Introdução ao Aprendizado de Máquina

Prof. Danilo Silva

EEL7514/EEL7513 - Tópico Avançado em Processamento de Sinais:
Introdução ao Aprendizado de Máquina

EEL / CTC / UFSC

Introdução

O que é Aprendizado de Máquina (*Machine Learning*)?

- Ciência de fornecer aos computadores a habilidade de aprender sem ser explicitamente programados

Tom Mitchell (1997)

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

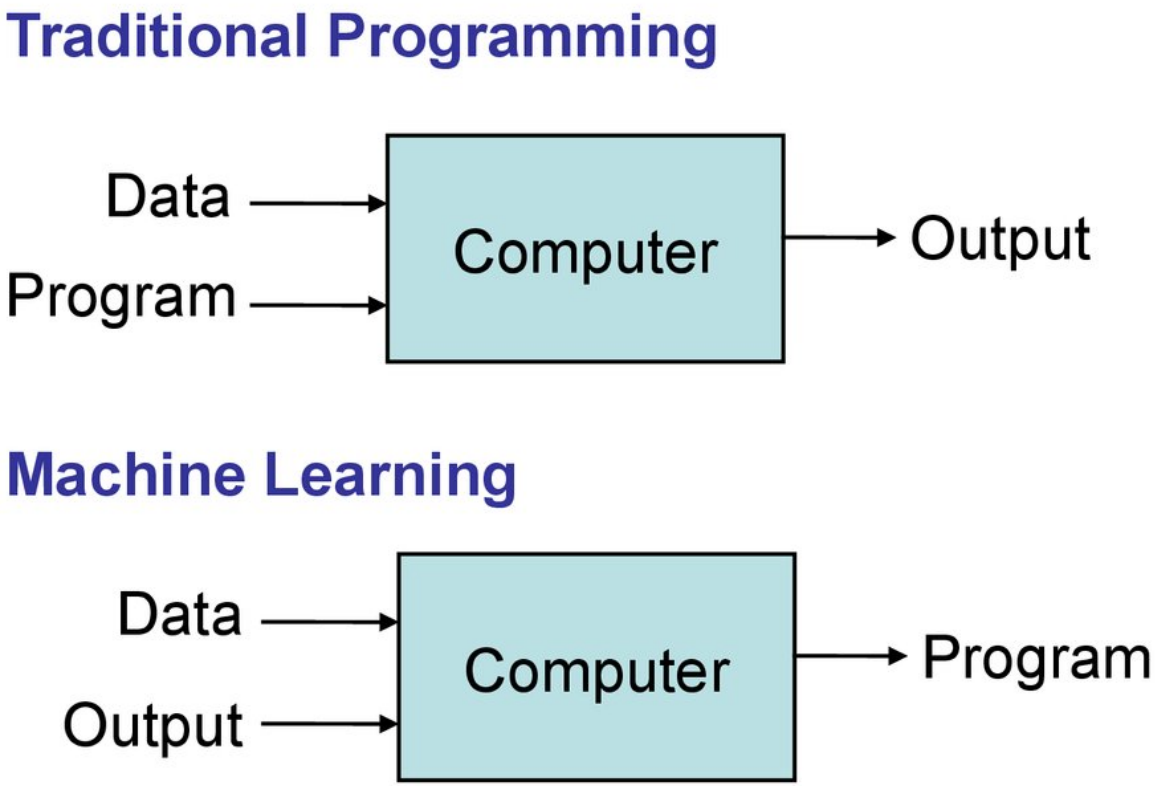
- Experiência refere-se à observação de exemplos (conjunto de variáveis observadas) e/ou de feedback (recompensa/punição) sobre seu desempenho na tarefa

O que é Aprendizado de Máquina (*Machine Learning*)?

Geoffrey Hinton (2007)

- "It is very hard to write programs that solve problems like recognizing a face.
  - We don't know what program to write because we don't know how its done in our brain.
  - Even if we had a good idea about how to do it, the program might be horrendously complicated.
- Instead of writing a program by hand, we collect lots of examples that specify the correct output for a given input.
- A machine learning algorithm then takes these examples and produces a program that does the job.
  - The program produced by the learning algorithm may look very different from a typical hand-written program. It may contain millions of numbers.
  - If we do it right, the program works for new cases as well as the ones we trained it on.
- Massive amounts of computation are now cheaper than paying someone to write a task-specific program."

O que é Aprendizado de Máquina (*Machine Learning*)?



O que é Aprendizado de Máquina (*Machine Learning*)?

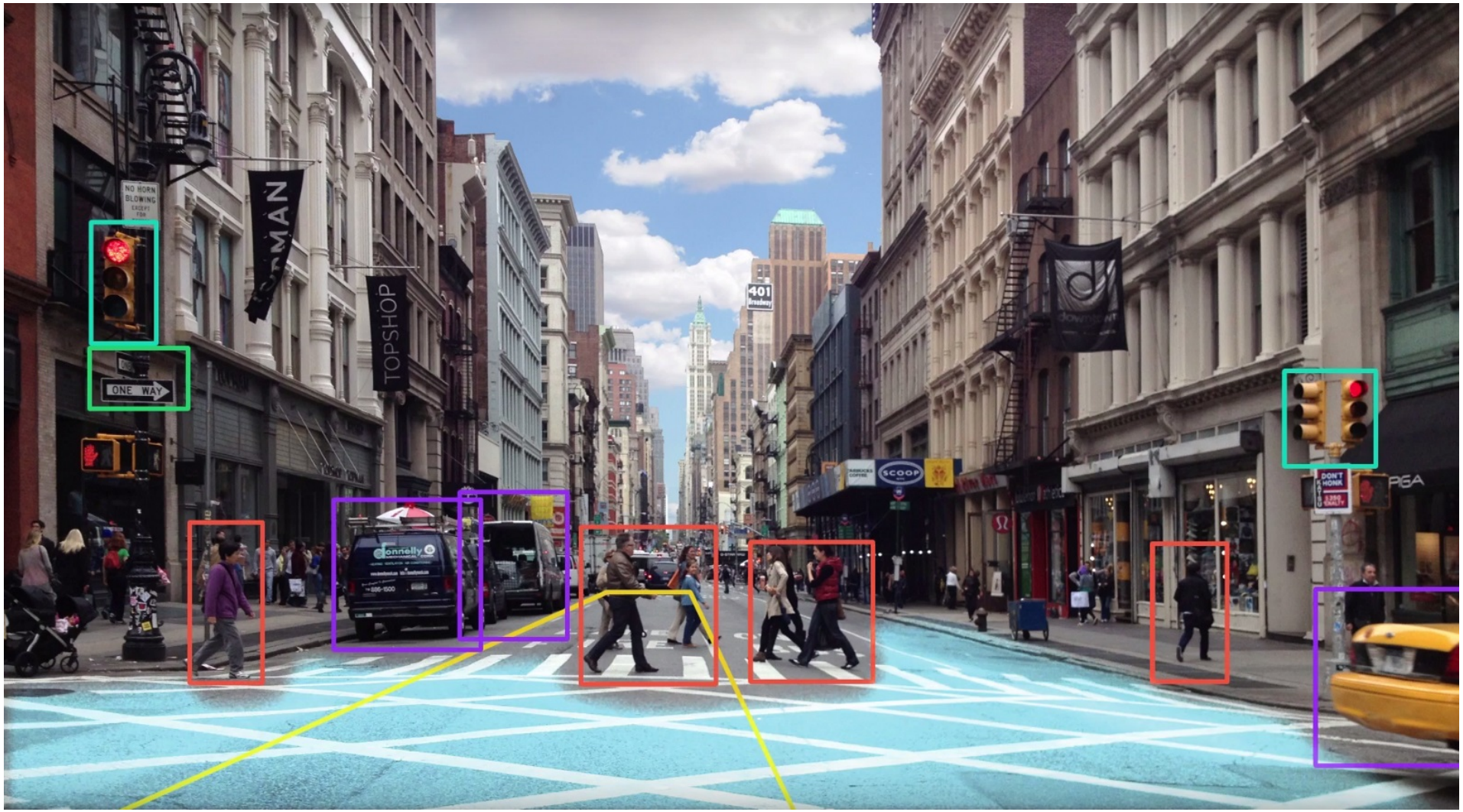
- Ramo da inteligência artificial que combina estatística, otimização, ciência da computação e teoria da informação
- Difere-se fundamentalmente da inteligência artificial simbólica clássica (baseada em regras lógicas e buscas estruturadas programadas por humanos), por permitir o aprendizado a partir de dados e o tratamento estatístico da incerteza
- Difere-se da estatística convencional por dar ênfase aos métodos computacionais e ao modelamento preditivo (foco nos resultados)
- Aprendizado profundo (*deep learning*) é um subcampo do aprendizado de máquina focado em modelos que utilizam múltiplos níveis de representação, tipicamente baseados em redes neurais

Exemplos de Tarefas

- Reconhecimento de caracteres

Exemplos de Tarefas

- Detecção e localização de objetos em imagens:



Exemplos de Tarefas

- Navegação autônoma
- Reconhecimento de faces
- Reconhecimento de fala
- Tradução automática
- Detecção de spam
- Detecção de transações fradulentas
- Diagnóstico médico
- Predição de preços de ações
- Avaliação de risco
- Recomendação de produtos
- Segmentação de clientes
- Jogos eletrônicos
- ...

Tipos de Aprendizado

- Aprendizado supervisionado
- Aprendizado não-supervisionado
- Aprendizado por reforço

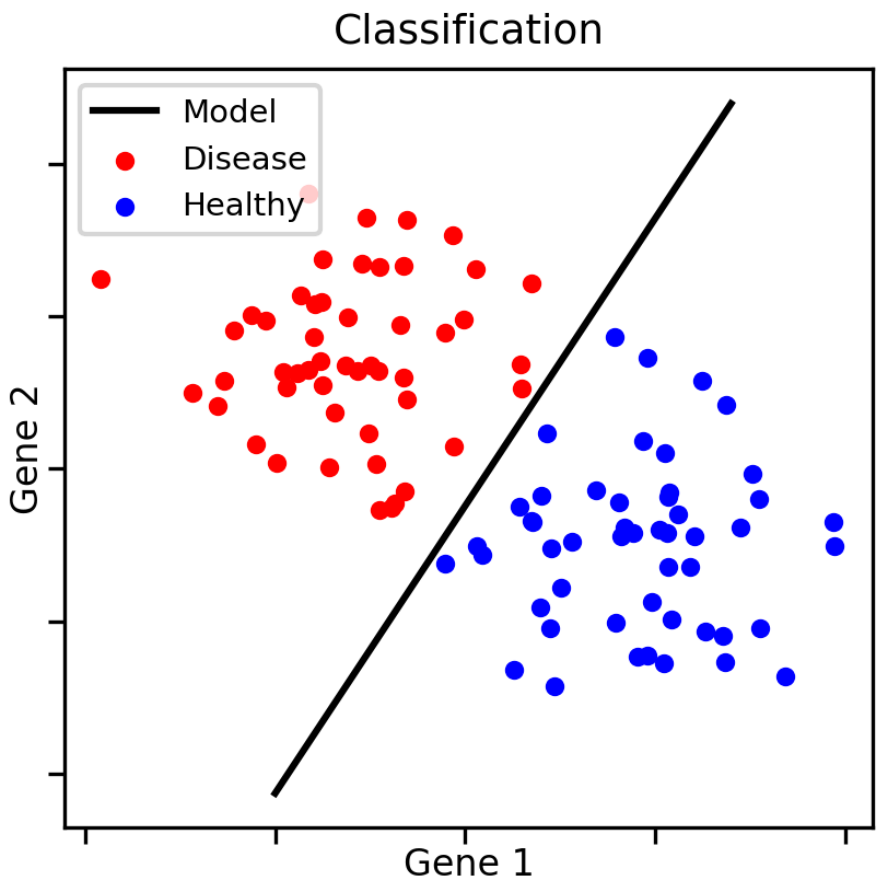
Aprendizado Supervisionado

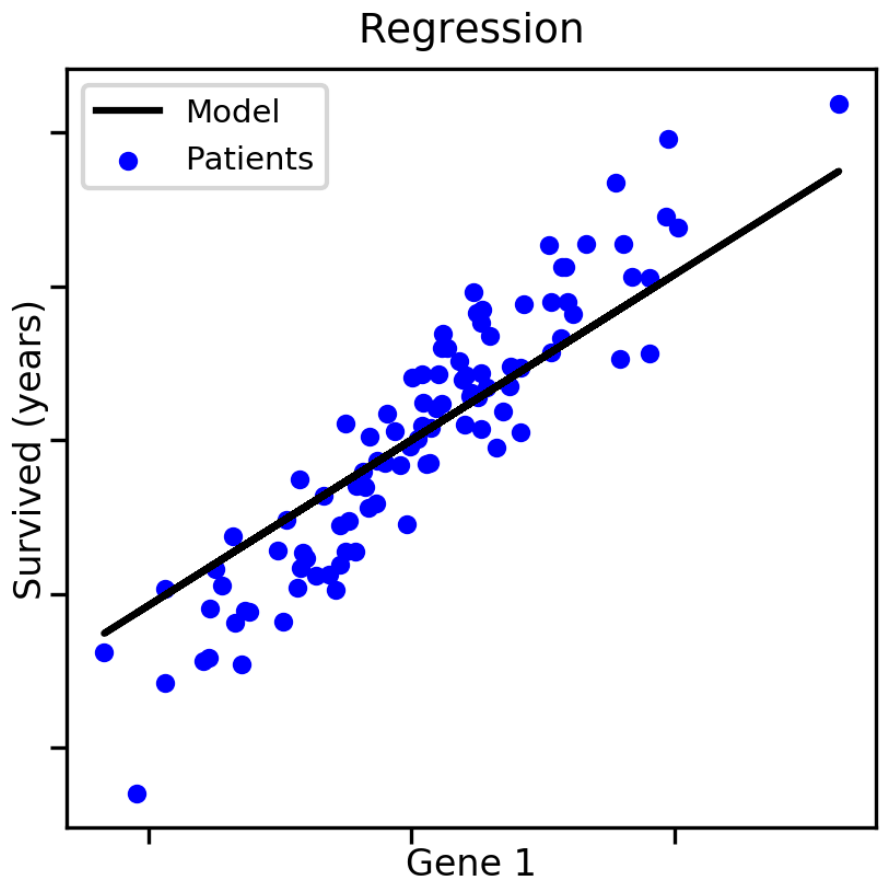
- O objetivo é aproximar uma função desconhecida $f : \mathcal{X} \to \mathcal{Y}$
- Dispõe-se de um conjunto de dados formado por exemplos de vetores de entrada $\mathbf{x}^{(i)}$ rotulados com a saída $y^{(i)} = f(\mathbf{x}^{(i)})$ correspondente

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$$

- Problemas clássicos:
  - Classificação: a variável de saída é discreta: $\mathcal{Y} = \{1, \ldots, K\}$
    - Exemplos: classificação de imagens, detecção de câncer, reconhecimento de fala, detecção de spam
  - Regressão: a variável de saída é contínua: $\mathcal{Y} = \mathbb{R}$
    - Exemplos: predição do preço de um imóvel, predição de demanda por um serviço, avaliação de risco de um empréstimo

Exemplos: Classificação e Regressão

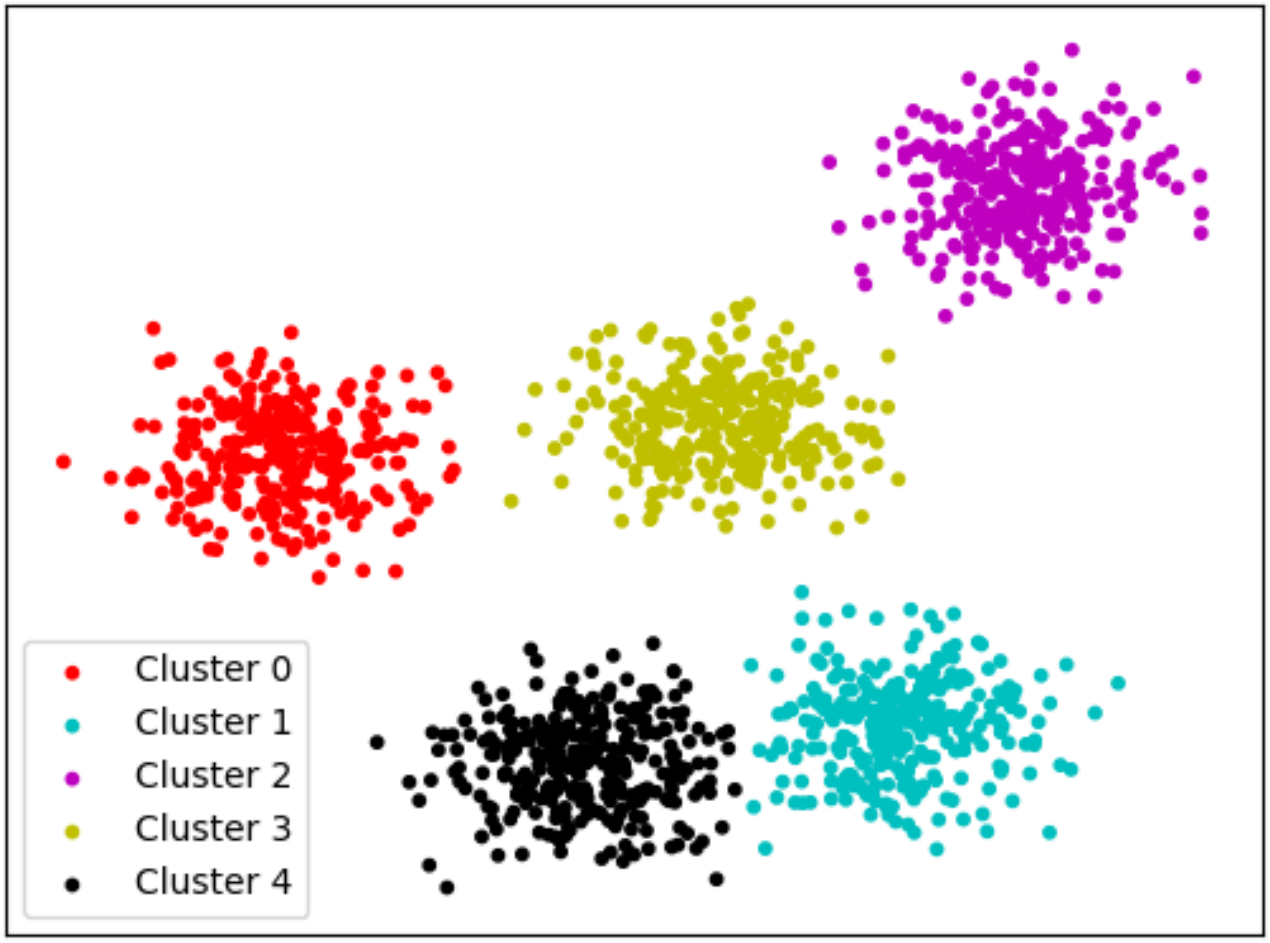




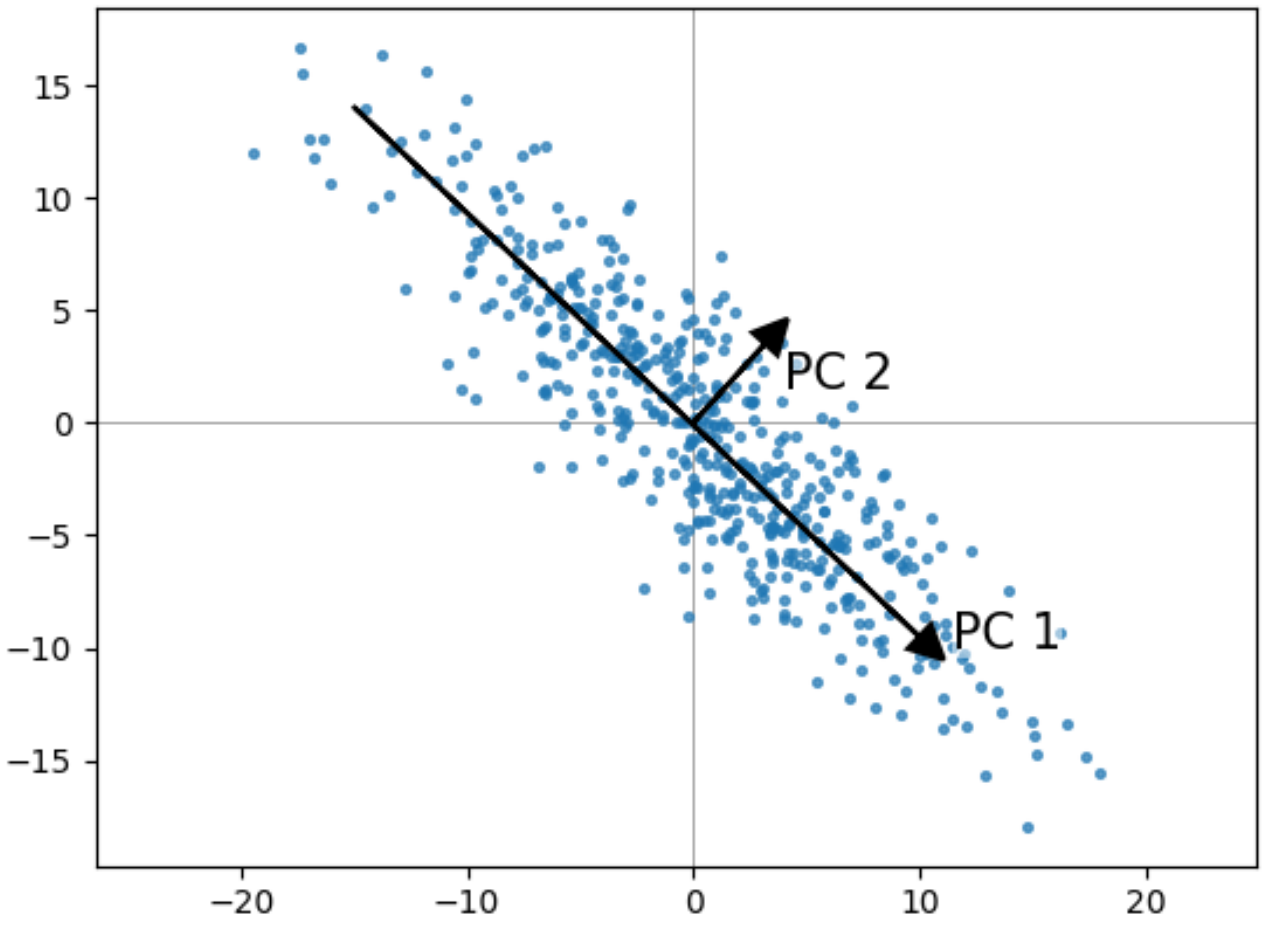
Aprendizado Não-Supervisionado

- Conjunto de dados não-rotulados: $\mathcal{D} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$
- O objetivo é descobrir propriedades "interessantes" da estrutura do conjunto de dados (caracterizada pela densidade $p(\mathbf{x}) = p(x_1, \ldots, x_n)$)
- Problemas clássicos:
  - Clustering: descobrir grupos de exemplos (dados) similares
    - Exemplos: segmentação de mercado, agrupamento de resultados de busca, identificação de famílias de genes, segmentação de imagens
  - Redução de dimensionalidade: encontrar uma representação mais simples dos dados para agilizar algoritmos ou permitir visualização
  - Detecção de anomalias: identificar casos que fogem ao padrão esperado
    - Exemplos: detecção de fraudes, detecção de falhas em sistemas, monitoramento de saúde
  - Sistemas de recomendação: prever a preferência de um usuário por um item a partir de suas preferências por outros itens

Exemplo: Clustering



Exemplo: Redução de Dimensionalidade



Aprendizado por Reforço

- O algoritmo interage com o ambiente, recebendo um sinal de feedback (recompensa/punição) a cada ação tomada
  - Formulação envolve um conjunto de estados $\mathcal{S}$, um conjunto de ações $\mathcal{A}$, uma probabilidade de transição de estados $p(s'|s,a)$ e uma recompensa associada $R_a(s,s')$
- O objetivo é descobrir e executar as melhores ações em cada situação de forma a maximizar a recompensa obtida
- O aprendizado é feito por tentativa e erro e deve balancear descoberta (*exploration*) e aproveitamento (*exploitation*)
- Exemplos: movimentação de robôs, jogos eletrônicos, otimização de redes de comunicação, aplicações financeiras, publicidade
- Frequentemente combinado com técnicas de aprendizado supervisionado para aprender uma função utilidade $Q(s,a)$

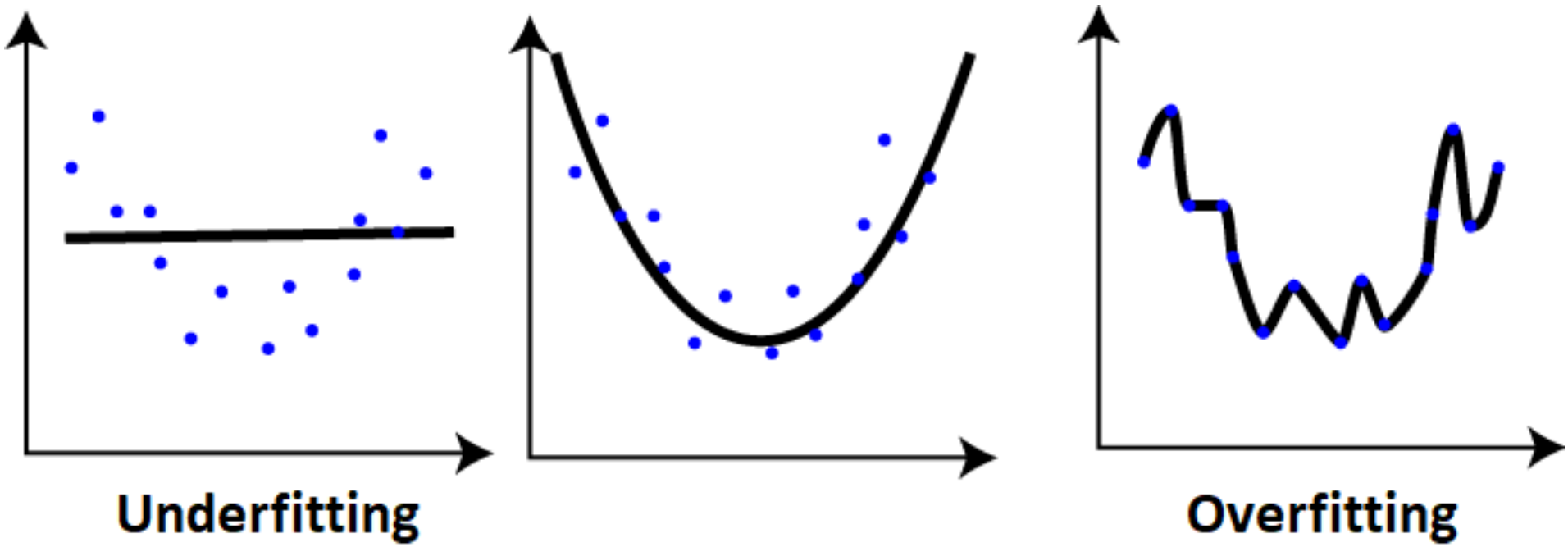
Conceitos Básicos (em Aprendizado Supervisionado)

- Assume-se a existência de uma distribuição (desconhecida) $p(\mathbf{x}, y)$ relacionando um vetor de atributos (*feature vector*) $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ e seu respectivo rótulo ou valor-alvo $y \in \mathcal{Y}$
- São conhecidas amostras dessa distribuição chamadas de exemplos de treinamento $(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})$
- Um modelo de aprendizado determina uma função hipótese $g : \mathbb{R}^n \to \mathcal{Y}$ que realiza a predição $y = g(\mathbf{x})$
  - A função g é escolhida dentre um espaço de hipóteses $\mathcal{G}$
  - Alguns métodos também modelam $p(\mathbf{x}, y)$ ou $p(y|\mathbf{x})$
- Um modelo é dito paramétrico se g depende de um número fixo de parâmetros que independe de m (caso contrário é dito não-paramétrico)

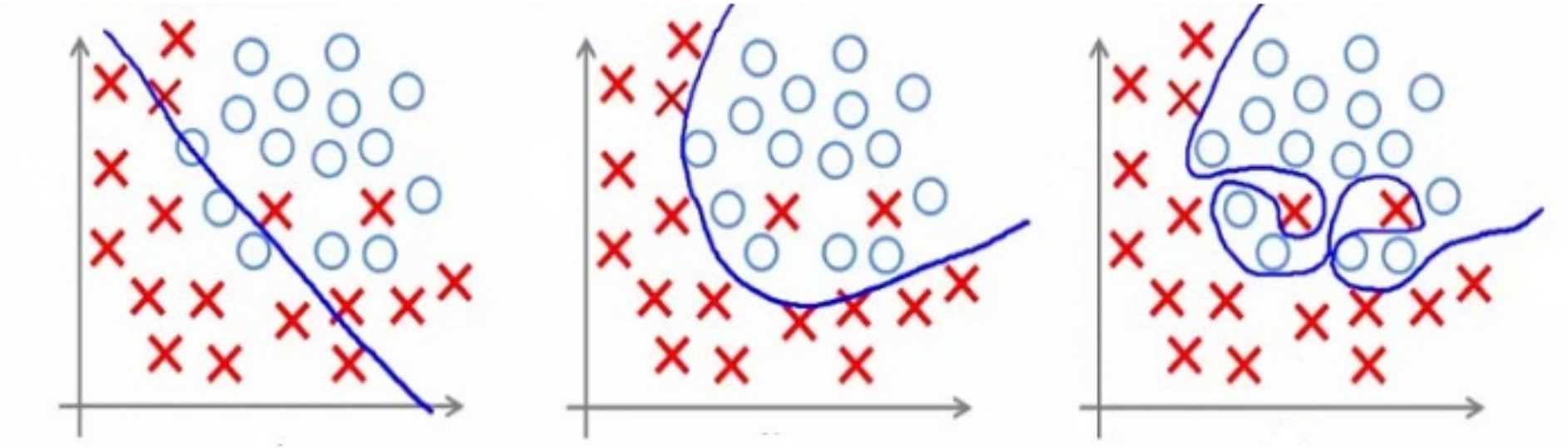
Conceitos Básicos (em Aprendizado Supervisionado)

- O desempenho do modelo é avaliado em um conjunto diferente de exemplos rotulados, denominado conjunto de teste
- A habilidade de desempenhar bem no conjunto de teste é chamada de generalização
- Se um modelo não consegue desempenhar bem nem mesmo no conjunto de treinamento, diz-se que ocorre underfitting
- Por outro lado, se o desempenho no conjunto de treinamento é muito melhor do que no conjunto de teste, diz-se que ocorre overfitting
- A habilidade de representar uma gama variada de funções (e portanto de desempenhar bem em qualquer conjunto de treinamento) é chamada de capacidade do modelo
  - A capacidade do modelo controla o tradeoff entre underfitting e overfitting

Underfitting e Overfitting: Regressão



Underfitting e Overfitting: Classificação



Under-fitting

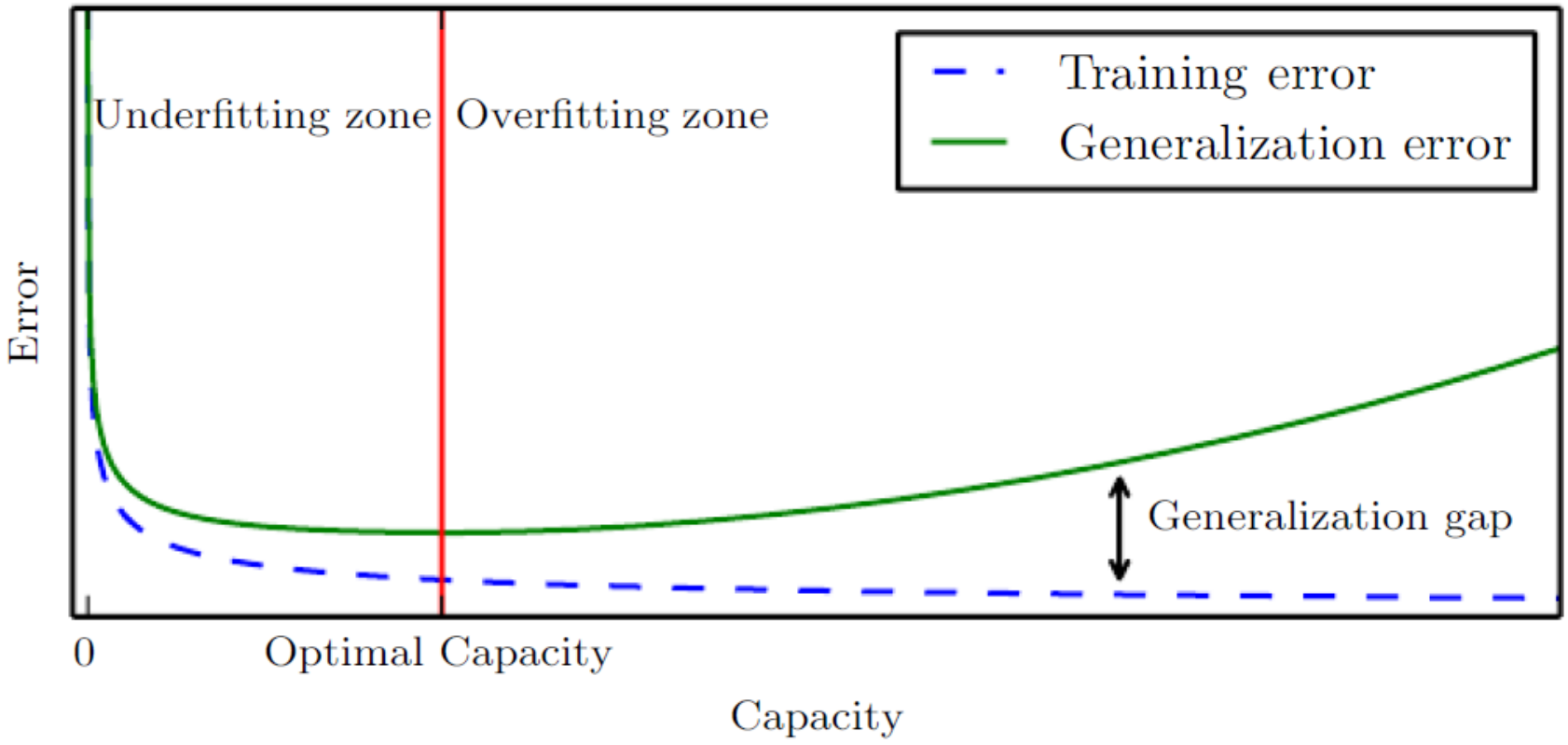
(too simple to explain the variance)

Appropriate-fitting

Over-fitting

(forcefitting -- too good to be true)

Underfitting e Overfitting: Tradeoff



- O tradeoff entre underfitting e overfitting é um problema central na área de aprendizado de máquina

Hiperparâmetros e Validação

- Treinar um modelo corresponde a escolher a hipótese $g \in \mathcal{G}$ que melhor representa o conjunto de treinamento de acordo com alguma métrica de desempenho (função objetivo)
- Parâmetros que alteram o espaço de hipóteses $\mathcal{G}$ ou a métrica de desempenho (em particular, parâmetros que alteram a capacidade do modelo) são chamados de hiperparâmetros
- Estes parâmetros não podem ser escolhidos analisando o desempenho no conjunto de teste, pois isto causaria overfitting no conjunto de teste, comprometendo a generalização
  - O conjunto de teste só deve ser usado no teste final
- Para esse fim reserva-se outro conjunto de dados diferente, denominado conjunto de desenvolvimento ou conjunto de validação

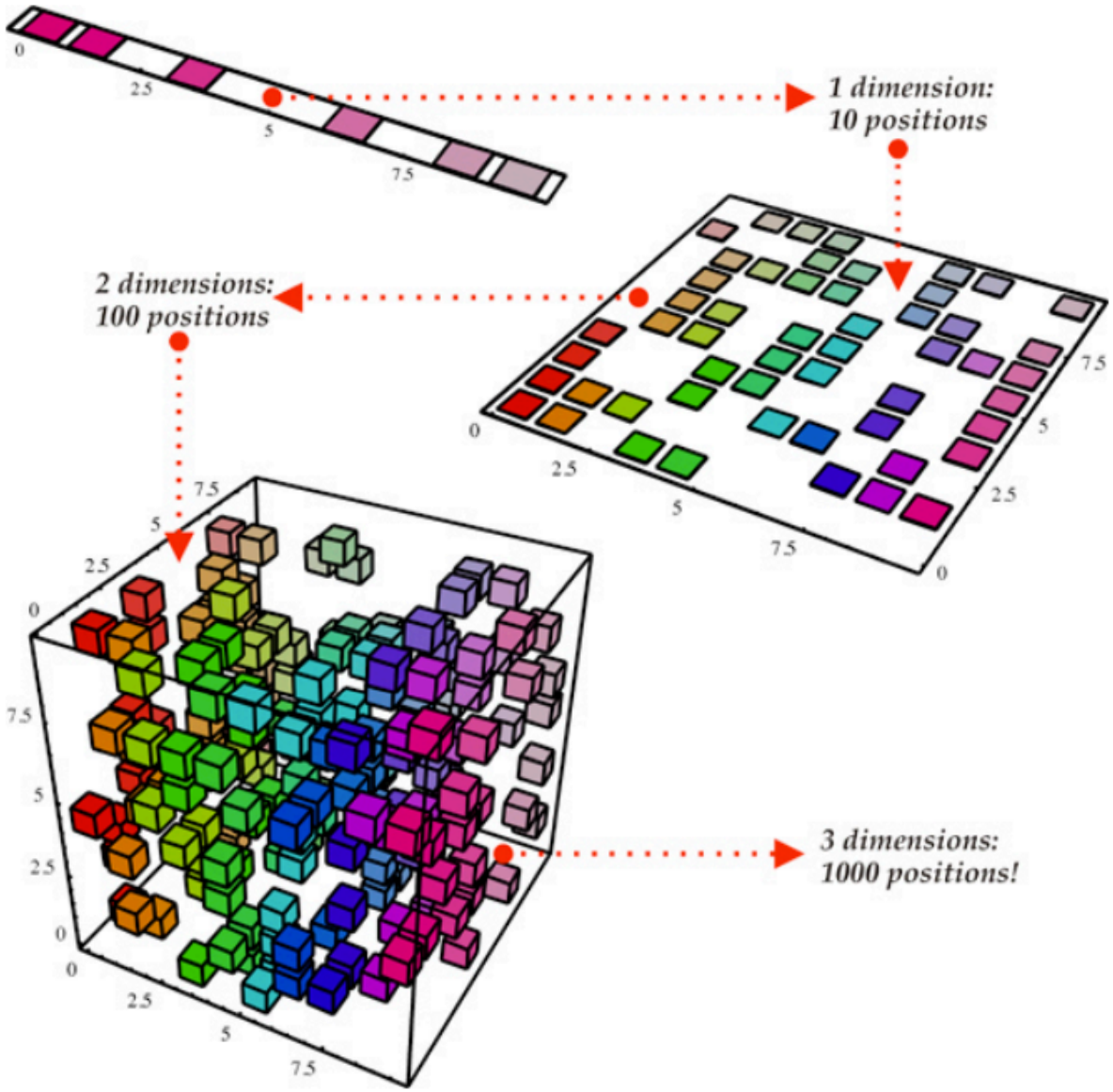
A Maldição da Dimensionalidade

- Muitos problemas de aprendizado de máquina envolvem um grande número de atributos (centenas, milhares ou mesmo milhões)
- No entanto, a dificuldade de tais problemas tende a aumentar com o número de atributos, isto é, com a dimensão n do espaço de atributos
- Em particular, métodos que buscam particionar o espaço de atributos para realizar predições (baseando-se na hipótese de suavidade local) tem dificuldade de generalizar em dimensões elevadas, um problema conhecido como maldição da dimensionalidade
- Isto ocorre pois o número de exemplos de treinamento necessários para que estes métodos generalizem corretamente cresce exponencialmente com a dimensão

A Maldição da Dimensionalidade

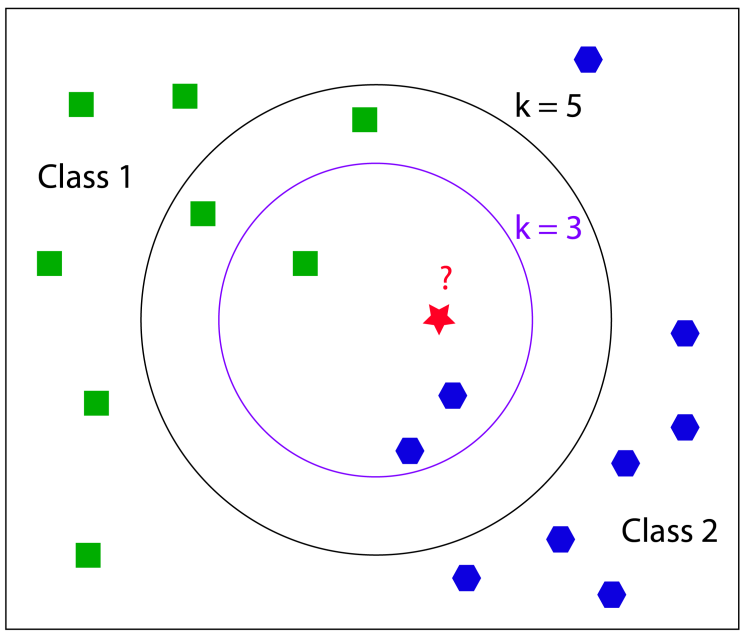
- O número de possíveis configurações distintas de um conjunto de variáveis cresce exponencialmente com o número de variáveis
  - Ex: se $x_j \in \{0, 1\}$, então $(x_1, \ldots, x_n) \in \{0, 1\}^n$ (2^n possibilidades)
- Exemplo:
  - Suponha que $x_j \in [0, 10]$ e que cada ponto $\mathbf{x} \in \mathbb{R}^n$ seja representativo de sua vizinhança a uma distância ≤ 1
  - Para $n = 1$, são suficientes $m = 10$ exemplos de treinamento para cobrir todo o espaço ($\mathcal{D} = \{0.5, 1.5, \ldots, 9.5\}$)
  - Em geral, $m = 10^n$ exemplos de treinamento são necessários

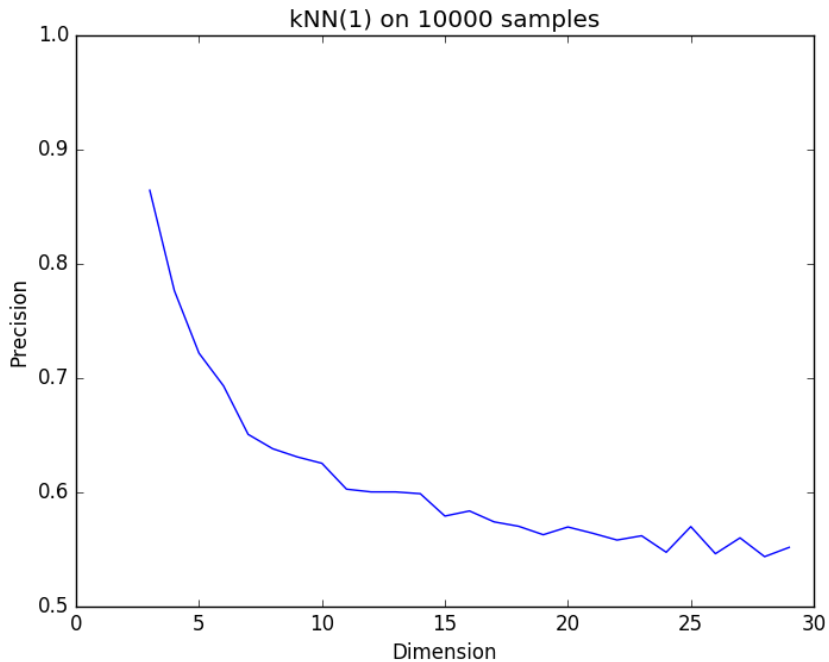
A Maldição da Dimensionalidade



A Maldição da Dimensionalidade

- Algoritmo k-NN (k-*nearest neighbors*): estima o rótulo de um vetor de entrada a partir dos rótulos dos k vizinhos mais próximos
  - calcula a média entre os rótulos, no caso de regressão
  - escolhe o rótulo mais comum, no caso de classificação
- Desempenho se degrada à medida que a dimensão aumenta





Como evitar a Maldição da Dimensionalidade?

- Aprendizado de máquina “tradicional”:
  - Requer conhecimento específico do problema (*domain-specific knowledge*) para desenvolver atributos relevantes (*feature engineering*)
  - Técnicas de redução de dimensionalidade (como PCA) para extrair e/ou selecionar os melhores atributos
- Aprendizado profundo:
  - O uso de múltiplos níveis de representação hierárquicos resulta em uma extração automática e eficiente dos atributos mais importantes
  - Requer um número grande de exemplos de treinamento, porém que não cresce ou cresce de maneira suave com o número de atributos

No Free Lunch Theorem

- Embora possamos determinar empiricamente qual método funciona melhor para um problema específico, não existe um modelo universalmente melhor que todos os outros
- A razão é que cada modelo embute um conjunto de hipóteses sobre o problema, e hipóteses que funcionam bem em um problema podem não funcionar bem em outro
- Conclusão: conhecer uma única ferramenta é insuficiente. É preciso desenvolver inúmeros modelos diferentes para cobrir a ampla variedade de dados que ocorrem no mundo real

"All models are wrong, but some models are useful."

Plano de Ensino

Aulas

1. Apresentação da disciplina
2. Regressão linear (2 aulas)
3. Regressão logística
4. Redes neurais (2 aulas)
5. Máquinas de vetores de suporte (SVM)
6. Técnicas gerais e recomendações
7. Aprendizado não-supervisionado (clustering e PCA)
8. Redes profundas
9. Outros tópicos (1-3 aulas)

Metodologia

- Aula da quinta-feira: exposição da teoria
  - Será passado um exercício computacional para ser feito em casa
- Aula da terça-feira: resolução detalhada do exercício
  - Apresentado pelos alunos com correção pelo professor
- Total de 10 a 12 exercícios, valendo ao todo 50% da nota final

Projeto Final

- Corresponde aos demais 50% da nota final
- A ser feito necessariamente em dupla
- Consiste de:
  - Relatório
  - Apresentação de slides
  - Implementação comentada
- Opções:
  - Técnica abordada na disciplina $\Longrightarrow$ enfoque na aplicação (de interesse do aluno)
  - Técnica não abordada na disciplina $\Longrightarrow$ a aplicação pode ser mais modesta, para permitir mais ênfase na teoria

Software

- Python 3
- Bibliotecas científicas (numpy, matplotlib, etc)
- Jupyter notebook
- Recomendável instalar a distribuição Anaconda