CSC 580 Principles of Machine Learning

03 Nearest Neighbors

Chicheng Zhang

Department of Computer Science

THE UNIVERSITY OF ARIZONA

*slides credit: built upon CSC 580 Fall 2021 lecture slides by Kwang-Sung Jun

Motivation

- Example: given student course survey data, predict whether Alice likes Algorithms course

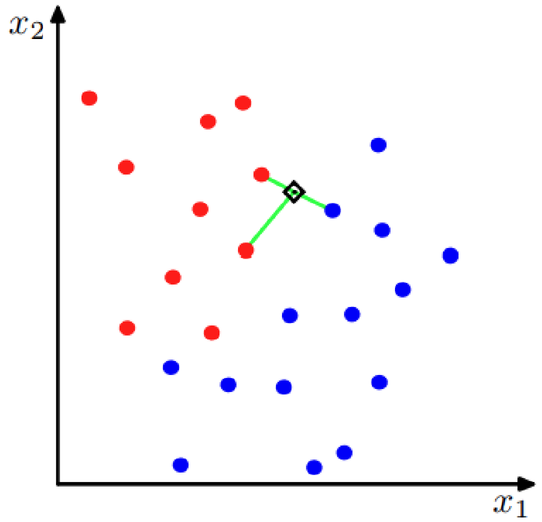
- A natural idea: find a student ``similar'' to Alice & has taken Algorithm course before, say Jeremy
  - If Jeremy likes Algorithms, then Alice is also likely to have the same preference.
  - Or even better, find *several* similar students

k-nearest neighbors (k-NN): main concept

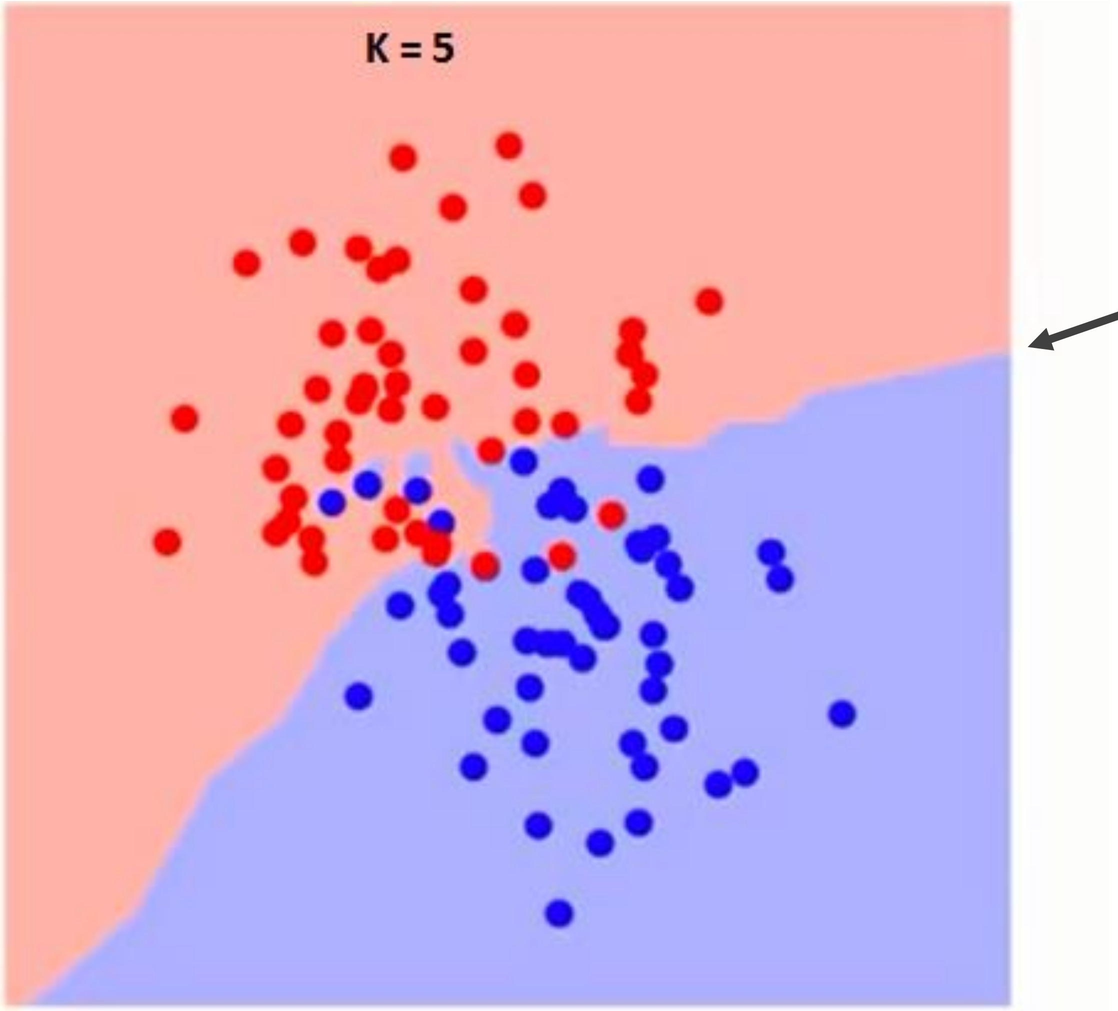
- Training set: $S = \{ (x_1, y_1), \ldots, (x_m, y_m) \}$

- **Inductive bias**: given test example x, its label should resemble the labels of **nearby points**

- Function
  - input: x
  - find the k nearest points to x from S; call their indices $N(x)$
  - output: the majority vote of $\{y_i : i \in N(x)\}$
    - For regression, the average.



k-NN classification example



decision boundary

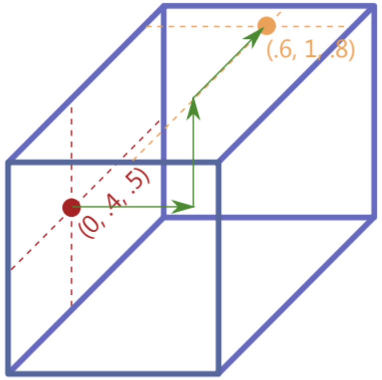
Basics

- Oftentimes convenient to work with feature $x \in \mathrm{R}^d$
- Distances in R^d:

  notation $x(f)$: $x = (x(1), \ldots, x(d))$

  - (popular) Euclidean distance $d_2(x, x') = \sqrt{\sum_{f=1}^{d} \left(x(f) - x'(f)\right)^2}$
  - Manhattan distance $d_1(x, x') = \sum_{f=1}^{d} |x(f) - x'(f)|$

  - If we shift a feature, would the distance change?
  - What about scaling a feature?



- How to extract features as **<u>real values</u>**?
  - Boolean features: {Y, N} -> {0,1}
  - Categorical features: {Red, Blue, Green, Black}
    - Convert to {1, 2, 3, 4}?
    - Better one-hot encoding: (1,0,0,0), .., (0,0,0,1) (IsRed?/isGreen?/isBlue?/IsBlack?)

k-NN classification: pseudocode

- Training is trivial: store the training set
- Test:

Algorithm 3 KNN-PREDICT($\mathbf{D}$, K, $\hat{x}$)

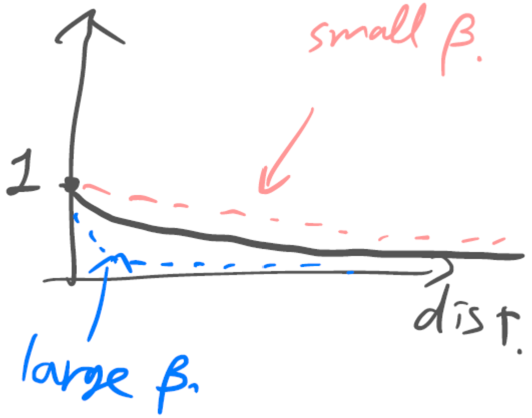
list → $S \leftarrow [\,]$
for $n = 1$ **to** N **do**
append to list → $\quad S \leftarrow S \oplus \langle \mathrm{d}(x_n, \hat{x}), n \rangle$ // store distance to training example n
end for
sort in first coordinate → $S \leftarrow \textsc{sort}(S)$ // put lowest-distance objects first
$\hat{y} \leftarrow 0$
for $k = 1$ **to** K **do**
$\quad \langle dist, n \rangle \leftarrow S_k$ // n this is the kth closest data point
$\quad \hat{y} \leftarrow \hat{y} + y_n$ // vote according to the label for the nth training point
end for
Majority vote of $\{y_i : i \in N(x)\}$ → **return** $\textsc{sign}(\hat{y})$ // return $+1$ if $\hat{y} > 0$ and -1 if $\hat{y} < 0$

- Time complexity (assuming distance calculation takes $O(d)$ time)
  - $O(m\,d\ + m \log m\ + k\ ) = O\big(m(d\ + \log m)\big)$
- Faster nearest neighbor search: k-d trees, locality sensitive hashing

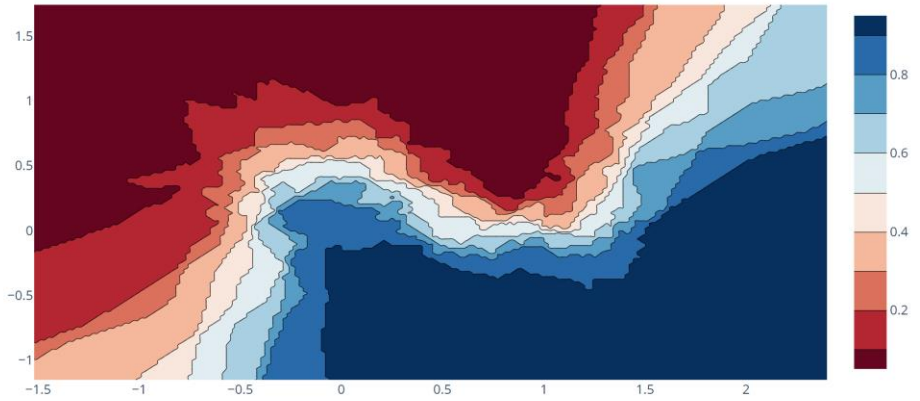
Variations

- Classification
    - Recall the majority vote rule: $\hat{y} = \arg\max_{y \in \{1,\ldots,C\}} \sum_{i \in N(x)} 1\{y_i = y\}$
    - Soft weighting nearest neighbors: $\hat{y} = \arg\max_{y \in \{1,\ldots,C\}} \sum_{i=1}^{m} w_i \, 1\{y_i = y\}$,

      where $w_i \propto \exp(-\beta \, d(x, x_i))$, or $\propto \frac{1}{1+d(x,x_i)^{\beta}}$



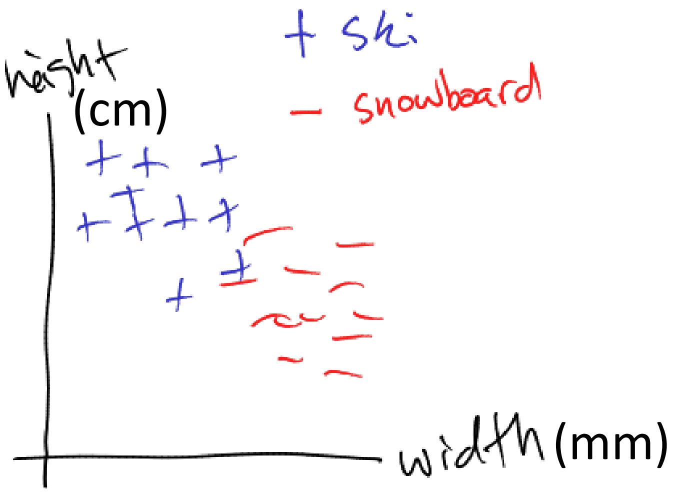
- Class probability estimates
    - $\hat{P}(Y = y \mid x) = \frac{1}{k} \sum_{i \in N(x)} 1\{y_i = y\}$
    - Useful for "classification with rejection"

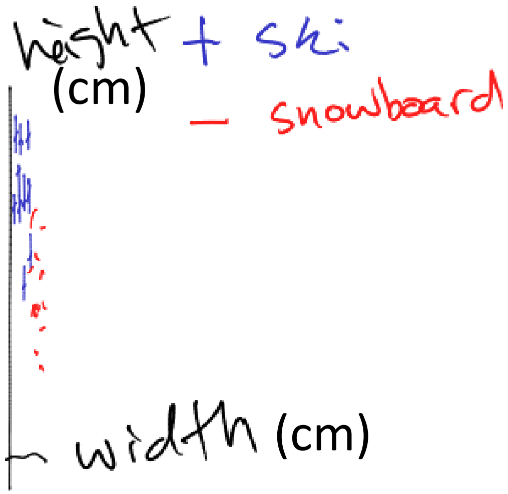


Feature issue 1: scaling

- Features having different scale can be problematic.

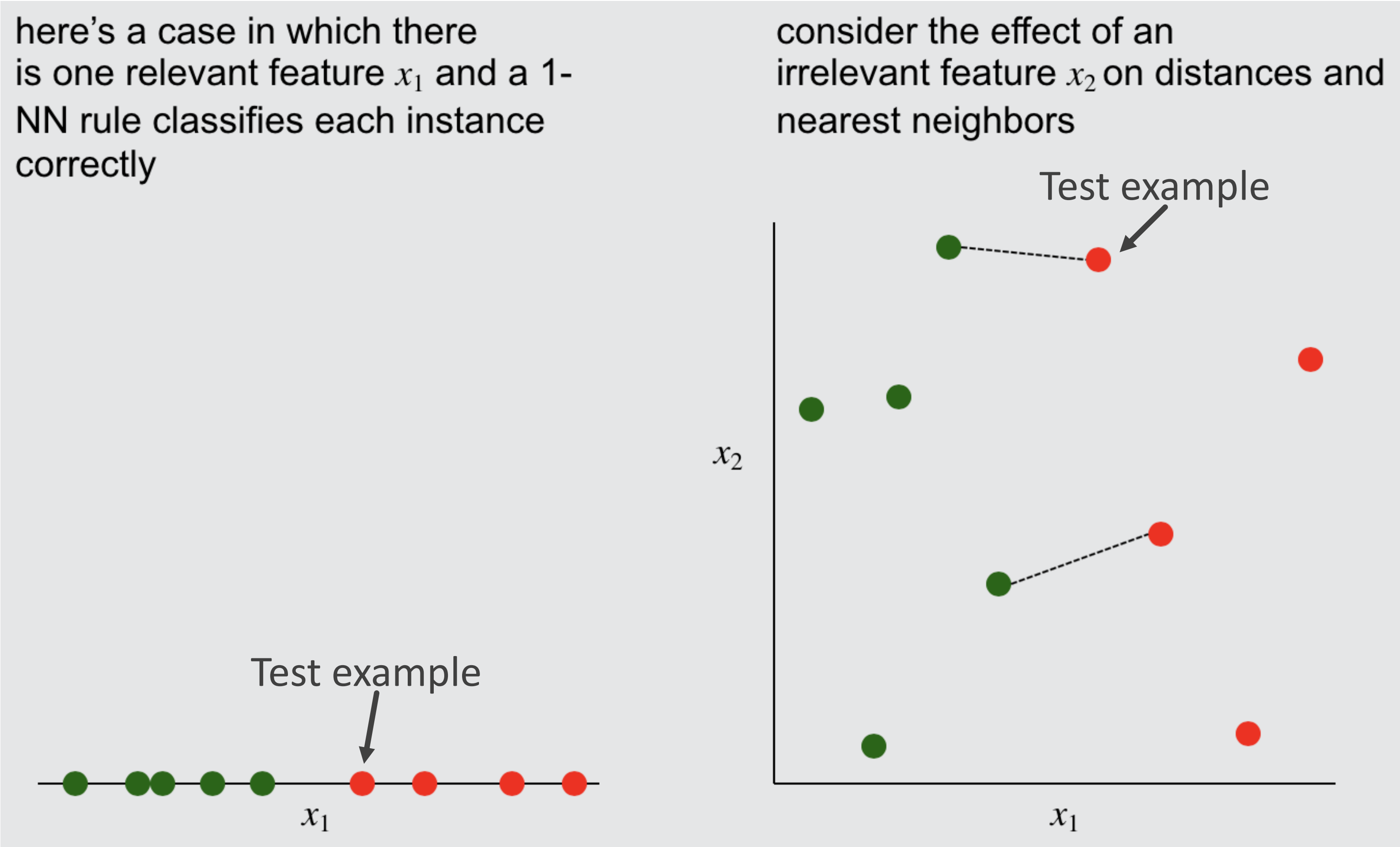
- Ex: ski vs. snowboard classification





- Solution: feature standardization (later in the course)

Feature issue 2: irrelevant features

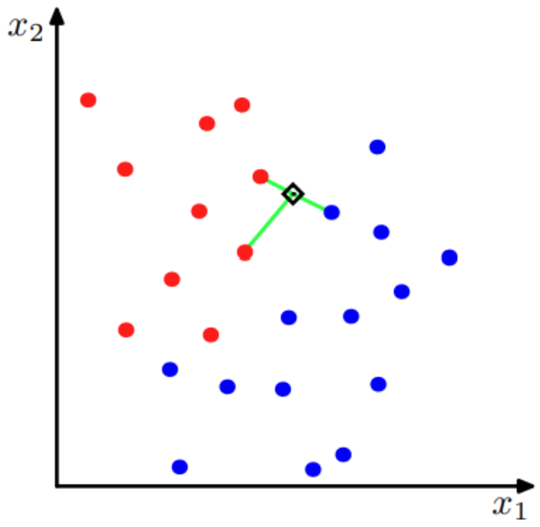


- Recall: how did we deal with these in decision trees?
- Solution: feature selection (later in the course)

Hyperparameter tuning in k-NN

- Hyperparameter: k

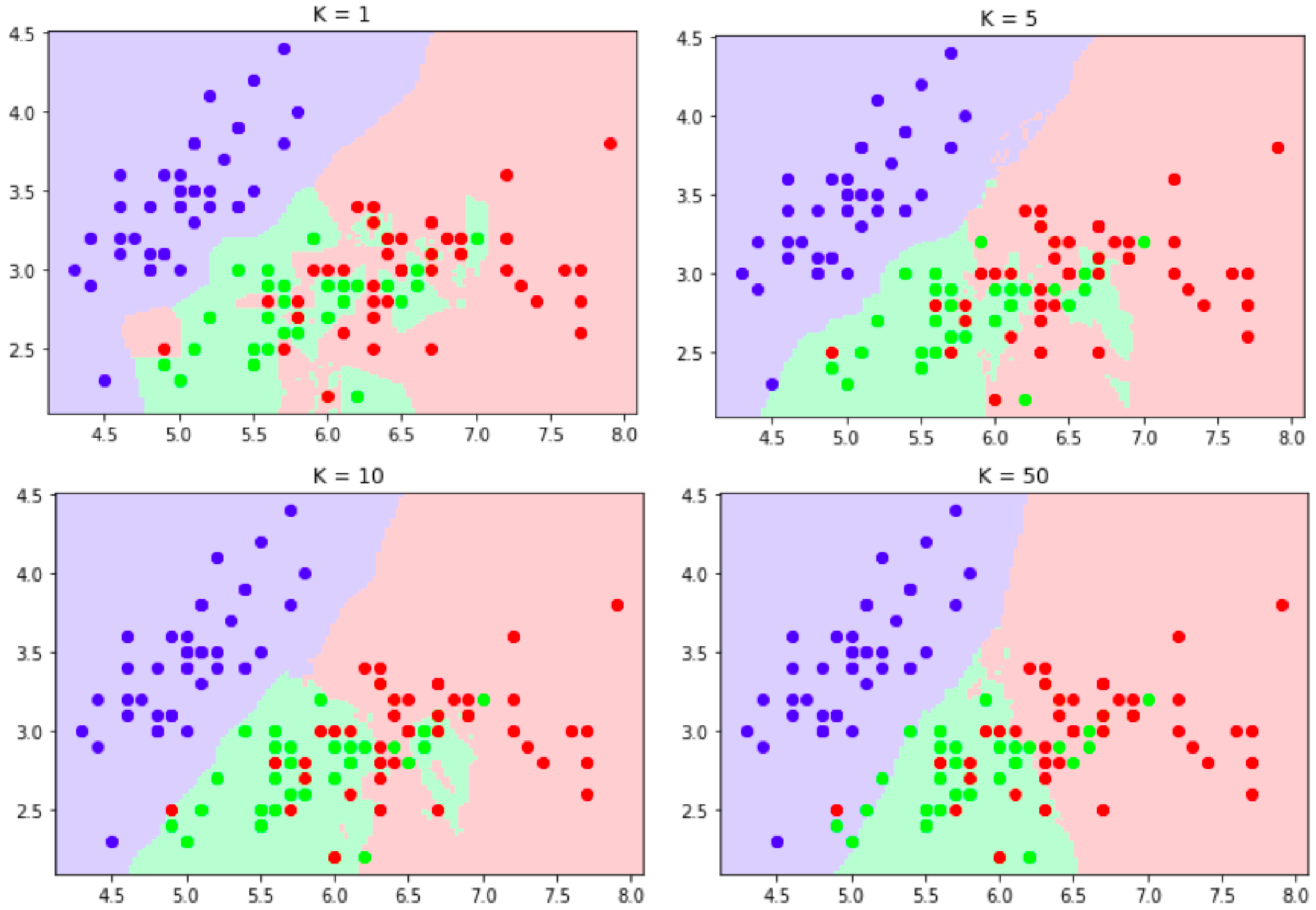
- $k = 1$:
  - Training error = 0, overfitting



- $k = N$:
  - Output a constant (majority class) prediction, underfitting

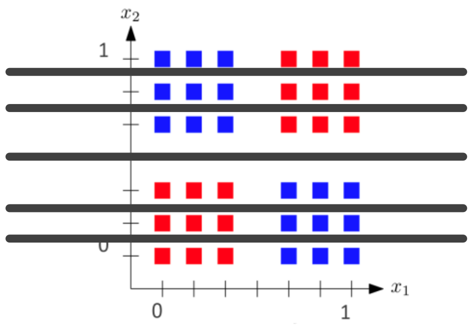
- Can use hold-out validation to choose k

Hyperparameter tuning in k-NN



Comparison (feature $x \in \mathbb{R}^d$)

	Decision Tree	k-NN
• Interpretability	High	Medium (example-based)
• Sensitivity to irrelevant features	Low	High
• training time	$O(\#\text{nodes} \cdot d \cdot (m + m \log m))$ $\le \tilde{O}(d\, m^2)$ (when no two points have the same feature)	0
• test time per example	$O(\text{depth})$	$O(m(d + \log m))$



Next lecture (9/7)

- Linear classification; the Perceptron algorithm

- Assigned reading: CIML Chap. 4