoritmo 1** Bagging |
| **Entrada:** tamanho *L* do ensemble; conjunto de treinamento *S* de tamanho *N*.  **Inicialização**   1. , ensemble   **Fase de Treinamento**   1. **para** i = 1 até *L* **faça** 2. Sorteio de amostras de *S*, com reposição. *bootstrap* 3. Constrói o classificador usando 4. Adiciona o classificador ao ensemble   **Fase de Classificação**   1. **para** cada novo padrão **faça** 2. **se** se as saídas são contínuas **então** 3. Combine as saídas dos classificados de pela média 4. **senão se** as saídas são os rótulos das classes **então** 5. Combine as saídas dos classificados de pela votação majoritária |

Apesar do bagging ser um bom algoritmo, como à diversidade dos classificadores é obtida usando várias “réplicas” *bootstrap* do conjunto de treinamento, o bagging só é eficaz quando uma pequena mudança no conjunto possa causar uma mudança significativa no modelo. Assim para fazer uso das variações do conjunto de treinamento, o classificador base deve ser instável ou não linear, ou seja, as pequenas alterações no conjunto de treinamento devem levar a grandes mudanças na saída do classificador. Caso contrário, o conjunto resultante será uma coleção de classificadores quase idênticos, portanto, pouco provável de melhorar o desempenho de um classificador único. Exemplos de classificadores instáveis são redes neurais e árvores de decisão, enquanto k-vizinho mais próximo é um exemplo de um classificador estável.

Como pode ser observado anteriormente o algoritmo do *bagging* é uma solução *ensemble* completa, ou seja, o algoritmo executa desde a geração do ensemble de classificadores até a classificação. Porém, o objetivo de relacionar o *bagging* ou outros algoritmos como sendo geração de ensemble é simplesmente pela preocupação apenas com a geração dos classificadores bases, no caso do *bagging* a fase de treinamento.

* + 1. **Boosting**

Similar ao *bagging*, o *boosting* (Schapire 1990)também cria um ensemble de classificadores por subconjuntos de amostras dos dados de treinamento e combina a saída por votação majoritária. No entanto, no *boosting*, a amostragem é estrategicamente orientada para fornecer os dados de treinamento mais informativos para cada classificador consecutivamente. Em essência, cada iteração do *boosting* cria três classificadores fracos: o primeiro classificador C1 é treinado com um subconjunto aleatório dos dados de treinamento. O subconjunto de dados de treinamento para o segundo classificador C2 é escolhido como o subconjunto mais informativo, dado C1. Mais especificamente, C2 é treinado com os dados de treinamento somente metade dos quais é corretamente classificados por C1, e a outra metade que é classificada erroneamente. O terceiro classificador C3 é treinado com as instâncias em que C1 e C2 discordarem. Os três classificadores são combinados através da votação majoritária de três vias. O detalhe do pseudocódigo é mostrado na figura abaixo a seguir:

|  |
| --- |
| **Algoritmo 2** Boosting |
| **Entrada:** conjunto de treinamento *S* de tamanho *N* com as classes , ; algoritmo de aprendizagem fraco **WeakLearn**.  **Fase de Treinamento**   1. Sorteio de amostras sem reposição de *S*. 2. Chama **WeakLearn** e treina usando . 3. Cria o subconjunto com os mais informativos, dado , tal que metade de é classificada corretamente por e a outra metade são erroneamente classificada.    1. Jogue uma moeda. Se cara, selecione amostras de *S*, e apresente a até a primeira instância erroneamente classificada. Adicione essa instância à .    2. Se coroa, selecione amostras de *S*, e apresente a até a primeira instância corretamente classificada. Adicione essa instância à .    3. Continue lançando moedas até que nenhum padrão possa ser mais adicionado a . 4. Treina o segundo classificador usando . 5. Cria selecionando aquelas instâncias pras quais e discordam. 6. Treina o terceiro classificador usando .   **Fase de Classificação** – dado um padrão de teste **x**   1. Classifique **x** por e . Se eles concordarem na classe, esta classe é a classificação final. 2. Se eles discordarem, escolha a classe predita por como a classificação final. |

Schapire mostrou que o erro desse algoritmo tem um limite superior: se o algoritmo A usado para criar os classificadores , , tem um erro de (calculado em *S*), então o erro do ensemble é limitado acima por . Note que para . Ou seja, enquanto o algoritmo original A pode fazer pelo menos melhor do que adivinhar aletoriamente, então o ensemble *boosting* que combina os três classificadores gerados por A de três distribuições de *S* descrito acima, sempre terá melhor desempenho que A. Portanto, um classificador forte é gerado a partir de três classificadores fracos. Um classificador forte no sentido provavelmente aproximadamente correto (*PAC*) pode ser criado por aplicações recursivas do *boosting.* Uma limitação específica do *boosting* é que ele se aplica somente a problemas de classificação binária. Esta limitação é superada com o algoritmo AdaBoost (Adaptive Boosting).

* + 1. **Random Subspace**

Uma maneira de melhorar a diversidade em um ensemble é treinar os classificadores individuais com diferentes subconjuntos do espaço de características. A seleção de características visa uma computação mais eficiente e uma maior precisão do conjunto. O *RSM* (*Random Subspace Method*) (Ho 1998) é um método de seleção de subconjuntos de características aleatória para a construção de ensemble. É similar ao *bagging* mas ao invés de fazer a amostragem de instâncias, ele faz uma amostragem de características sem repetição, uma vez que seria inútil incluir uma característica mais de uma vez. O *RSM* seleciona randomicamente um número arbitrário de subespaços de características do espaço original, e constrói cada classificador em cada subespaço. Essa randomização deve criar classificadores que são complementares e assim a combinação pode ser feita por simples regras de fusão.

|  |
| --- |
| **Algoritmo 3** Random Subspace Method |
| **Entrada:** tamanho *L* do ensemble; conjunto de treinamento *S* de tamanho *N*, onde o número de características é *D*; escolher número de características para treinar cada classificador individual, onde , para .  **Inicialização**   1. , ensemble   **Fase de Treinamento**   1. **para** i = 1 até *L* **faça** 2. Sorteio de características de *D*, sem reposição. 3. Constrói o classificador usando 4. Adiciona o classificador ao ensemble   **Fase de Classificação**   1. **para** cada novo padrão **faça** 2. **se** se as saídas são contínuas **então** 3. Combine as saídas dos classificados de pela média 4. **senão se** as saídas são os rótulos das classes **então** 5. Combine as saídas dos classificados de pela votação majoritária |

Experimentalmente evidências mostram que *RSM* trabalha bem com espaços de características com grandes conjuntos de características e redundâncias de características. Isso evita a “maldição” da dimensionalidade. Os conceitos de *RSM* podem ser relacionados à teoria da discriminação estocástica de Kleinberg.

* 1. **Seleção DCS e DES**

Nesta seção serão descritos os principais métodos de seleção dinâmica DCS e DES. Estas estratégias de *DS* tem o propósito de selecionar o classificador (DCS) ou o conjunto de classificadores (DES) que melhor classifica a região de competência, dado uma amostra de teste. A seguir serão explicados primeiros os métodos de *DCS* baseado no DCS-LA:OLA e LCA. Em seguida os métodos *DES* baseados no *KNORA*: KNORA-Eliminate e KNORA-Union.

* + 1. **DCS-LA**

O DCS-LA (Dynamic Classifier Selection by Accuracy) proposto por (Woods 1997) é uma abordagem para seleção dinâmica de classificador, baseado no conceito da estimativa da precisão local “*Local Accuracy Estimates”*. Portanto, a ideia é estimar a precisão de cada classificador em regiões locais do espaço de características ao redor da amostra de teste desconhecida, e então usar a decisão do classificador mais preciso localmente. Na implementação de Woods, as “regiões locais” são definidas em termos do K-vizinhos mais próximos (K-NN) no conjunto de dados de treinamento. Woods propôs dois métodos para estimar a precisão local (*Local Accuracy)* que são: precisão local total (*OLA – Overrall Local Accuracy*) e a precisão local da classe (*LCA – Local Class Accuracy*). O pseudocódigo da abordagem *DCS-LA* é descrita abaixo:

|  |
| --- |
| **Algoritmo 4** DCS-LA |
| **Entrada:** ensemble *E* de tamanho *L*; parâmetro *K* de vizinhos mais próximos (recomendado K=10); conjunto de dados de treinamento *T*.   1. **para** cada novo padrão de teste **faça** 2. Defina a região local ao redor da amostra de teste () como sendo o conjunto dos K-vizinhos mais próximos (K-NN) a partir do conjunto de treinamento *T*. 3. **para** cada classificador individual do ensemble *E* **faça** 4. Calcule a estimativa de precisão local de para a região local . 5. Utilize a decisão do classificador para classe da amostra que tiver a maior estimativa da precisão local . |

Os dois métodos, OLA e LCA, para calcular a estimativa de precisão local serão descritos nas próximas seções a seguir.

* + - 1. **OLA**

O *OLA* (*Overrall local accuracy*) usa a abordagem de *rank* de classificador (*Classifier Rank*), que seleciona o classificador que classifica corretamente mais vizinhos da amostra de teste na região local. O classificador selecionado é dito ter o maior “rank”. Um exemplo prático do *OLA*, pode ser visto a seguir:

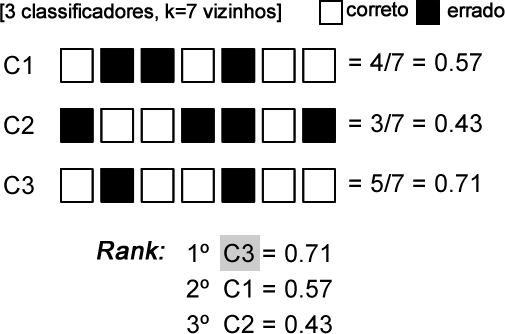


Figura 2. Exemplo da abordagem OLA, onde três classificadores (C1, C2 e C3) classificam os 7 vizinhos do padrão de teste. O OLA faz o rank dos classificadores com maior percentagem de acerto dos vizinhos, nesse exemplo o seleciona o classificador C3.

* + - 1. **LCA**

Este método é similar ao método OLA, a única diferença é que a precisão local é estimada em respeito a classe de saída. Considerando um classificador que atribui uma classe a uma amostra de teste. Com isso é possível determinar a percentagem das amostras de treinamento atribuídas à classe por este classificador que tem sido classificada corretamente. A seguir um exemplo do LCA:

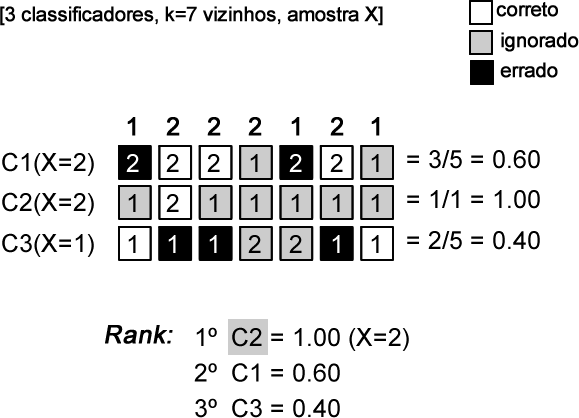


Figura 3. Exemplo da abordagem LCA. Os três classificadores C1, C2 e C3 classificam o padrão de teste X e suas precisões locais são calculadas baseadas na classe de saída atribuída a X. Conforme o exemplo, o classificador C3 tem a maior precisão com relação a classe, pois acertou 100% dos vizinhos que classificou como sendo da mesma classe atribuída a X.

* + 1. **KNORA**

Todos os métodos de seleção dinâmica descritos acima são projetados para encontrar o classificador com maior possibilidade de ser correto para uma amostra em uma vizinhança pré-definida. No entanto a abordagem do KNORA é outra: ao invés de encontrar o classificador mais adequado, é selecionado o conjunto mais adequado para cada amostra utilizando o conceito *Oracle*.

O conceito do KNORA (K-nearest-oracles) (Ko 2008) é similar aos conceitos de OLA e LCA na consideração da vizinhança de padrões de teste, mas pode ser distinguido dos outros pelo uso direto de sua propriedade de ter amostras de treinamento na região com o qual quer encontrar o melhor conjunto para uma determinada amostra. Para qualquer dado de teste, KNORA simplesmente encontra os *K* vizinhos mais próximos no conjunto de validação, descobre quais os classificadores classificam corretamente esses vizinhos no conjunto de validação e usa-os como ensemble para classificar o padrão fornecido naquele conjunto de teste.

Ko ainda propôs quatro diferentes métodos para a abordagem KNORA: KNORA-E (KNORA-Eliminate), KNORA-U (KNORA-Union), e suas versões ponderadas, KNORA-E-W e KNORA-U-W. As descrições detalhadas desses métodos serão vistas a seguir.

* + - 1. **KNORA-Eliminate**

O KNORA-E seleciona o subconjunto de classificadores por meio da eliminação de classificadores utilizando a abordagem KNORA. Portanto, dado K vizinhos , de um padrão de teste , e supondo que o conjunto de classificadores , classificam corretamente todos os seus K-vizinhos mais próximos, então cada classificador pertencente a este conjunto de classificadores corretos deve apresentar um voto sobre a amostra . No caso onde nenhum classificador pode classificar corretamente todos os K-vizinhos mais próximos do padrão de teste, então simplesmente é diminuído o valor de K até pelo menos um classificador classificar corretamente os seus vizinhos. Um exemplo prático pode ser visto a seguir:

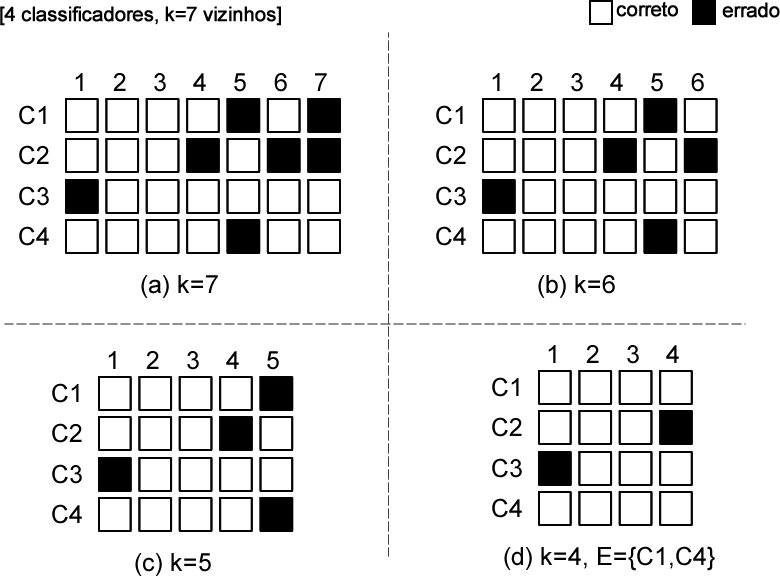


Figura 4. Exemplo da execução do KNORA-Eliminate para um ploblema com 4 classificadores no ensemble inicial e a região de competência com k=7 vizinhos. Depois de executar os passos mostrados em (a), (b), (c) e (d), o KNORA-E seleciona os classificadores C1 e C4. Dessa forma esses classificadores apresentarão cada um, um voto sobre a amostra X.

* + - 1. **KNORA-Union**

O KNORA-U utiliza a abordagem KNORA selecionar os classificadores que classificam corretamente cada padrão vizinho. Dessa forma, dado K vizinhos , de um padrão de teste , e supondo que o -vizinho mais próximo tem sido corretamente classificado por um conjunto de classificadores , , então cada classificador pertencentes ao conjunto de classificadores correto devem apresentar um vote sobre a amostra . Note que, uma vez que todos os K-vizinhos próximos são considerados, um classificador pode ter mais de um voto se este classificar corretamente mais de um vizinho. Quanto mais vizinhos o classificador classificar corretamente, mais votos este classificador terá para um padrão de teste. Um exemplo do KNORA-U pode ser visto na figura a seguir:

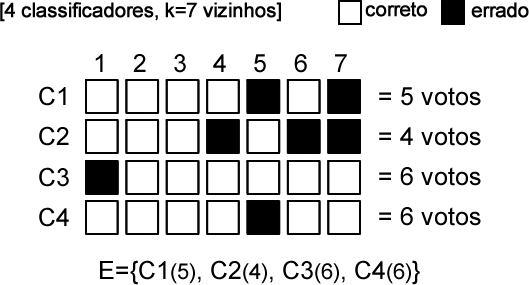


Figura 5. Exemplo do KNORA-Union com 4 classificadores e 7 vizinhos. Nesse exemplo, todos os classificadores foram selecionados com seus respectivos votos sobre a amostra X.

* + - 1. **KNORA-Eliminate-W**

Esse método é similar ao KNORA-Eliminate, a diferença é que cada voto do classificador é ponderado pela distância Euclidiana entre o padrão vizinho e o padrão de teste .

* + - 1. **KNORA-Union-W**

Esse método é similar ao KNORA-Union, a diferença é que cada voto é ponderado pela distância Euclidiana entre o padrão vizinho e o padrão de teste .

* 1. **Combinação de Decisões**

Depois obter um ensemble, por um método de geração de ensemble ou por um método de seleção dinâmica de ensemble (*DES*). A decisão final pode ser obtida por várias regras de combinação (fusão). As possíveis maneiras de combinar as saídas dos *L* classificadores de um ensemble dependem de qual informação é obtida dos membros individuais. Alguma destas regras de combinação opera somente na classe de saída, enquanto outros precisam das saídas contínuas que podem ser interpretadas como suporte dado pelo classificador para cada uma das classes. Segundo Xu os modelos de informação produzida pelos membros de um ensemble podem ser classificados em três tipos:

* Abstrato (*The Abstract Level*):
  + Cada classificador produz um rótulo da classe de saída . Portanto, para qualquer amostra de teste ser classificada, as saídas classificadores definem um vetor . No nível abstrato, não há nenhuma informação que indica o grau certeza da classificação atribuída. Por definição, qualquer classificador pode ser capaz de produzir uma classe de saída para .
* Rank (*The Rank Level*):
  + A saída de cada classificador é um subconjunto de , com as alternativas em ordem de possibilidade de serem corretamente classificadas. Sendo assim é uma lista de classes ranqueadas para cada padrão de entrada.
* Medição (*The Measurement Level*):
  + A saída de cada classificador produz um vetor -dimensional de valores contínuos. O valor contínuo representa a estimativa da probabilidade a posteriori da classe ou o valor de confiança relacionado a classe que representa o suporte para a possível hipótese de classificação do padrão de teste . Sem perca de generalidade, pode ser assumido que o vetor de saída contem valores .
    1. **Votação Majoritária**

A votação majoritária é uma das estratégias mais antigas para fazer a decisão e trabalha no nível de abstração. Ela é utilizada por diversos métodos de criação de ensemble vistos anteriormente e é tido como um combinador ideal. Junto com as regras de média e produto, que serão vistos na próxima seção, a votação majoritária é um dos métodos mais usados.

Assuma que as saídas dos classificadores são dadas por um vetor binário -dimensional , , onde se o classificador classificou como sendo da classe , e caso o contrário. Portanto, a decisão do ensemble é feita para a classe se este recebe a maior votação:

Vários estudos tem mostrado que a votação majoritária é um dos métodos mais utilizados devido a sua propriedade característica: sob a condição de que as saídas dos classificadores são independentes, pode ser demonstrado que a combinação por votação majoritária vai sempre levar a uma melhoria de desempenho. Se há um total de *L* classificadores para um problema de duas classes, a decisão do ensemble estará correta se ao menos classificadores escolher a classe correta.

Agora, assuma que cada classificador tem uma probabilidade de tomar uma decisão correta. Então, a probabilidade do ensemble tomar uma decisão correta tem uma distribuição binomial, especificamente, a probabilidade de escolher classificadores corretos de *L* é:

Então,

quando se e

quando se .

Note que a exigência de é necessária e suficiente para um problema de duas classes, enquanto que é suficiente, mas não necessário para problemas multiclasse.

Uma variante da votação majoritária é a votação majoritária ponderada. Os votos são multiplicados por pesos que é frequentemente obtido pela estimativa da precisão dos classificadores no conjunto de validação. Uma possível estimativa é:

Onde é a precisão do *i­*-ésimo classificador.

* + 1. **Combinadores Algébricos**

Os graus de suporte dado por um padrão de entrada pode ser interpretado em diferentes maneiras, às duas mais comuns são: o valor de *confiança* em uma sugestão de classificação e a *estimativa das probabilidades a posteriori* para as classes.

Dado que o padrão seja um vetor de características e seja o conjunto das classes do problema. Cada classificador no ensemble têm saídas de graus de suporte. Assumindo que todas as saídas são valores no intervalo , então trabalha no nível de medição. Portanto, denota-se o suporte que o classificador dá a hipótese que é da classe . Quanto maior o suporte, maior a chance de ser da classe . As saídas dos *L* classificadores para um dado padrão pode ser organizada em uma matriz de perfil de decisão (*decision profile*) como:

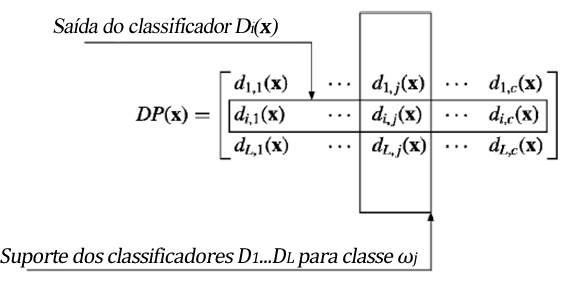


Figura 6. Matriz .

Os combinadores algébricos usa a matriz para encontrar o suporte geral de cada classe para entrada com o maior valor de suporte. Isso é feito através de , o suporte de graus geral para , obtido após aplicar uma função de combinação de expressões algébricas aos suportes individuais da classe dado pelo ensemble. Sendo assim, a decisão final pode ser obtida por . As funções de combinação podem computar o suporte geral da classe de diferentes maneiras como visto a seguir:

* Média: calcula a média dos suportes para cada classe, e a decisão final é dada pela classe com maior média.
* Soma: realiza a soma dos suportes para cada classe, e a decisão final é dada pela classe com maior soma. Essa regra é equivalente à média.
* Produto: multiplica os suportes para cada classe, e atribui a decisão final para a classe que tiver o maior produto.
* Máximo: encontra o suporte máximo para cada classe, e atribui a decisão final para a classe com o maior suporte máximo.
* Mínimo: encontra o suporte mínimo para cada classe, e atribui a decisão final para a classe com o maior suporte mínimo.
* Mediana: encontra a mediana dos suportes para cada classe, e atribui a decisão final para a classe com o maior mediana dos suportes.

A figura a seguir mostra um exemplo de combinação dos classificadores utilizando as funções algébricas através da matriz .

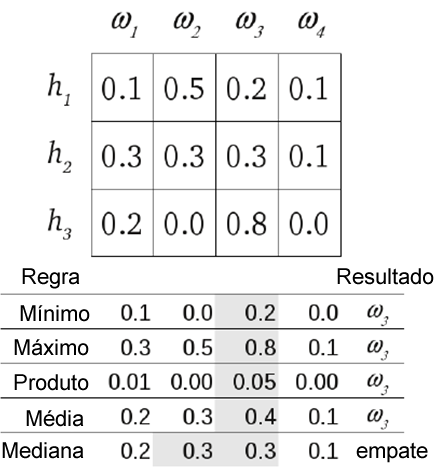


Figura 6. Problema utilizando um ensemble de três classificadores e várias regras de combinação algébrica.