Ensembles

Inteligência na Web e Big Data

Fabricio Olivetti de França e Thiago Ferreira Covões
folivetti@ufabc.edu.br, thiago.covoes@ufabc.edu.br

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

UFABC

CMCC
Centro de Matemática, Computação e Cognição
Center for Mathematics, Computation and Cognition

Introdução

O que é um ensemble?

- As técnicas de classificação/regressão vistas até o momento consideram um classificador que foi treinado em cima da base dados.
- Mas e se pudéssemos combinar vários modelos (classificadores/regressores)?
  - Para simplificar vamos focar em classificação

Wisdom of Crowds

- Em 1906, houve uma competição para adivinhar o peso de um boi, da qual 800 pessoas participaram. A média dos palpites (1.197lb) foi extremamente próximo do peso real (1.198lb).

Analogia:

- Você suspeita que está doente:
  - Você pode ir a **um** médico e confiar na opinião dele;
  - Ou você pode ir a **vários** e tentar agregar as opiniões.

Combinando classificadores

- Podemos montar um comitê (ensemble) de classificadores na esperança de obter um classificador melhor.
- A partir de uma mesma base, treinamos vários classificadores.
- Quando um novo exemplo aparece, teremos várias predições que devem ser agregadas de alguma forma.

Agregar predições

- A classe pode ser obtida simplesmente contando os votos de cada classe;
- Ou ainda, podemos ponderar o voto de cada classificador pela sua acurácia.

Exemplo

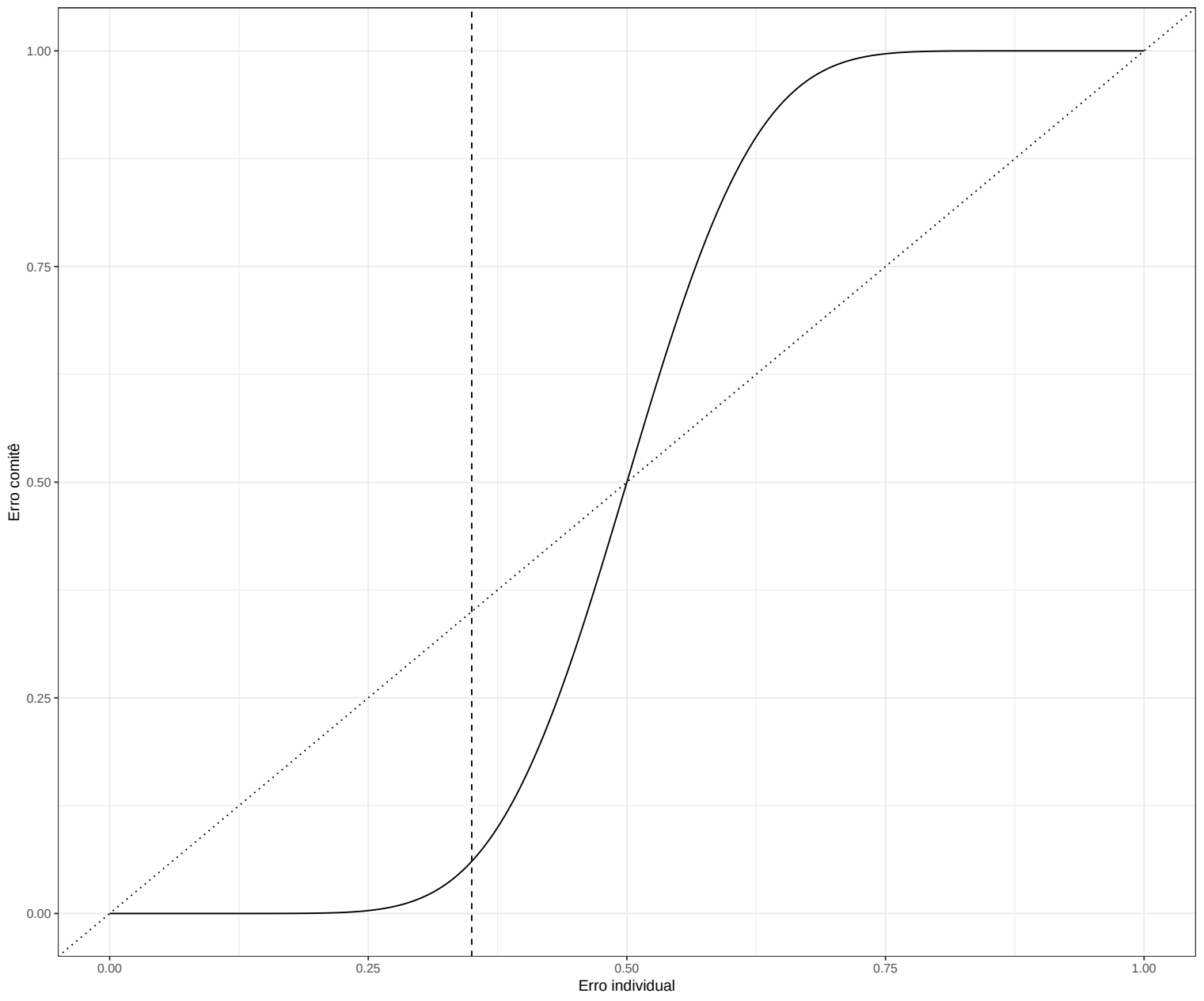
- Considere um comitê com 25 classificadores binários, com taxa de erro $\epsilon = 0.35$.
- Geramos a saída do comitê simplesmente pegando a classe com a maior parte dos votos.

Exemplo

- Se todos os classificadores são iguais, nossa taxa continua 0.35.
- No entanto, se temos 25 classificadores independentes (seus erros não tem relação entre si):

$$\epsilon_{ens} = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.06$$

Exemplo


1.00
0.75
0.50
0.25
0.00
Erro comitê
0.00
0.25
0.50
0.75
1.00
Erro individual

Exemplo

- Para fazer a predição de um objeto de teste, podemos assumir que temos um conjunto de modelos gerados
- É fácil gerar os dados no formato (id-teste, [$\mathbf{x}_t$, modelo]) para cada modelo (basta um map)
- Podemos fazer as predições no map, combinando pelo identificador no reduce
  - Se a agregação é feita pelo voto majoritário precisamos apenas da predição, caso contrário precisamos ser capaz de identificar o modelo no reduce

Exemplo

```
1  def mapper(key : int, value : tuple) -> (int, String):
2    xt = first(value)
3    model = second(value)
4    pred = model.predict(xt)
5    yield (key, (model.id, pred))
6
7  def reducer(key : int, values : [tuple]) -> (int,
   ↪  String):
8    preds = second(values)
9    pred_ens = scipy.stats.mode(preds)
10   yield (key, pred_ens)
```

Exemplo (com pesos)

```
def mapper(key : int, value : tuple) -> (int, tuple):
  xt = first(value)
  model = second(value)
  #predict_proba similar ao score
  pred = model.predict_proba(xt)
  yield (key, (model.id, pred))

def reducer(key : int, values : [tuple]) -> (int,
 ↪  String):
  votes = np.zeros(num_classes)
  for model_id, pred in values:
    votes += weight_function(model_id, pred)
  pred_ens = np.argmax(votes)
  yield (key, classes[pred_ens])
```

Analogia (Parte II)

- Você tem que tomar uma decisão para a empresa:
  - Você pode ouvir **um** conselheiro;
  - Você pode ouvir **um grupo** de conselheiros que normalmente concordam com você;
  - Você pode ouvir **um grupo** de conselheiros com perspectivas complementares.
- Qual você escolhe? Deixando o ego de lado :)

Perspectivas Complementares

- Difícil de garantir
- Podemos manipular:
  - Os atributos de entrada (Random Projections);
  - Os objetos do treinamento (Bagging, Boosting);
  - Parametrizações dos algoritmos.

Gerando variabilidade

Manipulando os atributos

- Simples
- Projeção dos dados para algum sub-espaço
    - Basta um map

Manipulando os objetos

- Entre as principais opções, vamos focar em Bagging (Bootstrap Aggregating) e Boosting

O que é um Bootstrap?

- Se queremos k classificadores, vamos gerar k bases a partir da base original.
- Cada "sub-base" terá o mesmo número de objetos que a original;
  - A chance de um objeto ficar de fora é aproximadamente 1/3
- Escolheremos os objetos de acordo com uma distribuição uniforme, **com reposição**.

O que é Bagging?

- Bootstrap aggregating
  - Gera modelos e cada amostra bootstrap e combina as saídas

Lembram do perceptron?

Vamos usá-lo como exemplo.

Exemplo

Considerem que queremos classificar:

x	0	.2	.4	.6	.8	1
y	1	1	-1	-1	1	1

- Qual ponto de corte escolhemos?
- Podemos escolher pela esquerda ou pela direita;
- De qualquer maneira, erraremos dois objetos.

Exemplo

x	0	.2	.4	.4	.6	.6
y	1	1	-1	-1	-1	-1

Perceptron A: $x < 0.4 \rightarrow y = 1$

Exemplo

x	.4	.4	.6	.6	.8	1
y	-1	-1	-1	-1	1	1

Perceptron B: $x > 0.6 \rightarrow y = 1$

Exemplo

x	0	.2	.2	.8	1	1
y	1	1	1	1	1	1

Perceptron C: $x \geq 0 \rightarrow y = 1$

Exemplo

- Perceptron A: $x < 0.4 \rightarrow y = 1$
- Perceptron B: $x > 0.6 \rightarrow y = 1$
- Perceptron C: $x \geq 0 \rightarrow y = 1$

x	0	.2	.4	.6	.8	1
y	1	1	-1	-1	1	1

Observações

- Bagging reduz a variância;
- Logo, funciona melhor com classificadores instáveis, ou seja, sensíveis a perturbações menores no conjunto de treinamento;
- Bem útil para dados ruidosos.

Possível implementação para grandes bases

- Para gerar uma amostra bootstrap poderíamos gerar de forma sequencial
  - Mas, no nosso cenário, seria ideal que pudéssemos tratar cada objeto isoladamente
- Uma vez que N é grande, podemos aproximar $B(N, \frac{1}{N})$ por Poisson(1).
- Para cada objeto geramos B valores $z \sim \text{Poisson}(1)$
  - Cada valor z reflete o número de vezes que o objeto foi selecionado no b-ésimo bootstrap
- Não necessariamente os bootstraps terão o mesmo tamanho
  - Podemos filtrar os indesejados

Possível implementação para grandes bases

```
1  def mapper(key : int, value : tuple) -> (int, tuple):
2    xi = first(value)
3    yi = second(value)
4    bs = np.random.poisson(1, B)
5    for ib, qtd in enumerate(bs):
6      if qtd > 0:
7        key2 = "b" + str(ib)
8        yield (key2, (xi, yi, qtd))
9
10 def reducer(key : int, values : [tuple]) -> (int,
   ↪  Model):
11   model = Perceptron()
12   for xi, yi, qtd in values:
13     model.partial_fit(xi, yi, sample_weights = qtd)
14   yield (key, model)
```

Outras alternativas

- Assumimos que o modelo poderia ser aprendido de forma incremental
  - Em grandes bases de dados isso é desejado
- Teríamos alternativas ?
  - Bag of Little Bootstraps (BLB)

Bag of Little Bootstraps

- Ideia básica é gerar R subconjuntos disjuntos dos dados com tamanho b, sendo $b < n$
- Para cada subonjunto são feitos B bootstraps e computa-se a média das estimativas em cada um
- Selecionamos $b < n$ de forma que os dados possam ser carregados em memória

Boosting

O que é?

- Processo iterativo de mudar a distribuição dos dados, de modo que cada classificador foque mais nos exemplos que o anterior teve dificuldade.
- Cada objeto tem um peso associado, que representa a chance deste objeto ser escolhido para a base.

Algoritmo

1. Inicialmente, cada exemplo tem $1/N$ de chance de ser escolhido;
2. Gerar uma amostra da base de acordo com as probabilidades, e treinar um classificador;
3. Atualizar o peso/chance dos objetos: os pesos dos objetos classificados incorretamente são incrementados, e os corretos são decrementados;
4. Voltar para o passo 2 até se atingir o número desejado de classificadores.

Mais opções?

O que fizemos até agora?

- Após treinados k classificadores base, estamos juntando as predições deles de formas simples.

E se fôssemos além?

- E se treinássemos um classificador para juntar estas predições da melhor forma que ele encontrar?

Stacking

- Chamamos essa técnica de stacking, pois estamos empilhando classificadores.

Stacking

- É desejável que se tenha diferentes classificadores como base.
- Intuitivamente, o classificador final se beneficia de diferentes “visões” do mesmo problema: (hiperplano, árvore de decisão, vizinhos mais próximos, etc).

Analogia (Parte III)

- Quando você toma uma decisão:
  - Você leva em conta apenas a sugestão dos outros
  - Você leva em conta a sugestão dos outros e a informação que eles usaram para basear a decisão deles

Cascading

- Ao rodar um classificador podemos adicionar os scores de cada classe como novos atributos
- Um novo classificador é induzido neste novo espaço

Referências

- Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.
- Gama, J., & Brazdil, P. (2000). Cascade generalization. Machine learning, 41(3), 315-343.
- Wolpert, D. H. (1992). Stacked generalization. Neural networks, 5(2), 241-259.
- Chamandy, N., Muralidharan, O., Najmi, A., & Naidu, S. (2012). Estimating uncertainty for massive data streams.
- Kleiner, A., Talwalkar, A., Sarkar, P., & Jordan, M. I. (2014). A scalable bootstrap for massive data. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 76(4), 795-816.