Implementação do algoritmo básica K-NN Classification

Introdução

O algoritmo de KNN é um método para classificar objetos com base em objetos de treinamento mais próximos: um objeto é classificado de acordo com o voto da maioria de seus vizinhos. Aqui, o algoritmo KNN é implementado em linguagem orientado a objetos, e suporta diferentes métodos métricos e diferentes escolhas de K. Esse relatório está organizado da seguinte forma: Implementação que descreve o design da classe e como KNN é implementado. O resultado experimental é mostrado em Avaliação. Em Discussão, itens incluindo tempo de execução, a escolha de K, a escolha do tipo de métrica e a normalização são discutidos.

Implementação

- **Record:** uma classe básica que simula registros contendo um array duplo para atributos e um inteiro para label. Além disso, esse é o tipo de argumento da interface Metric, que define um método para obter a distância entre dois registros.
- **TrainRecord:** uma subclasse de Record com um número de tipo duplo que armazena sua distância para testar o registro como uma adição.
- **TestRecord:** uma subclasse de Record com um inteiro que armazena o rótulo previsto trabalhado pelo algoritmo KNN como uma adição.
- **Metric:** uma interface que define um método que calcula a distância entre dois registros.
- **CosineSimilarity:** uma classe que implementa Metric e fornece o caminho para obter a semelhança de cosseno entre dois registros. Note que no final do método getDistance(), 1 / cosineSimilarity é retornado para ser unificado com a distância euclidiana e a distância L1. A razão pela qual fazemos isso é porque a distância aumenta quando a semelhança dos cossenos é menor. O caso quando a similaridade de cosseno é igual a 0 também é levado em consideração.
- **L1Distance:** uma classe que implementa Metric e fornece o caminho para calcular a distância L1 entre dois registros.
- **EuclideanDistance:** uma classe que implementa Metric e fornece o caminho para calcular a distância euclidiana entre dois registros.
- **FileManager:** uma classe que fornece métodos estáticos para três finalidades: leitura dados de treinamento, leitura de dados de teste e saída de rótulos previstos.
- **KNN1:** a classe principal que implementa o algoritmo KNN. Detalhes serão discutidos na próxima seção.

Algoritmo

Nesta implementação, o KNN é realizado de acordo com as duas fases abaixo:

1. Encontre K vizinhos mais próximos

Para um registro de teste específico, um array com tamanho K é criado para manter o K mais próximo vizinhos, e o caminho para alcançar este objetivo é percorrer todo o conjunto de treinamento e atualize a matriz se houver um registro de treinamento cuja distância seja menor que a maior distância na matriz original. No final deste processo, o K mais próximo vizinhos são encontrados.

2. **Classifique de acordo com pesos**

Depois que os vizinhos K mais próximos são encontrados, um HashMap é criado para manter os pares <rótulo, peso> passando por todos os vizinhos. Se o HashMap acontece conheça um novo rótulo, <label, 1 / distance> é adicionado diretamente no mapa. De outra forma, uma versão atualizada que adiciona o peso original com 1 / distância é colocado no mapa. Após o HashMap ser construído corretamente, este programa passará pelo HashMap inteiro e encontre o rótulo associado ao maior peso.

Resultados

Dataset	iris	glass	vowel	vehicle	letter	DNA
K	1	1	3	3	3	5
Tipo metrica	1	0	2	1	0	2
Accuracy	96%	67,29%	94,95%	66,19%	94,32%	80,29%
Tempo em segundos	0,15	0,14	0,39	0,13	59,46	10,61

Discussão

- **O tempo de execução é O (m * d * n), m = # exemplos no conjunto de treinamento, d=# dimensões,n = # exemplos no conjunto de testes**

Por exemplo, existem 10000 amostras de treinamento, 16 dimensões e 10000 amostras de teste no conjunto de dados de carta. Como resultado, o tempo de execução do conjunto de dados de letras é significativamente maior que os outros.

- **A escolha do K**

Para o conjunto de dados de DNA, porque os dados são compostos de 1s e 0s, é muito provável que cada registro de teste possa ter vários registros de treinamento cujas distâncias para esse registro de teste são as mesmas. Como resultado, um K grande gerará um resultado relativamente melhor para o caso de DNA. Para melhor ilustrar a ideia acima, o experimento com K diferente é feito, e o resultado é mostrado da seguinte forma:

K	2	3	4	5
DNA Accuracy	71,75%	78,41%	79,03%	80,29%

No entanto, para alguns conjuntos de dados, um K relativamente maior pode não levar a melhores resultados. Por exemplo, o experimental com K diferente é feito na vogal do conjunto de dados e o resultado é mostrado da seguinte forma:

K	2	3	4	5
Vowel Accuracy	96,36%	96,36%	94,95%	93,54%

Como a tabela acima sugere, a precisão diminui com K aumentando. Ao todo, a escolha de K depende das características do conjunto de dados, e é melhor testar K diferentes para obter bons resultados.

- **A escolha do tipo de métrica**

Métodos métricos diferentes também têm impacto na execução de tarefas de classificação. Aqui, o conjunto de dados do veículo é testado e o resultado a seguir é elaborado:

metric_type	CosineSimilarity	L1Distance	EuclideanDistance
vehicle accuracy	65,48%	66,19%	64,30%

De acordo com a tabela acima, a distância L1 alcança o melhor resultado, enquanto a distância euclidiana é a pior. Portanto, o tipo de métrica deve ser definido de acordo com as características do conjunto de dados de destino, para que resultados ótimos possam ser obtidos.

- **Normalização**

A normalização também é discutida aqui, e o experimento em conjuntos de dados de vidro e veículo é conduzido para verificar como suas precisões mudam antes e depois da normalização. Além disso, observe que o conjunto de testes é normalizado de acordo com os valores máximo e mínimo no conjunto de treinamento. A tabela a seguir mostra o resultado:

	Sem Normalização	Com Normalização
glass accuracy	67,29%	64,49%
vehicle accuracy	66.19%	68.56%

Para o conjunto de dados do veículo, a precisão aumenta em 2,37% após a normalização. No entanto, esse aumento não se aplica ao conjunto de dados de vidro. A precisão do conjunto de dados de vidro diminui em 2,8% após a normalização. Como resultado, se normalizar ou não depende também do conjunto de dados de destino. Mas, em geral, a normalização ajuda a aumentar a precisão da classificação.

Conclusão

O algoritmo k-neighbor neighbor, que classifica os objetos de acordo com o voto da maioria de seus vizinhos, é implementado com sucesso. Os detalhes sobre como o algoritmo de k vizinhos mais próximos é implementado também são apresentados neste relatório. Além disso, itens como o tempo de execução, a escolha de K, a escolha do tipo de métrica e a normalização são discutidos de acordo com os resultados da experiência. A precisão varia com diferentes K, diferentes tipos de métricas e a opção de normalização. Para obter bons resultados, é necessário definir corretamente com base nas características do conjunto de dados.