Miguel de Souza Corti

Métodos de Active Learning para algoritmos de classificação

RELATÓRIO DE PROJETO FINAL

DEPARTAMENTO DE INFORMÁTICA
Programa de graduação em Engenharia de Computação

Rio de Janeiro
Julho de 2021

Pontifícia Universidade Católica
do Rio de Janeiro

Miguel de Souza Corti

Métodos de Active Learning para algoritmos de classificação

Relatório de Projeto Final

Relatório de Projeto Final, apresentado ao programa Engenharia de Computaçãoda PUC-Rio como requisito parcial para a obtenção do titulo de Engenheiro de Computação.

Orientador: Prof. Eduardo Sany Laber

Rio de Janeiro
Julho de 2021

Agradecimentos

À minha família, sem eles não teria começado essa jornada, constantemente me apoiando de todas as formas possíveis.

À minha companheira que sempre me apoia e inspira em tudo que faço.

Aos meus amigos, que sempre me proporcionam momentos memoráveis e tornam momentos difíceis muito mais fáceis.

Ao meu orientador Eduardo Laber, por todo o conhecimento e orientações precisas, que me guiaram e motivaram.

À PUC-Rio e todo seu corpo docente, que tanto me ensinaram e auxiliaram, principalmente durante este período tão complexo.

Resumo

de Souza Corti, Miguel; Sany Laber, Eduardo. **Métodos de Active Learning para algoritmos de classificação**. Rio de Janeiro, 2021. 36p. Relatório de Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Com a vontade de automatizar diversas tarefas utilizando Aprendizado de Máquina e ensinar computadores a classificar certas informações, se tornou crescente a necessidade de especialistas capazes de rotular amostras corretamente auxiliando na criação de bases para treino dos algoritmos. Como este processo pode facilmente se tornar caro, lento e/ou repetitivo, tem se buscado métodos para melhorá-lo. Aprendizado Ativo é um destes métodos, que, calculando um valor para designar o quão informativo é um exemplo, pergunta interativamente para o especialista classificar apenas os exemplos mais significativos para melhorar o modelo. Então, este trabalho estuda a aplicação de diversas estratégias de Aprendizado Ativo, utilizando variações de conjuntos de dados e modelos para entender quando, como e por que é interessante utilizar este método.

Palavras-chave

Inteligência Artificial. Aprendizado de Máquina. Classificação. Aprendizado Supervisionado. Aprendizado Ativo.

Abstract

de Souza Corti, Miguel; Sany Laber, Eduardo (Advisor). . Rio de Janeiro, 2021. 36p. Capstone Project Report – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

With the will to automate several tasks using Machine Learning and mostly teach computers how to classify certain informations, the need of specialists able to classify a sample correctly and help create training datasets for the algorithms to learn has grown in the last years. As this process can easily become costly, slow or repetitive, there has been a lot of research on methods that help improve it. Active Learning is one of those methods, that, by assigning a value based on how informative an entry is, interectivaly chooses the most significant entries to query the specialist. Knowing that, this project studies the application of several Active Learning methods, with varying datasets and models to understand when, how and why its useful to apply this method.

Keywords

Artificial Intelligence. Machine Learning. Classification. Supervised Learning. Active Learning.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

Seres vivos, em especial humanos, possuem raciocínio lógico que os auxiliam constantemente a resolver problemas e tomar decisões, sejam elas simples, como escolher tomar água quando está com sede, ou mais complexas, como descobrir uma vacina para um novo vírus. Este raciocínio avançado que encontramos em várias espécies é um dos grandes motivadores para pesquisadores que buscam alcançar o mesmo processamento utilizando computadores.

Porém, mesmo na era da tecnologia, o raciocínio dos computadores ainda possui muitas limitações quando comparado a capacidade do cérebro humano. Alguns problemas que para o humano são muito simples de resolver, mesmo recebendo dados desestruturados, para um computador pode demandar um algoritmo complexo com uma boa estruturação e definição dos dados de entrada e saída. Para muitas áreas e empresas é de muito interesse que estes processos sejam automatizados, podendo ser resolvidos sem depender de um ou mais humanos, reduzindo os custos e possivelmente aumentando a produtividade.

Com a grande explosão no volume de dados gerados e armazenados nas últimas décadas, um dos métodos que tem sido cada vez mais adotado dentro da Inteligência Artificial é o Aprendizado de Máquina (do inglês, Machine Learning, ou ML), que procura ensinar o computador a realizar certas tarefas treinando-o com dados previamente obtidos onde o resultado é conhecido para depois aplicar em dados novos. Alguns exemplos clássicos incluem reconhecimento facial, detecção de spams na caixa de e-mails, diagnosticar pacientes, reconhecimento de letras em imagens, entre outros.

Para executar um projeto utilizando Machine Learning, geralmente são necessários os seguintes passos:

1. Estudar o conjunto de dados desejado;
2. Escolher um modelo;
3. Treinar o modelo utilizando os dados conhecidos;
4. Utilizar o modelo treinado para fazer previsões ou tomar decisões em casos nunca antes apresentados.

A principal diferença deste método para a programação tradicional, é possibilitar chegar em resultados bastante precisos sem precisar programar explicitamente as regras que resolvem o problema. Como o próprio nome já diz, a máquina aprende a chegar na solução baseada em um modelo e um conjunto de treino determinado pelo desenvolvedor. Em muitas situações pode até ser bem complexo descobrir o caminho que levou o programa a um certo resultado.

Dentro destas diversas aplicações, uma grande área de problemas é a designada Classificação, que dado um conjunto de atributos e classes, tem como objetivo rotular observações como classes pré-determinadas. No exemplo da caixa de e-mail, é razoável definir duas classes, os que são spam e os que não são, e para treinar o algoritmo utilizar dados das caixas de entrada de usuários que já classificaram seus e-mails, como assunto, remetente e conteúdo, além da própria classificação que o usuário definiu. Após treinado, será possível apresentar novos e-mails ao algoritmo para descobrir se estes são ou não spam.

É evidente que quanto maior e mais diversa for a sua base de treino, melhor será o aprendizado do seu programa, pois ele terá mais dados para basear suas decisões. Portanto, para alcançar resultados satisfatórios é de extrema importância que o treinamento passe por muitos exemplos e que estes exemplos sejam bem descritivos do que se deseja ensinar.

Com isso, surge um problema. Para gerar essa base e treinar um bom algoritmo é necessário um extenso trabalho prévio, realizado manualmente por humanos que conheçam o assunto, de classificar os exemplos corretamente. Em algumas situações, esse trabalho já foi feito naturalmente e a base já está disponível para uso, como seria o caso de um provedor de email que desejasse implementar um detector de spam automático (como a maioria já faz atualmente). Porém, muitas vezes a base está parcial ou totalmente sem classificação e é necessário o auxílio de um especialista, muitas vezes também chamado de oráculo ou professor. Estes especialistas apesar de serem perfeitamente capazes de produzir uma boa base, acabam criando um processo caro, lento e repetitivo.

Estes processos, onde é necessário o especialista, são também conhecidos como Aprendizado Supervisionado (do inglês, Supervised Learning) e englobam uma série de algoritmos que auxiliam tanto na geração da base, como no treinamento do modelo.

Este trabalho tem como foco estudar um caso especial de Supervised Learning, o Active Learning, ou Aprendizado Ativo em português, também abreviado como AL. Dado uma base com observações não rotuladas (ou pouco rotuladas), o objetivo do AL é selecionar os exemplos mais informativos

sobre o problema para questionar interativamente ao oráculo qual a classe destes exemplos escolhidos. Assim, o trabalho do oráculo se torna muito mais efetivo para o treinamento do modelo e também possibilita alcançar resultados satisfatórios sem necessidade de classificar a base inteira

Ao longo do trabalho é explicado mais a fundo as principais formas como o AL é implementado, aplicando-o em diversos casos, variando os conjuntos de dados, os modelos de treinamento e as estratégias utilizadas para escolher os melhores candidatos a serem apresentados ao oráculo.

Na seção 4 são averiguados diversos resultados, comprovando que o uso de Active Learning pode resolver os problemas apresentados acima. Sendo capaz de produzir bons resultados com um número menor de instâncias rotuladas, auxiliando em situações onde o processo de rotulação da base é caro ou demorado.

2
Trabalhos Relacionados

Muitos estudos sobre Active Learning já foram desenvolvidos, um dos principais, frequentemente citado em artigos posteriores, é *Active Learning Literature Survey*, um relatório técnico escrito por Burr Settles em 2008 (2). Este trabalho compila diversas informações e resultados encontrados na área nos anos anteriores e cita trabalhos recentes, alguns desenvolvidos pelo próprio autor, que também são apresentados aqui nas seções a seguir.

Outro trabalho relacionado, porém com teor mais prático, é a biblioteca *modAL* e sua documentação (8), desenvolvida para facilitar a implementação de Active Learning na linguagem Python. Apesar de não ter sida utilizada diretamente neste trabalho, muitos quesitos da implementação feita para obter os resultados da seção 4 se assemelham, principalmente pelo uso da mesma linguagem, Python e outras bibliotecas comumente usadas para Machine Learning, como *Scikit-Learn* (18). Esta biblioteca também implementa e explica alguns métodos que não foram desenvolvidos neste trabalho, como *Expected Error Reduction* e o cenário de *Stream-base Sampling*.

Dado a eficácia e estrutura por trás de *Support Vector Machines* (SVMs), existem muitos estudos que exploram Active Learning focado neste modelo, alguns deles são (10), (11), (12), (13) e (14).

3
Active Learning

O processo de AL lida constantemente com 3 principais conjuntos, que vamos designar de:

- *Labelled*, o conjunto de todas as instâncias que a classe já é conhecida;
- *Pool*, o conjunto de todas as instâncias que a classe é desconhecida;
- *Selected*, o conjunto de instâncias para serem classificados.

Então, o fluxo esperado é ter inicialmente um *Pool* com um grande ou crescente número de instâncias e o *Labelled* com poucas ou nenhuma instância. Determinado um método para selecionar (ou gerar) instâncias para inquerir ao oráculo, cada iteração do algoritmo utiliza a *Pool* para montar o *Selected*, que após classificados são removidos de *Selected* e adicionados ao *Labelled*.

Na seção 2.1 são discutidos três principais fluxos para executar AL e depois, na seção 2.2, é apresentado algumas estratégias selecionadas para fazer a escolha das instâncias mais informativas ao modelo que entrarão no conjunto *Selected* a cada iteração.

3.1
Cenários

Existem algumas configurações possíveis para executar AL, as três seguintes serão apresentadas: *Membership Query Synthesis*, *Stream-Based Selective Sampling* e *Pool-Based Sampling*.

3.1.1
Membership Query Synthesis

Os primeiros trabalhos em AL utilizavam modelos de inquéritos (que se refere a consulta ao oráculo). Existem diversos formatos para realizar esse inquérito, tanto na forma como a pergunta é exibida como na resposta esperada do oráculo, como é possível ver em (1). Porém apenas o de pertencimento (*membership*) será debatido abaixo.

Diferente dos outros modelos de inquérito onde o oráculo precisa responder com diversas informações sobre a instância escolhida pelo algoritmo, no

Inquérito de Pertencimento só é necessário informar a qual classe a instância pertence.

Neste formato, cada consulta é feita utilizando instâncias novas, que foram geradas baseadas na amostra real. Esse método pode ser muito útil quando sua amostra inicial é pequena e sabe-se como simular novas instâncias.

O problema desse método é que a instância sintetizada pode não fazer muito sentido para o ser humano que está respondendo as consultas, gerando em certos casos um exemplo que não é classificável em nenhuma das classes disponíveis (3). Esse problema é resolvido pelas configurações apresentadas a seguir, porém ainda existem alguns trabalhos que utilizam esse método como (4) e (5).

3.1.2 *Stream-Based Selective Sampling*

Existem muitos problemas no mundo real em que a base de dados não está completamente disponível no momento do desenvolvimento, geralmente em sistemas que recebem dados de usuários ou de sensores ao longo do tempo. Nestes casos se torna muito caro manter oráculos disponíveis classificando todas as entradas ao longo do tempo.

Para auxiliar nesse problema é possível adotar uma abordagem de *Selective Sampling* (6), pois a cada nova entrada no sistema o algoritmo pesa, através de alguma métrica de informação pré-definida, a relevância da instância nova para o modelo. Assim, um novo exemplo só será questionado ao oráculo caso seja verificado que sua adição melhorará a performance do modelo baseado em algum parâmetro de corte.

Esse modelo é aplicado em alguns trabalhos como (7).

3.1.3 *Pool-Based Sampling*

A *Pool-Based Sampling* se assemelha a *Stream-Based Selective Sampling* no sentido que ambas utilizam uma métrica de informação para escolher que instâncias exibir ao oráculo. Porém na *Pool-Based Sampling*, como o próprio nome já indica, as instâncias não são recebidas uma a uma ao longo do tempo, neste caso a base já está completa desde o começo do desenvolvimento.

Portanto, a cada iteração o algoritmo escolhe de forma gulosa n instâncias mais informativas do *Pool* para questionar o oráculo.

As aplicações para este formato também são diversas e é facilmente encontrado, como é possível ver em (12), (16), (11), entre outros.

Neste trabalho é analisado diversas configurações utilizando este cenário na seção 4.

3.2 Estratégias de Seleção

Conhecido o cenário em que será feito o processo de obtenção dos dados e consulta ao oráculo, é preciso escolher alguma estratégia que avalie as instâncias no nível de informação que agregará ao modelo. Essa métrica é responsável por dizer quais instâncias serão escolhidas para serem classificadas pelo oráculo em cada iteração do processo.

Existem diversos métodos para obter esse resultado, são apresentados abaixo alguns dos mais comuns. Estas estratégias também são posteriormente utilizadas na seção 4, para realizar diversos experimentos e análise dos resultados.

3.2.1 Amostragem por Incerteza (*Uncertainty Sampling*)

Uma estratégia muito comum é *Uncertainty Sampling* (16). A ideia é quantificar as instâncias não classificadas utilizando uma fórmula que diga o quão incerto o modelo está sobre a previsão de sua classe. Afinal, excluindo casos de *outliers*, instâncias que o modelo consegue prever sem qualquer dúvida provavelmente já estão corretamente classificadas pelos parâmetros atuais e não trará muita melhoria na classificação de novas instâncias.

Essa estratégia funciona bem com modelos que utilizam probabilidade para classificar, pois é fácil tirar medições fazendo cálculos em cima das probabilidades previstas pelo modelo treinado.

Por exemplo, um critério simples e fácil de utilizar em classificadores binários é procurar as instâncias que a probabilidade de ser de uma das classes é mais próximo de $0,5$.

Para classificadores multi-classe existem algumas alternativas que podem performar melhor dependendo da aplicação. Um critério bem trivial é o *Least Confident*:

$$x^* = \underset{x}{\operatorname{argmax}}\, 1 - P_\theta(\hat{y}|x) \tag{3-1}$$

Sendo $\hat{y}$ a classe de maior probabilidade prevista, esse critério fornecerá os exemplos que o modelo possui menos certeza de sua classificação.

Porém esse critério só observa a probabilidade de uma das classes, a mais provável. Um critério que engloba mais sobre o resultado das previsões é o *Smallest Margin* (ou SMU) (17):

$$x^* = \underset{x}{\operatorname{argmin}}\, P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x) \tag{3-2}$$

Sendo $\hat{y}_1$ a classe mais provável e $\hat{y}_2$ a segunda classe mais provável, a intuição é achar os exemplos mais ambíguos, ou seja, os exemplos que o modelos está mais incerto sobre duas classes.

Outro método similar ao SMU é o LMU, ou *Largest Margin*:

$$x^* = \underset{x}{\operatorname{argmin}}\, P_\theta(\hat{y}_{max}|x) - P_\theta(\hat{y}_{min}|x) \tag{3-3}$$

Com $\hat{y}_{max}$ a classe mais provável de uma instância x e $\hat{y}_{min}$ a classe menos provável. A ideia é que se a distância entre probabilidade de $\hat{y}_{max}$ e $\hat{y}_{min}$ for grande o classificador tem mais certeza sobre a classificação feita. Então o algoritmo de AL procurará os exemplos que tiverem menor LMU.

Apesar do SMU e LMU considerarem duas classes para tomar uma decisão, para problemas com muitas classes eles podem acabar desconsiderando muita informação relevante do resultado e apresentar resultados piores. Para resolver isso existe uma métrica mais geral, utilizando Entropia (19):

$$x^* = \underset{x}{\operatorname{argmax}} \Big(-\sum_i P_\theta(\hat{y}_i|x) \log P_\theta(\hat{y}_i|x) \Big) \tag{3-4}$$

Com $\hat{y}_i$ variando por todas as possíveis classes de saída, quanto maior for a entropia mais incerto o classificador está sobre a instância. Este método funciona muito bem para cenários de multi-classe com classificadores probabilísticos.

3.2.2 Consulta por Comitê (*Query By Committee*)

Apesar de *Uncertainty Sampling* ser um dos métodos mais utilizados e fáceis de implementar, existem algumas alternativas como *Query By Committee*. Neste método, o objetivo é manter um comitê de modelos, que a cada iteração do AL, são treinados com o conjunto *Labelled* e usados para prever as instâncias do *Pool*. Então, são escolhidos os exemplos que mais causaram discordância nas previsões.

Para utilizar esse método é primeiro necessário montar um comitê de modelos que representem hipóteses diferentes. Depois é preciso estabelecer uma métrica para calcular a discordância entre os resultados do comitê.

Uma possível abordagem é utilizar um cálculo baseado em entropia, similar ao visto na equação 3-4. Sendo C o tamanho do comitê, e $V(y_i|x)$ a quantidade de votos que uma classe recebeu dado uma instância x, o método chamado de *Vote Entropy* é definido por (20):

$$x^* = \underset{x}{\operatorname{argmax}} \left(-\sum_i \frac{V(y_i|x)}{C} \log \frac{V(y_i|x)}{C} \right) \tag{3-5}$$

Vale notar que o QBC pode encontrar problemas de performance, pois é necessário reajustar todos os modelos do comitê a cada iteração. Caso a *Pool* seja muito grande (comum em *Pool-Based Sampling*) ou o problema seja complexo (com muitos atributos e/ou classes), fazer o ajuste desses modelos será demorado.

3.2.3 *Density-Weighted*

As duas estratégias citadas acima (QBC e Amostragem por Incerteza) se utilizam de um conceito central para escolher a(s) nova(s) instância(s), sempre buscando instâncias que tragam a maior mudança para o modelo. Portanto, ao selecionar uma nova instância as métricas exibidas acima (entropia, smallest margin, etc.) olham para as amostras da *Pool* de forma individual.

O problema é que esse método favorece a escolha de *outliers*, dado que estes de fato trazem grande mudança no modelo. Para resolver esse problema, existem algumas estratégias que tiram o foco de avaliar a instância individualmente e passam a considerar todo o espaço definido pelas amostras para tomar uma decisão sobre escolher ou não uma dada instância da *Pool*. Algumas dessas estratégias são *Expected Error Reduction* e *Variance Reduction*, que focam em buscar instâncias que ao adicionadas no conjunto de treino reduzem o erro de generalização. Porém vamos focar em uma estratégia de *Density-Weighting*, que pesa as instâncias utilizando o resto da *Pool* através de alguma regra.

Uma dessas regras é a *information density*, proposta por *Settles and Craven* (9):

$$x^* = \underset{x}{\operatorname{argmax}}\, \phi(x) \times \left(\frac{1}{P} \sum_p sim(x, x_p) \right)^{\beta} \tag{3-6}$$

Onde P é o tamanho da *Pool*, $\phi(x)$ é uma estratégia base, como amostragem por incerteza ou QBC e $sim(x, x_p)$ mede a similaridade entre duas amostras. Assim, o segundo termo da equação servirá como peso para a estratégia base, usando a similaridade média de uma instância com relação a todo o resto da *Pool*. O parâmetro β serve apenas para possibilitar reduzir ou amentar o impacto do peso na decisão.

Agora é necessário definir uma função para medir a similaridade, para isso vamos primeiro definir um vetor de atributos para uma dada amostra i:

$$\vec{x_i} = \left[f_1(x_i), ..., f_j(x_i) \right] \tag{3-7}$$

Onde $f_j(x_i)$ é o valor do atributo j para a instância i. Com esse vetor definido podemos usar similaridade por cosseno:

$$sim_{cos}(x, x_p) = \frac{\vec{x} \cdot \vec{x_p}}{||\vec{x}|| \times ||\vec{x_p}||} \tag{3-8}$$

Um detalhe importante dessa estratégia é que há um custo quadrático no tamanho de *Pool* para calcular a similaridade, porém como os valores dos atributos são fixos, é possível pré-computar as densidades, efetuando os cálculos uma única vez e apenas consultando-as posteriormente. Mantendo o tempo para escolha de uma nova instância praticamente indiferente do tempo ao usar apenas a estratégia $\phi(x)$ sem pesar usando densidade.

3.2.4 Amostragem com SVM

Support Vector Machines são modelos utilizados para tarefas de classificação e regressões. Seu uso se tornou bastante comum principalmente por ter uma base teórica simples apresentando bons resultados em diversas áreas (13).

Este modelo não utiliza probabilidades para efetuar classificações, portanto os métodos de Amostragem por Incerteza apresentados na seção 3.2.1 não podem ser utilizados. Para isso vamos apresentar um novo método levando em consideração a estrutura do SVM.

De forma simples, os SVMs procuram definir um hiperplano que separe a base de treino, dividindo o espaço em dois. Em problemas de classificação binária, a escolha desse hiperplano é de tal forma que separe os pontos das duas classes. É evidente que existem diversos hiperplanos que satisfazem essa condição, portanto para escolher o hiperplano ótimo uma regra razoável é maximizar a margem entre os pontos das duas classes.

Para este hiperplano é definido um vetor normal w, perpendicular ao plano, com sentido voltado para o espaço da classe positiva. Após treinar o modelo, é possível prever uma nova instância utilizando a distância ao hiperplano, caso a distância seja positiva (ou seja, está no espaço "apontado"pela normal) será classificado positivamente, caso seja negativa será rotulada com a classe negativa.

Portanto, a principal informação utilizada para rotular uma instância são as distâncias ao hiperplano. Uma estratégia que utiliza desse dado é a *Simple Margin* (11):

$$x^* = \underset{x}{\operatorname{argmin}} \, |f(x)| \tag{3-9}$$

Sendo $f(x)$ a função de decisão, que informa as distâncias. A ideia dessa estratégia é buscar as instâncias com menor margem, ou seja, que mais

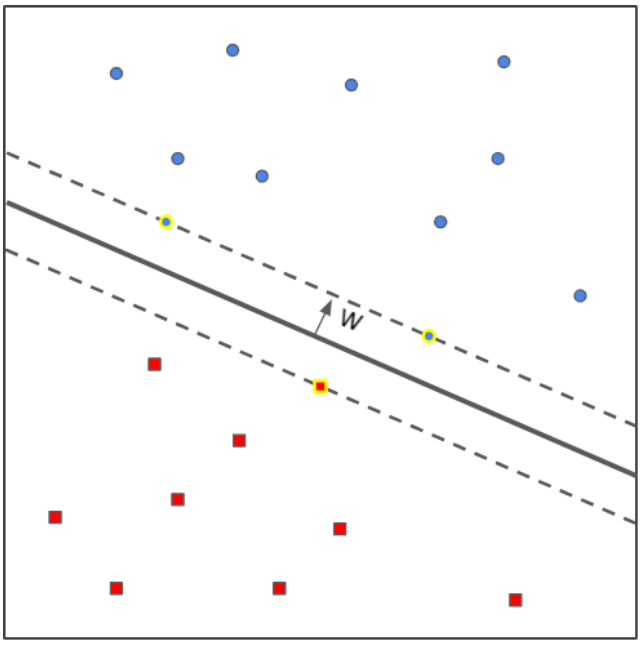


Figura 3.1: Um exemplo de ajuste de um SVM para um problema binário em 2 dimensões. A linha cinza sólida representa o hiperplano e as linhas tracejadas representam a margem. As instâncias com borda amarela representam os *Support Vector's*. Está também representando o vetor normal ao hiperplano w.

se aproximam ao hiperplano. Estas instâncias são as que apresentam maior incerteza, visto que estão próximas do espaço da outra classe delimitado pelo hiperplano.

Devido a sua estrutura, os SVM's resolvem apenas problemas de classificação binários, porém é possível estendê-lo para multi-classe treinando vários SVM's. Pode-se treinar um SVM para cada classe, que saibam distinguir sua classe do resto e ao prever rotula-se utilizando o SVM que apresentou maior certeza sobre sua classificação (ou seja, maior distância do hiperplano), esse método é chamado de *One-vs-Rest (OvR)*. Outra possibilidade, chamada de *One-vs-One (OvO)*, é treinar um classificador para cada par de classes e rotular com a classe que ganhou mais "disputas".

Para ambos os métodos, a estratégia de *Simple Margin* precisa ser adaptada. Para estratégias utilizando as distâncias ao hiperplano, pode-se usar *Nearest to All Hyper-Planes (NAH)* (15):

$$x^* = \underset{x}{\operatorname{argmin}} \sum_{h \in H} |d(x, h)| \tag{3-10}$$

Com H o conjunto de todos os hiperplanos e $d(x, h)$ a distância de uma instância x ao hiperplano h.

Outra estratégia, baseada no número de votos (disputas ganhas) é a *Least*

Votes Margin (LVM) (15):

$$x^* = \underset{x}{\operatorname{argmin}}\, V(\hat{y}_1|x) - V(\hat{y}_2|x) \tag{3-11}$$

Sendo $V(\hat{y}_i|x)$ a quantidade de disputas que uma classe $\hat{y}_i$ para uma dada instância x, $\hat{y}_1$ a classe que ganhou mais votos e $\hat{y}_2$ a segunda classe que ganhou mais votos. Esta estratégia se assemelha a ideia da *Smallest Margin*, buscando as instâncias que o modelo está mais em dúvida entre as duas classes mais votadas.

4
Experimentações

Para validar o uso de *Active Learning* e verificar sua usabilidade, foram implementados e testados diversos cenários. Variando *datasets*, classificadores e estratégias de AL.

Todas as implementações utilizam a abordagem de *Pool-based Sampling*, apresentada na seção 3.1.3, com as variadas estratégias de seleção, já apresentadas na seção 3.2. Cada combinação de *dataset*, classificador e estratégia é referenciado abaixo como *experimentos*.

Para evidenciar as diferenças entre os classificadores, foram escolhidos 4 modelos que representam conceitos diferentes:

- Random Forest
- k-NN
- Logistic Regression
- SVM

Para o Random Forest, k-NN e Logistic Regression, que são modelos de base probabilística, as estratégias experimentadas foram *Smallest Margin* (3-2), *Entropy* (3-4), *Query-by-committee* usando *Vote Entropy* (3-5) e *Information Density* (3-6) usando *Smallest Margin* como medida de incerteza.

Para o SVM, que se baseia na distância ao hiperplano para classificar, são apresentadas as seguintes estratégias: *Nearest to All Hyper-Planes* (3-10), *Least Votes Margin* (3-11), *Query-by-committee* usando *Vote Entropy* e *Information Density* usando *Nearest to All Hyper-Planes* como medida de incerteza.

Em todos os experimentos, o comitê usado para a votar na estratégia *Query-by-committee* é fixo. Composto pelos 3 classificadores: k-NN, Decision Tree e SVM. Cada um representando uma hipótese diferente para o problema.

Os classificadores e estratégias acima foram aplicados em nove *datasets*, comumente usados em problemas de classificação. Os *datasets* variam em contexto e propriedades, com tamanhos de amostra, classes e atributos diferentes. Resolvendo tanto problemas binários como multi-classe. São eles:

- Iris
  - Tamanho: 150 instâncias

  - Classes: 3
  - Descrição: Contém dados sobre três espécies diferentes da flor Iris
  - Objetivo: Identificar a espécie de uma flor Iris

- Digits

  - Tamanho: 1797 instâncias
  - Atributos: 64
  - Classes: 10
  - Descrição: Contém imagens, de tamanho 8x8 pixels, com dígitos de 0-9 escritos a mão
  - Objetivo: Classificar corretamente qual o número escrito na imagem

- Wine

  - Tamanho: 178 instâncias
  - Atributos: 13
  - Classes: 3
  - Descrição: Contém informações sobre vinhos feitos utilizando diferentes tipos de uvas de uma mesma região da Itália
  - Objetivo: Classificar de qual cultivar foi feito o vinho

- Breast Cancer

  - Tamanho: 569 instâncias
  - Atributos: 30
  - Classes: 2
  - Descrição: Contém dados sobre um paciente
  - Objetivo: Classificar se o paciente possuí ou não câncer de mama

- Olivetti Faces

  - Tamanho: 400 instâncias
  - Atributos: 4096
  - Classes: 40
  - Descrição: Contém 10 fotos diferentes dos rostos de 40 indivíduos tirados pelo *ATT Laboratories Cambridge*, de tamanho 64x64 pixels
  - Objetivo: Classificar corretamente de quem pertence o rosto da imagem

- Mice Protein

  - Tamanho: 1080 instâncias
  - Atributos: 77
  - Classes: 8

  - Descrição: Contém dados sobre os níveis de 77 proteínas detectadas no cérebro de ratos. Cada rato está incluso em uma entre 8 classes, que descrevem o tipo de tratamento pelo qual o rato passou
  - Objetivo: Classificar corretamente qual o tipo de tratamento o rato recebeu

- Diabetes

  - Tamanho: 768 instâncias
  - Atributos: 8
  - Classes: 2
  - Descrição: Contém algumas informações sobre diversos indivíduos, como idade, peso, pressão sanguínea, etc.
  - Objetivo: Identificar se o indivíduo possuí ou não diabetes

- Glass

  - Tamanho: 214 instâncias
  - Atributos: 10
  - Classes: 6
  - Descrição: Dos 10 atributos, 9 descrevem a proporção de certos elementos em um pedaço de vidro, o 10º indica a intensidade de refração
  - Objetivo: Distinguir os tipos de vidro, identificando o seu principal uso baseado na sua composição

- Vehicle

  - Tamanho: 846 instâncias
  - Atributos: 19
  - Classes: 4
  - Descrição: Cada atributo contém informações sobre o formato e geometria de um veículo
  - Objetivo: Classificar o veículo entre um dos 4 grupos: opel, saab, ônibus ou van

Em todos os experimentos realizados, os *datasets* foram previamente divididos em um conjunto para testes e um para treino (posteriormente dividido entre *Labelled* e *Pool*), a uma proporção de 30% para testes e 70% para treinamento. A cada iteração do algoritmo, os modelos são treinados utilizando o conjunto *Labelled* atual e o conjunto de testes é então usado para prever e medir a acurácia do classificador.

Além das estratégias discutidas na seção 3.2, sempre é feito um experimento usando uma estratégia aleatória para escolher as novas instâncias a

Dataset	Random	Smallest Margin	Entropy	QBC Entropy	Information Density
iris	88,51	89,29	89,28	89,25	**89,36**
digits	71,11	**78,41**	75,35	76,61	**78,41**
wine	93,98	**95,88**	95,73	94,54	95,86
breast cancer	92,06	**94,91**	**94,91**	93,47	94,85
olivetti faces	38,96	**45,36**	42,12	40,65	**45,36**
mice protein	58,04	**64,46**	61,19	60,27	**64,46**
diabetes	71,33	72,12	72,12	**72,27**	71,98
vehicle	61,74	62,79	60,49	59,72	**62,80**
glass	64,36	**66,47**	65,60	64,65	66,37

Tabela 4.1: Valores da área abaixo da curva de acurácia para Random Forest. Está marcado em negrito e sublinhado as estratégia que obtiveram melhor e pior resultado, respectivamente.

serem rotuladas. Assim é possível verificar se tanto o esforço computacional como o esforço de desenvolvimento das estratégias de AL fazem sentido para o *dataset* e classificador especificado.

Em todos os experimentos os modelos já são inicialmente treinados utilizando 10 instâncias escolhidas de forma aleatória. A cada iteração apenas uma instância é selecionada para ser rotulada, portanto o eixo horizontal dos gráficos vistos a seguir pode ser interpretado como número de iterações ou a quantidade de instâncias rotuladas pelo algoritmo de AL.

Cada experimento foi executado 20 vezes, fixando os parâmetros e a *seed* dos classificadores (caso tenha aleatoriedade), variando apenas o embaralhamento dos conjuntos de teste e treino. Os valores apresentados a seguir são uma média dos resultados encontrados nessas 20 execuções.

É possível visualizar os resultados em formato gráfico, representando a evolução da acurácia (média) a cada iteração do processo de aprendizado e em tabelas informando o valor da área abaixo da curva de acurácia para cada combinação de *dataset*, classificador e estratégia.

4.1 Random Forest

Para os experimentos com Random Forest, ss estratégias que apresentam melhor resultado no geral são *Smallest Margin* e *Information Density*, como visto na tabela 4.1. Em alguns casos é verificável que não há diferença significativa entre estas estratégias, isto em geral ocorre pois alguns *datasets* utilizados possuem poucas amostras ruidosas e pouca presença de *outliers*, um problema que o *Information Density* procura resolver ao selecionar novas instâncias.

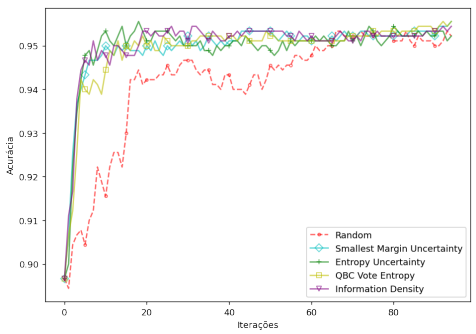
O único *dataset* que fugiu dessa análise foi o *Diabetes*, que apresentou

melhor acurácia no *Query-by-Committee*.

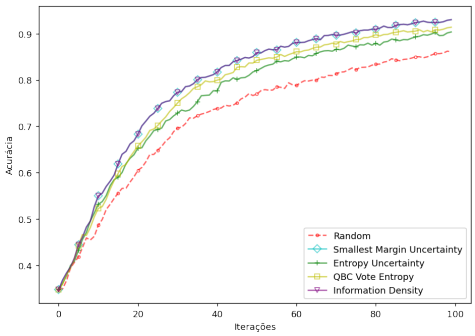
Em todos os experimentos foi possível verificar que o uso de Active Learning compensa o esforço, visto que as curvas de acurácia para as estratégias selecionadas aparecem na maior parte dos casos por cima da curva *Random*, como verificável nas Figuras 4.1(a-i).

É importante notar que conforme a quantidade de instâncias rotuladas aumenta, a acurácia das estratégias (inclusive a aleatória) tende a se igualar, como é possível ver principalmente nas figuras 4.1(a), 4.1(c) e 4.1(g). Isto é evidente, pois a partir de certo ponto a base *Labelled* se torna praticamente a mesma.

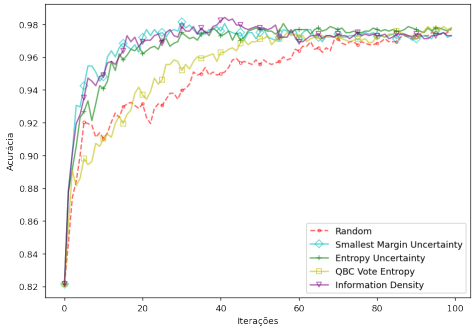
Neste momento vale relembrar a motivação inicial de Active Learning: alcançar bons resultados com menos esforço de classificação manual. Com isto em mente, entende-se que as estratégias experimentadas satisfazem essa motivação, pois é possível alcançar acurácias satisfatórias rotulando muito menos instâncias. Utilizando o exemplo da figura 4.1(a), com menos de 15% da base rotulada já se alcança a acurácia que, de forma aleatória, só seria alcançada após 50% da base estar rotulada. O mesmo é verificável nos outros *datasets*, porém em proporções diferentes.



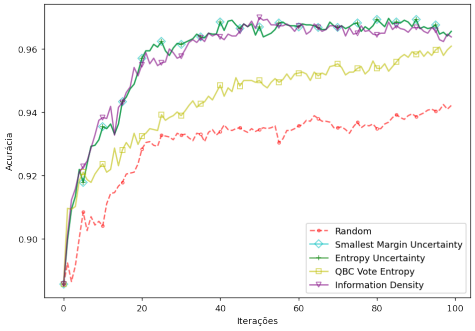
4.1(a): Iris



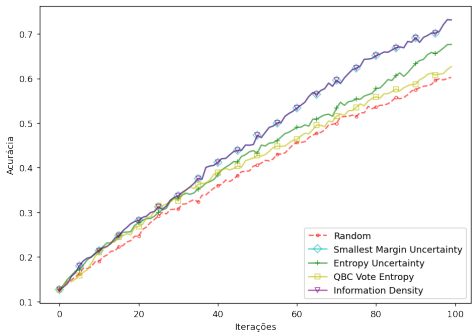
4.1(b): Digits



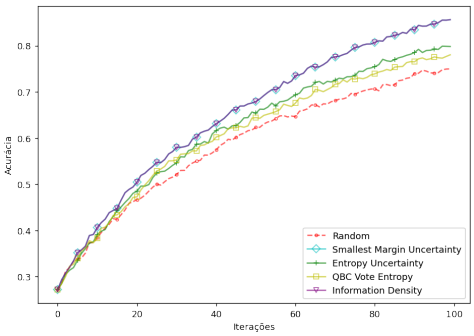
4.1(c): Wine



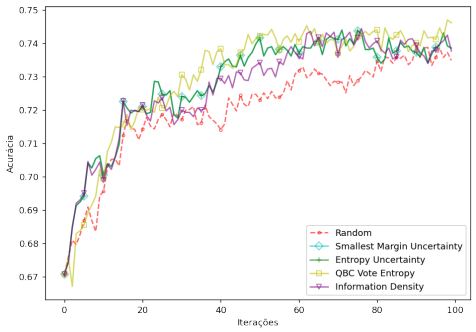
4.1(d): Breast Cancer



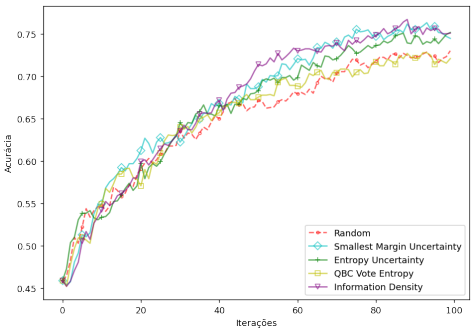
4.1(e): Olivetti Faces



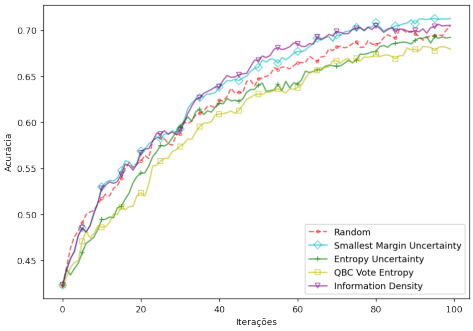
4.1(f): Mice Protein



4.1(g): Diabetes



4.1(h): Glass



4.1(i): Vehicle

Figura 4.1: Acurácia média por iterações para cada dataset e estratégia utilizando o Random Forest.

4.2 k-NN

Os mesmos experimentos executados para o Random Forest foram feitos para o k-NN, fixando o número de vizinhos consultados em 5 e utilizando distância euclidiana.

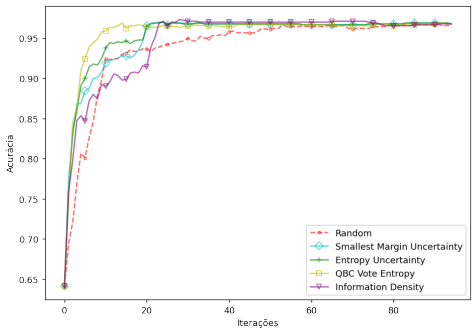
Uma diferença notável entre os métodos são os valores de acurácia menores para o k-NN em comparação ao Random Forest. Essa diferença pode ser justificada por uma falta de pré-processamento específico para cada modelo, o que não é o foco desse estudo.

Além disso, em quesitos de Active Learning, é possível observar pela

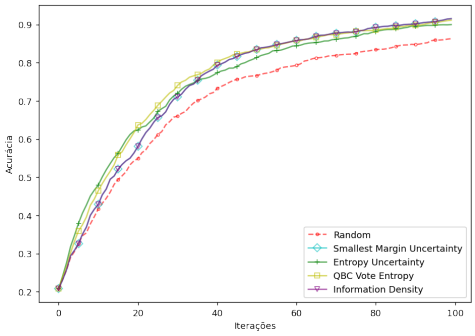
Dataset	Random	Smallest Margin	Entropy	QBC Entropy	Information Density
iris	88,29	89,55	89,79	**90,03**	89,11
digits	68,98	73,75	73,92	**74,75**	73,75
wine	67,77	**68,99**	68,47	68,81	68,68
breast cancer	89,69	**91,79**	**91,79**	91,39	91,38
olivetti faces	23,97	24,58	**25,74**	25,13	24,58
mice protein	41,02	43,07	**43,93**	42,99	43,07
diabetes	66,23	65,86	65,86	**66,77**	64,29
vehicle	45,06	45,19	45,53	**45,84**	45,37
glass	56,41	56,26	**57,12**	56,02	56,09

Tabela 4.2: Valores da área abaixo da curva de acurácia para k-NN. Está marcado em negrito e sublinhado as estratégia que obtiveram melhor e pior resultado, respectivamente.

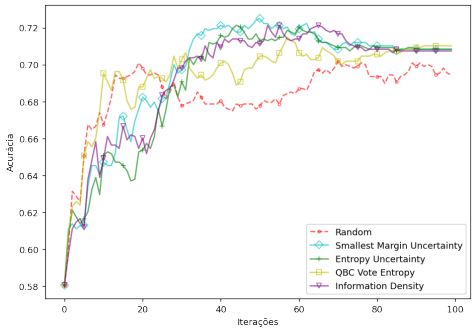
tabela 4.2 e figuras 4.2(a-i) que as estratégias com melhores resultados não são mais as mesmas. Enquanto no Random Forest, *Smallest Margin* e *Information Density* foram as principais estratégias, no k-NN *Entropy* e *QBC Entropy* apresentam resultados melhores, apenas o *dataset Wine* foge desse padrão, tendo melhor acurácia no *Smallest Margin*.



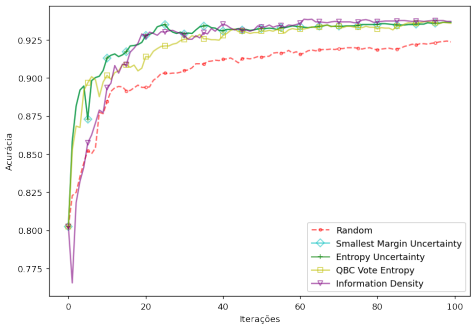
4.2(a): Iris



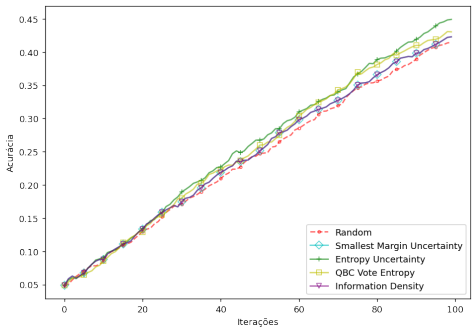
4.2(b): Digits



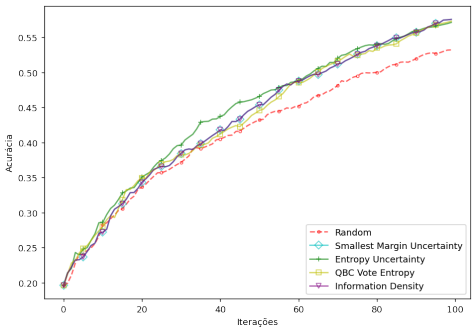
4.2(c): Wine



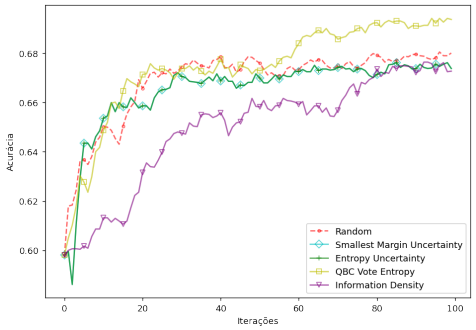
4.2(d): Breast Cancer



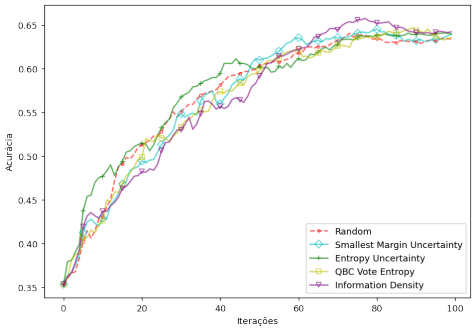
4.2(e): Olivetti Faces



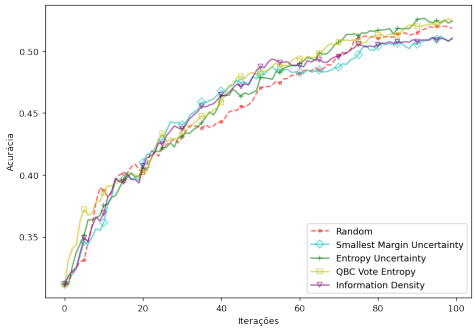
4.2(f): Mice Protein



4.2(g): Diabetes



4.2(h): Glass



4.2(i): Vehicle

Figura 4.2: Acurácia média por iterações para cada dataset e estratégia utilizando o k-NN.

4.3 Logistic Regression

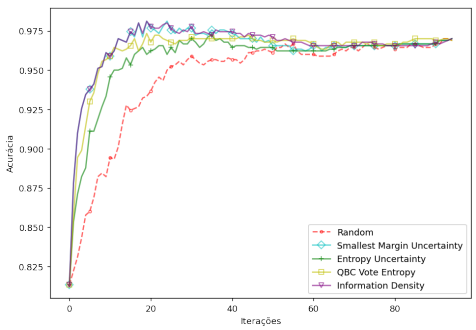
Novamente, os mesmos experimentos foram executados, porém agora utilizando Logistic Regression para efetuar a classificação. O *solver* utilizado em todos os experimentos foi o *lbfgs*. Para melhorar os resultados e o tempo de convergência do modelo, os datasets foram normalizados utilizando o método Min-Max entre 0 e 1. Em especial o *Olivetti Faces* passou por uma redução de dimensionalidade utilizando o método de PCA, reduzindo para 90 componentes. Isto foi necessário pois Logistic Regression não consegue convergir para bases com muitos atributos e poucas entradas.

Dataset	Random	Smallest Margin	Entropy	QBC Entropy	Information Density
iris	88,85	90,70	89,98	90,50	**90,77**
digits	75,97	**82,77**	77,07	80,74	82,28
wine	92,42	95,13	**95,38**	93,40	94,88
breast cancer	91,36	**95,37**	**95,37**	93,82	95,34
olivetti faces	31,98	38,45	**45,38**	31,92	35,50
mice protein	52,01	**61,10**	55,62	57,58	60,90
diabetes	67,35	70,86	70,86	69,27	**70,98**
vehicle	46,03	49,71	44,75	46,69	**50,11**
glass	43,48	**47,33**	43,86	46,07	46,80

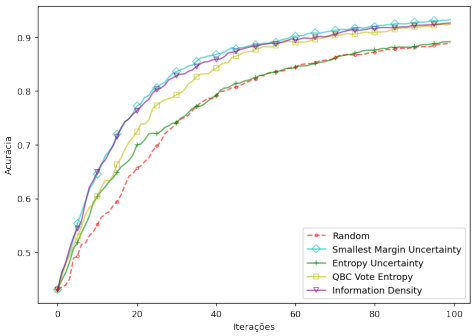
Tabela 4.3: Valores da área abaixo da curva de acurácia para Logistic Regression. Está marcado em negrito e sublinhado as estratégia que obtiveram melhor e pior resultado, respectivamente.

É possível ver nas figuras 4.3 e pela tabela 4.3 que o Active Learner se comportou como esperado para a maioria dos casos e estratégias. Apresentando constantemente resultados melhores que a seleção aleatória.

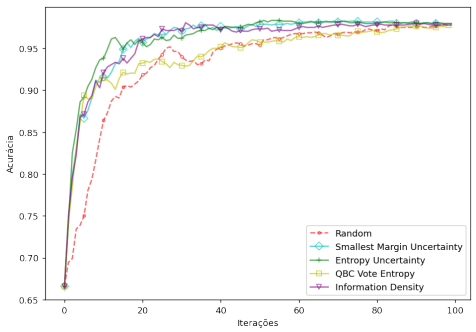
Similar ao Random Forest, *Smallest Margin* e *Entropy* apresentaram os melhores resultados em 6 dos 9 datasets. Os outros 3 apresentaram acurácias maiores no *Information Density*, e apresentou pouca melhora quando comparado ao *Smallest Margin*. Portanto os métodos de incerteza, em principal o *Smallest Margin* apresentou melhores resultados com Logistic Regression.



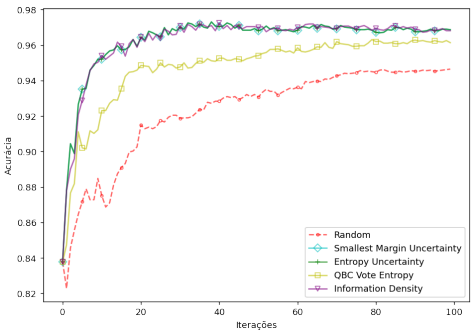
4.3(a): Iris



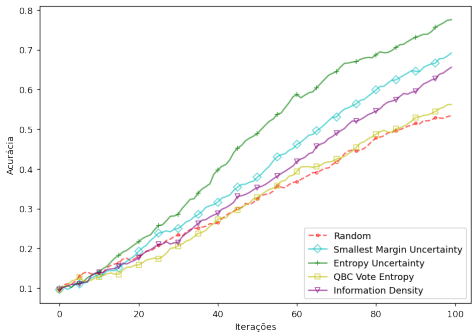
4.3(b): Digits



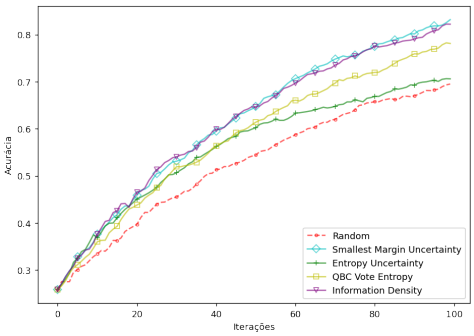
4.3(c): Wine



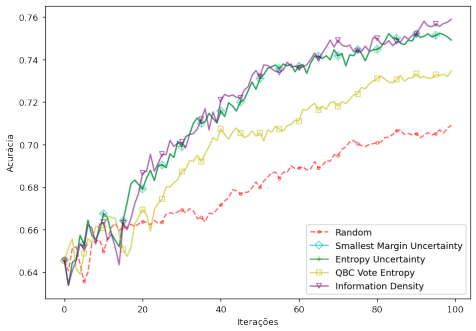
4.3(d): Breast Cancer



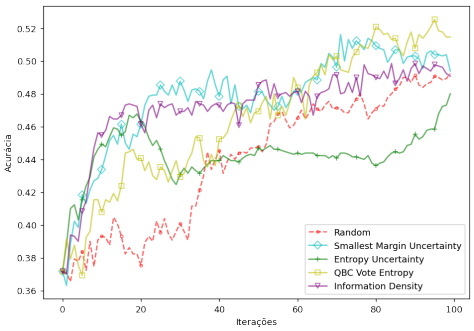
4.3(e): Olivetti Faces



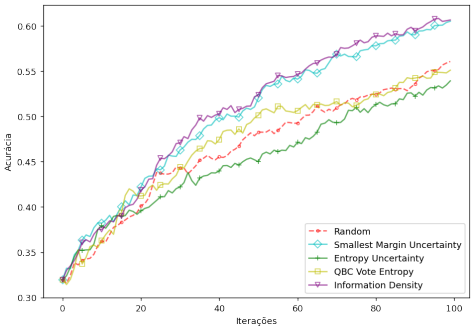
4.3(f): Mice Protein



4.3(g): Diabetes



4.3(h): Glass



4.3(i): Vehicle

Figura 4.3: Acurácia média por iterações para cada dataset e estratégia utilizando Logistic Regression.

4.4 SVM

Diferente dos modelos anteriores, para o SVM foi necessário implementar novas estratégias, dado que este não utiliza probabilidades para efetuar classificação, informação essencial para os métodos de *Uncertainty Sampling* da seção 3.2.1. Portanto, as estratégias de *Smallest Margin* e *Entropy* foram substituídas por *Nearest to All Hyper-Planes* e *Least Votes Margin*. *Information Density* e *Query-by-Committee* foram mantidas, pois não dependem da incerteza medida por probabilidade.

Para os *datasets* multi-classe foi utilizada a estratégia One-Vs-One,

Dataset	Random	NAH	LVM	QBC Entropy	Information Density
iris	86,41	**89,27**	88,34	89,19	80,74
digits	68,15	67,35	**76,78**	75,10	46,34
wine	94,98	**96,32**	95,68	95,87	88,52
breast cancer	88,12	89,38	-	**89,50**	89,37
olivetti faces	21,92	**28,91**	26,61	23,31	21,68
mice protein	23,13	22,80	**27,12**	26,30	22,39
diabetes	66,88	**67,88**	-	67,74	67,63
vehicle	48,09	47,56	**52,64**	50,70	36,85
glass	52,55	50,67	53,52	**55,55**	52,05

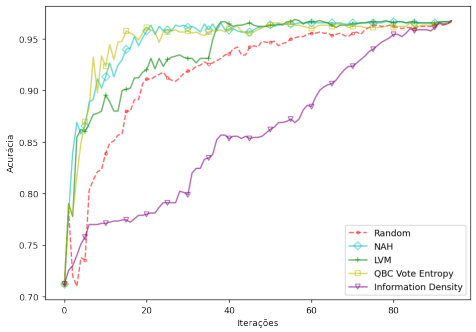
Tabela 4.4: Valores da área abaixo da curva de acurácia para SVM. Está marcado em negrito e sublinhado as estratégia que obtiveram melhor e pior resultado, respectivamente.

mantendo os parâmetros padrões para SVM's da biblioteca *scikit-learn* (18).

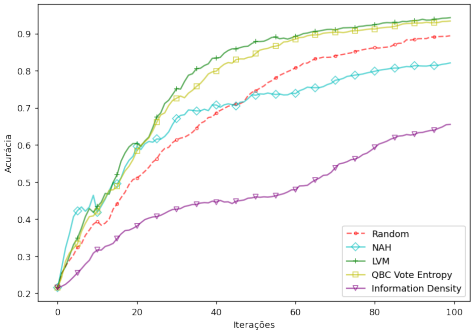
É possível entender a *Simple Margin* como um caso específico da *Nearest to All Hyper-Planes*, portanto no caso dos *datasets* binários (*diabetes* e *breast cancer*) a NAH estará representando o *Simple Margin*. Já a *Least Votes Margin* não poderá ser utilizada nesse casos, visto que não há votação.

Analisando a tabela 4.4, novamente as estratégias de Active Learning se mostram eficientes. Para o SVM, mostra-se também que as estratégias NAH e LVM, criadas pensando na teoria do SVM, foram as que apresentaram melhor resultado em 7 dos 9 datasets.

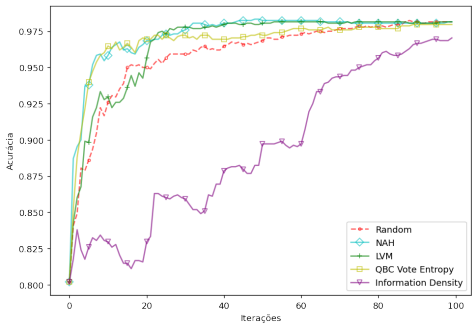
Com exceção dos *datasets* binários, é verificável que a estratégia de *Information Density*, que combina incerteza com a representatividade de uma instância, obteve os piores resultados com o SVM, sendo pior até que a seleção aleatória. Portanto, é uma estratégia que não vale a pena utilizar da forma implementada (em conjunto com NAH e utilizando similaridade por cosseno). Ainda assim, não é possível descartar *Representative Sampling* ou *Density Weighting* como um todo para SVM's. Já foi mostrado que representatividade apresenta bons resultados em problemas de classificação de *temporal relation* (15), utilizando a média das distâncias entre os membros da *Pool* para remover o efeito negativo de *outliers*.



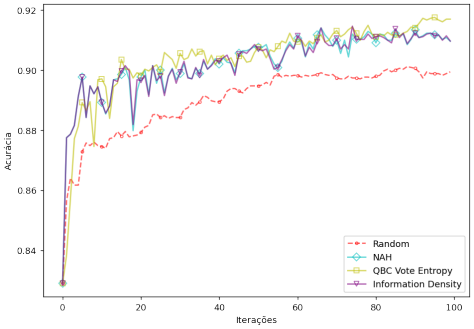
4.4(a): Iris



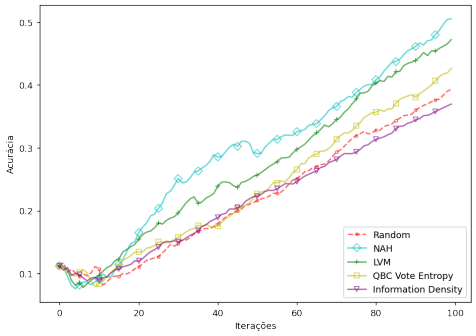
4.4(b): Digits



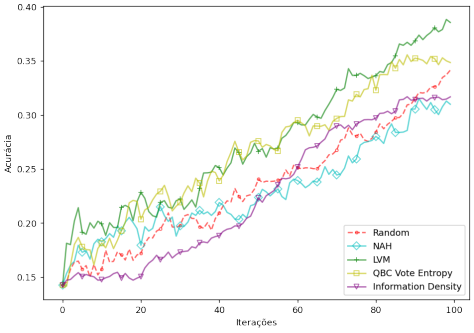
4.4(c): Wine



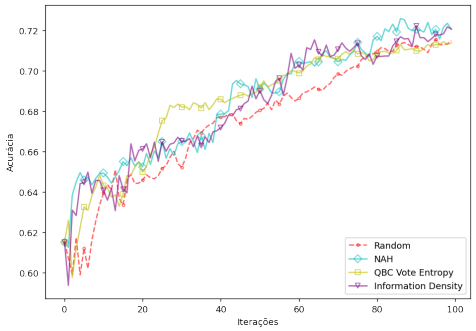
4.4(d): Breast Cancer



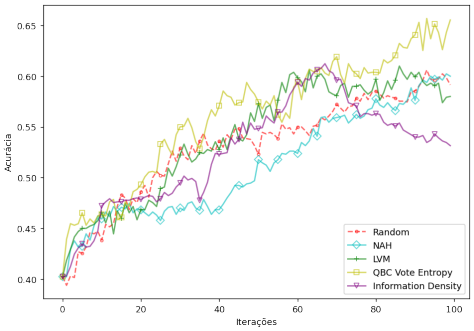
4.4(e): Olivetti Faces



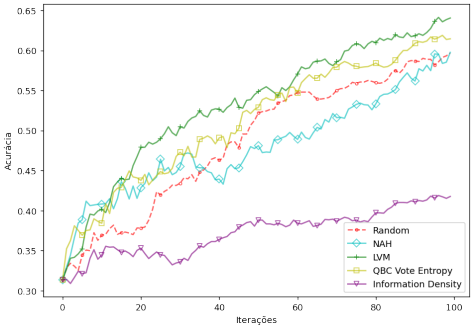
4.4(f): Mice Protein



4.4(g): Diabetes



4.4(h): Glass



4.4(i): Vehicle

5
Conclusões

Um passo importante para problemas de classificação é obter uma base para treino do modelo já rotulada. Esse processo, caso a base precise ser montada do zero, necessita de um grande trabalho manual executado por um especialista na área de estudo, que frenquentemente pode ser caro e/ou demorado. A fim de obter bons resultados reduzindo o esforço que surge o Active Learning, um método que, utilizando certas heurísticas e estratégias, diminui o trabalho feito pelo especialista, aumentando a produtividade e reduzindo os gastos.

Este trabalho apresentou e implementou diversas dessas estratégias de Active Learning que foram desenvolvidas durante as últimas décadas, apresentando uma extensa variedade de experimentos.

Com estes experimentos foi possível comprovar a eficácia do método estudado e entender a necessidade para o seu uso. O foco do estudo não foi voltado para resolver um problema em específico, mas sim verificar resultados de forma geral, portanto caso seja dedicado mais esforço em um único problema os resultados podem se mostrar ainda melhores, aplicando ajustes específicos ao problema.

Foi averiguável também que não há uma estratégia ou um modelo que consistentemente apresente melhores resultados, obtivemos resultados variados entre os experimentos. De toda forma, na maioria dos experimentos todas as estratégias de Active Learning conseguiram acurácias maiores com uma base rotulada menor quando comparadas a estratégia aleatória.

Para trabalhos futuros também é possível ampliar o leque de estratégias. Existem diversas outras estratégias como *Variance Reduction* e *Expected Error Reduction*, que foram pouco exploradas neste trabalho. Além destas, é possível encontrar novas soluções, mais interessantes, quando se foca em uma área ou em um modelo, como é o caso de *Hinted Sampling* para SVM's (10).

Referências

[1] ANGLUIN, D.. **Queries and Concept Learning**. Machine Learning, 2:319 – 342, 1988.

[2] SETTLES, B.. **Active learning literature survey**. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[3] LANG, K.; BAUM, E.. **Query learning can work poorly when a human oracle is used**. In: PROCEEDINGS OF THE IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, p. 355–340, IEEE Press, 1992.

[4] SCHUMANN, R.; REHBEIN, I.. **Active learning via membership query synthesis for semi-supervised sentence classification**. In: PROCEEDINGS OF THE 23RD CONFERENCE ON COMPUTATIONAL NATURAL LANGUAGE LEARNING, p. 472–481, Hong Kong, China, November 2019. Association for Computational Linguistics.

[5] LIANTAO WANG, XUELEI HU, BO YUAN, JIANFENG LU. **Active learning via query synthesis and nearest neighbour search**. Neurocomputing, 2013.

[6] DAVID COHN, LES ATLAS, R. L. M. E.-S. R. M. I. M. A.; PARK, D.. **Training connectionist networks with queries and selective sampling**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NIPS), p. 566–573, 1990.

[7] A. FUJII, T. TOKUNAGA, K. I.; TANAKA, H.. **Selective sampling for example-based word sense disambiguation**. Computational Linguistics, 24(4):573 – 597, 1998.

[8] DANKA, T.; HORVATH, P.. **modAL: A modular active learning framework for Python**. available on arXiv at `https://arxiv.org/abs/1805.00979`.

[9] SETTLES, B.; CRAVEN, M.. **An analysis of active learning strategies for sequence labeling tasks**. In: PROCEEDINGS OF THE CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP), p. 1069–1078. ACL Press, 2008.

[10] CHUN-LIANG LI, C.-S. F.; LIN, H.-T.. **Active learning with hinted support vector machine**. In: JMLR: WORKSHOP AND CONFERENCE PROCEEDINGS, p. 25:221–235. Asian Conference on Machine Learning, 2012.

[11] TONG, S.; KOLLER, D.. **Support vector machine active learning with applications to text classification**. Journal of Machine Learning Research (JMLR), p. 2:45–66, 2001.

[12] TONG, S.; KOLLER, D.. **Support vector machine active learning for image retrieval**. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON MULTIMEDIA (MM), p. 107–118. ACM Press, 2001.

[13] TONG, S.. **Active learning: Theory and applications**. PhD thesis, Stanford University, 2001.

[14] JAN KREMER, K. S. P.; IGEL, C.. **Active learning with support vector machines**. 2014.

[15] SEYED MIRROSHANDEL, G. G.-S.; NASR, A.. **Active learning strategies for support vector machines, application to temporal relation classification**. In: PROCEEDINGS OF THE 5TH INTERNATIONAL JOINT CONFERENCE ON NATURAL LANGUAGE PROCESSING, p. 56—-64. AFNLP, 2011.

[16] LEWIS, D.; GALE, W.. **A sequential algorithm for training text classifiers**. In: PROCEEDINGS OF THE ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, p. 3—-12. ACM/Springer, 1994.

[17] SCHEFFER, T.; DECOMAIN, C. ; WROBEL, S.. **Active hidden markov models for information extraction**. In: Hoffmann, F.; Hand, D. J.; Adams, Nialland Fisher, D. ; Guimaraes, G., editors, ADVANCES IN INTELLIGENT DATA ANALYSIS, p. 309–318, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[18] PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M. ; DUCHESNAY, E.. **Scikit-learn: Machine learning in Python**. Journal of Machine Learning Research, 12:2825–2830, 2011.

[19] C.E. SHANNON. **A mathematical theory of communication**. Bell System Technical Journal, p. 27:379–423,623–656, 1948.

[20] DAGAN, I.; ENGELSON, S.. **Committee-based sampling for training probabilistic classifiers**. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML), p. 150–157. Morgan Kaufmann, 1995.