ECE 884 Deep Learning

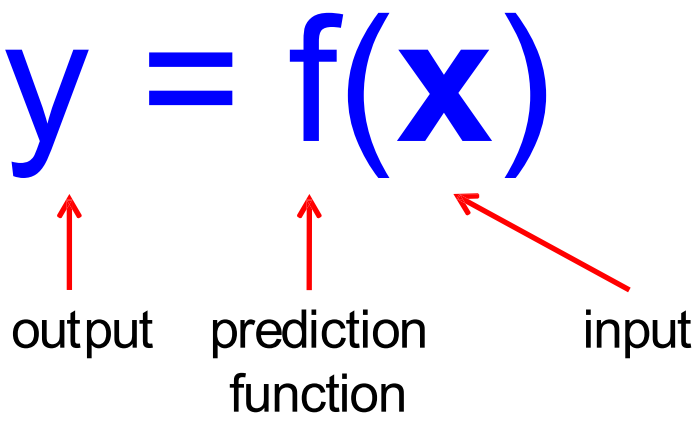
Lecture 4: Nonparametric Model

01/28/2021

Review of last lecture

- Other flavors of machine learning
  - Unsupervised learning
  - Semi-supervised learning
  - Active learning
  - Self-supervised learning
  - Reinforcement learning

Supervised Learning Formulation

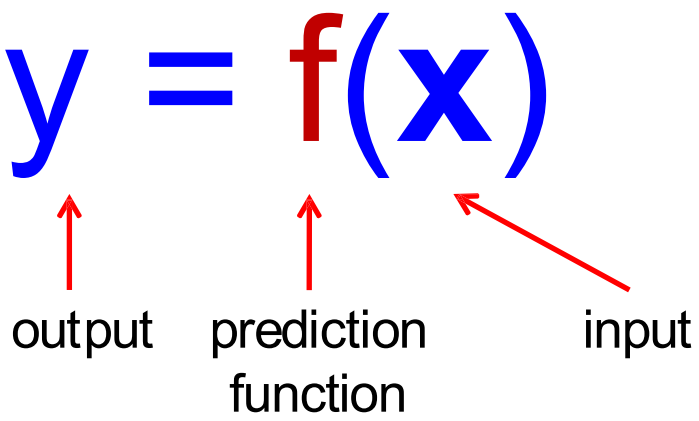


Formulation:

- Given training data: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$,
- Find $y = f(\mathbf{x}) \in$ " using training data
- such that f is correct on test data

Credit: Lazebnik, Liang

Supervised Learning Formulation



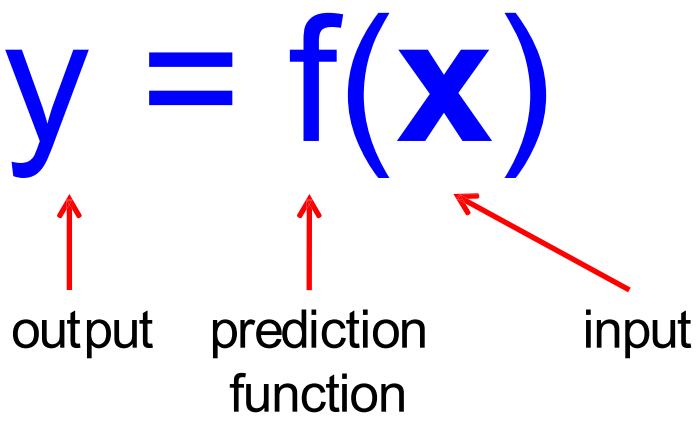
Task#1: function form

Formulation:

- Given training data: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$,
- Find $y = f(\mathbf{x}) \in$ " using training data
- such that f is correct on test data

Credit: Lazebnik, Liang

Supervised Learning Formulation



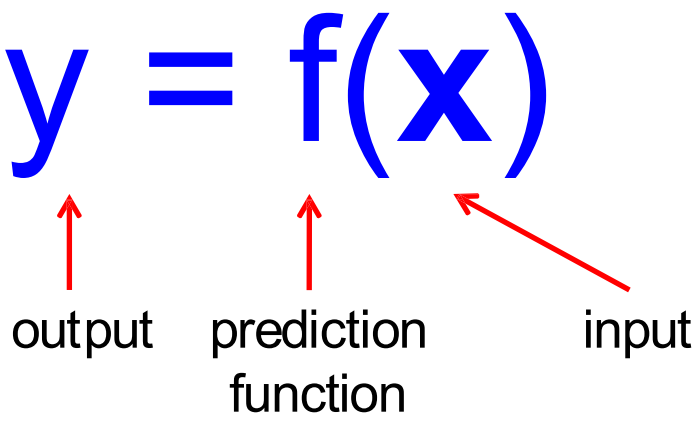
Task#2: how to tell whether f is good or bad?

Formulation:

- Given training data: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$,
- Find $y = f(\mathbf{x}) \in$ " using training data
- such that f is correct on test data

Credit: Lazebnik, Liang

Supervised Learning Formulation



Task#3: how to train f?

Formulation:

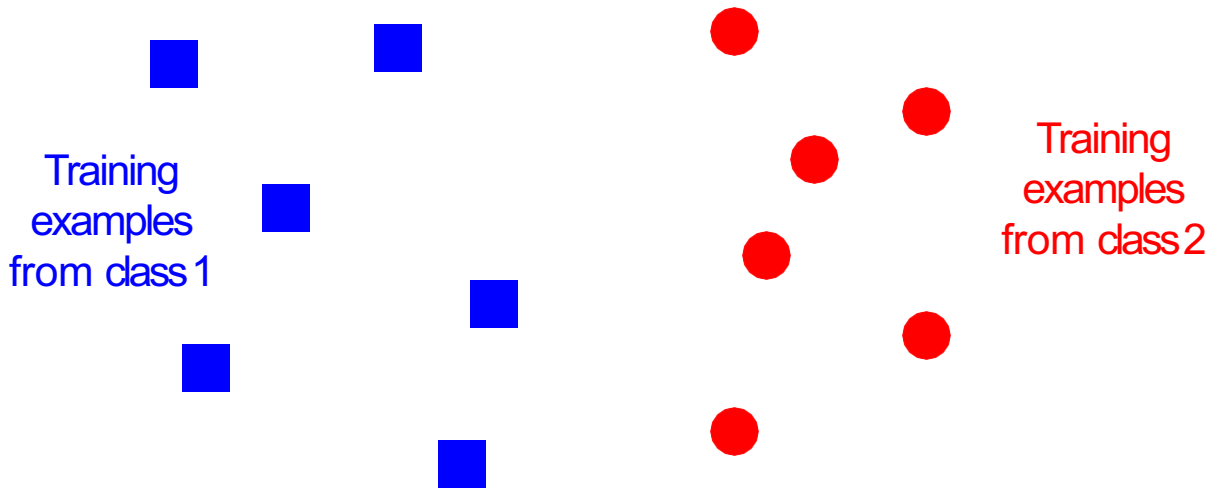
- Given training data: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$,
- Find $y = f(\mathbf{x}) \in$ " using training data
- such that f is correct on test data

Credit: Lazebnik, Liang

Today’s lecture

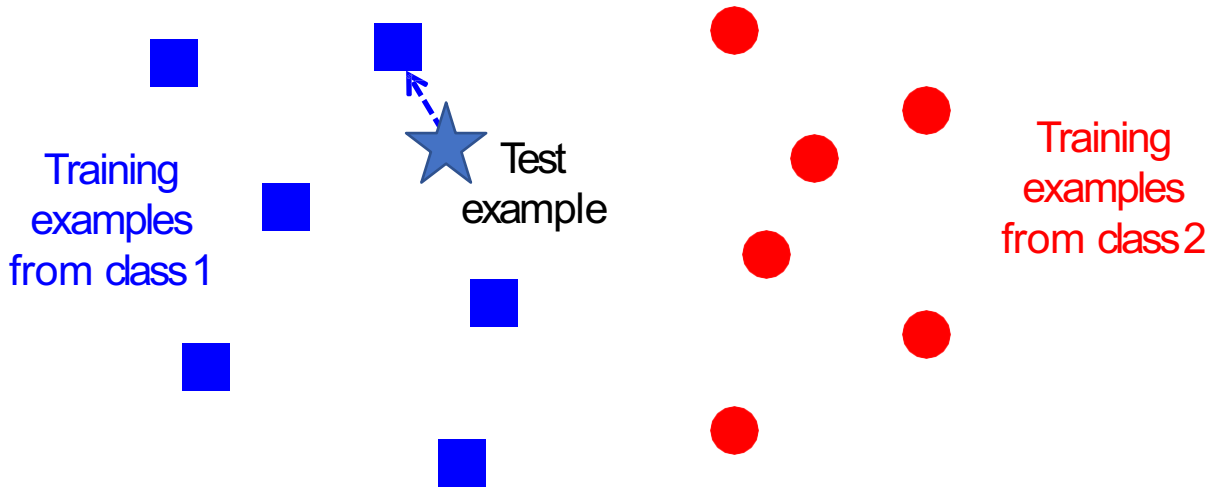
- Function form#1: nonparametric models

Nearest Neighbor Classifier (classification model)



Credit: Lazebnik

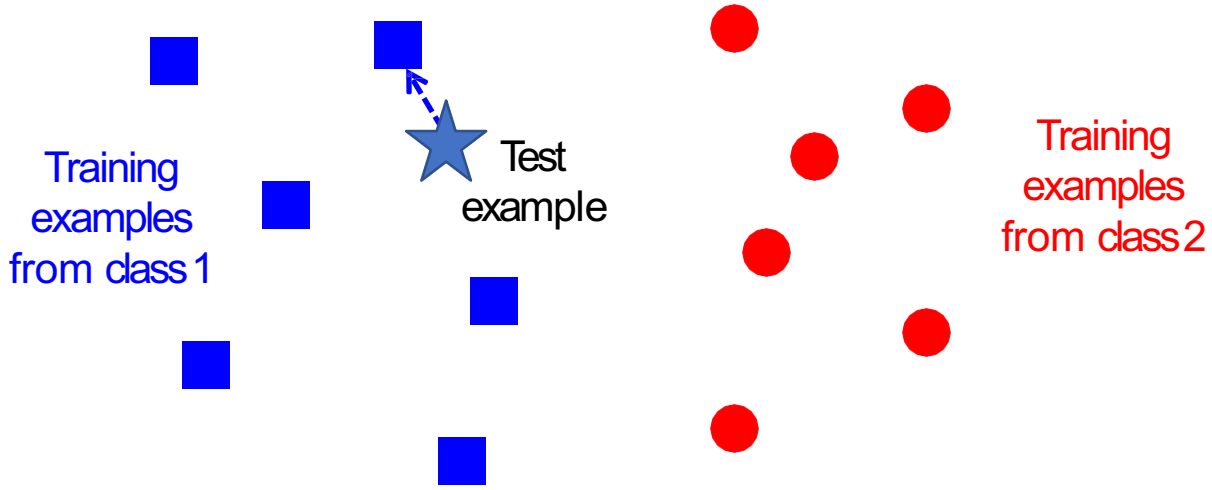
Nearest Neighbor Classifier (classification model)



f(x) = ?

Credit: Lazebnik

Nearest Neighbor Classifier (classification model)



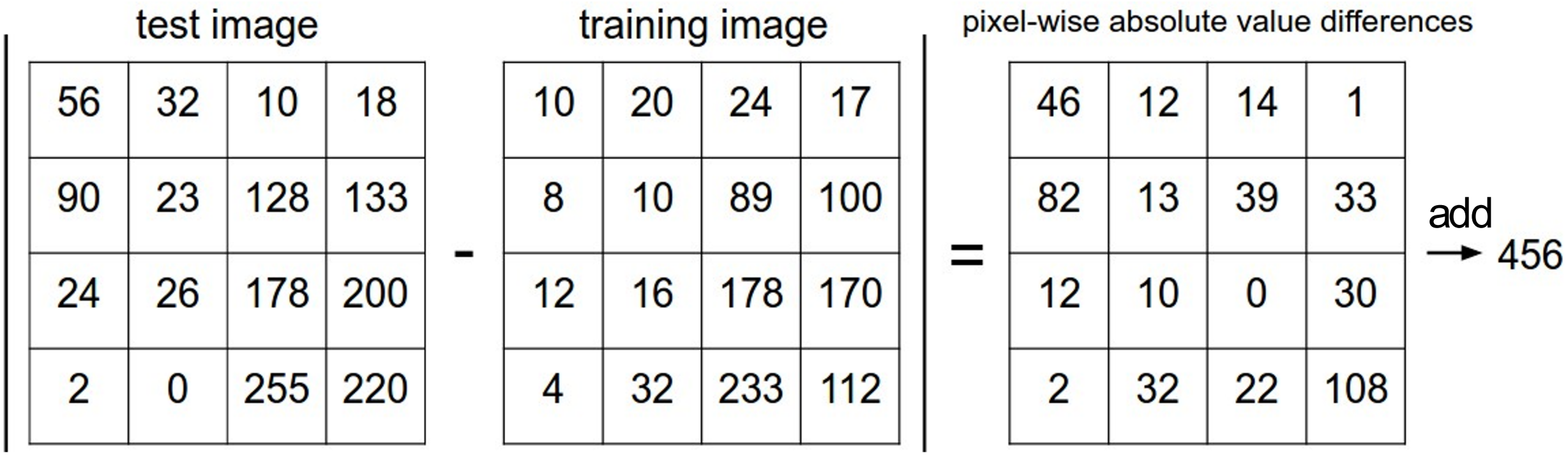
f(x) = label of the training example nearest to x

- All we need is a distance metric for our inputs
- No training required!

Credit: Lazebnik

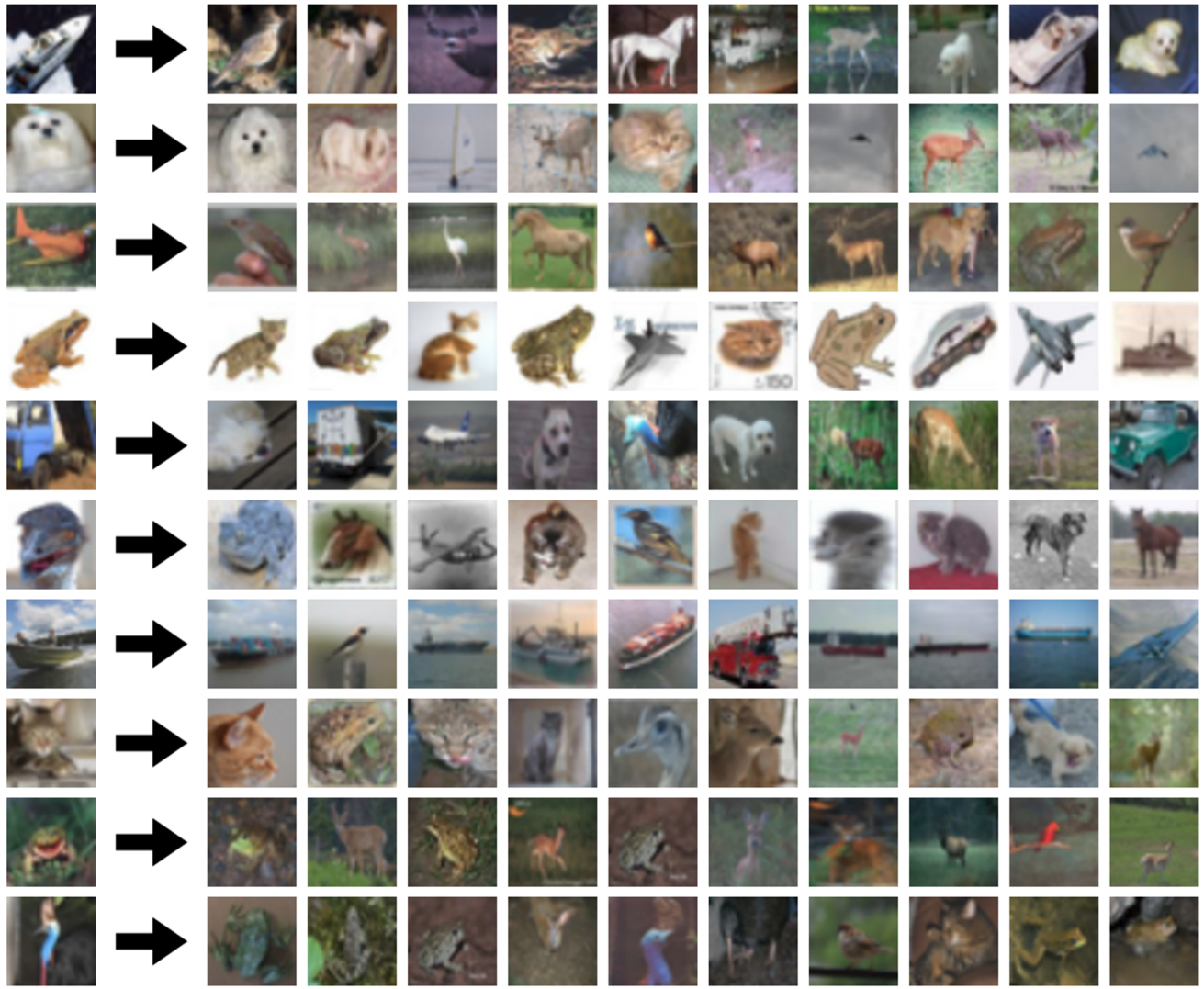
Distance Metric to compare images

L1 distance: $$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



Credit: Justin

What does this look like?

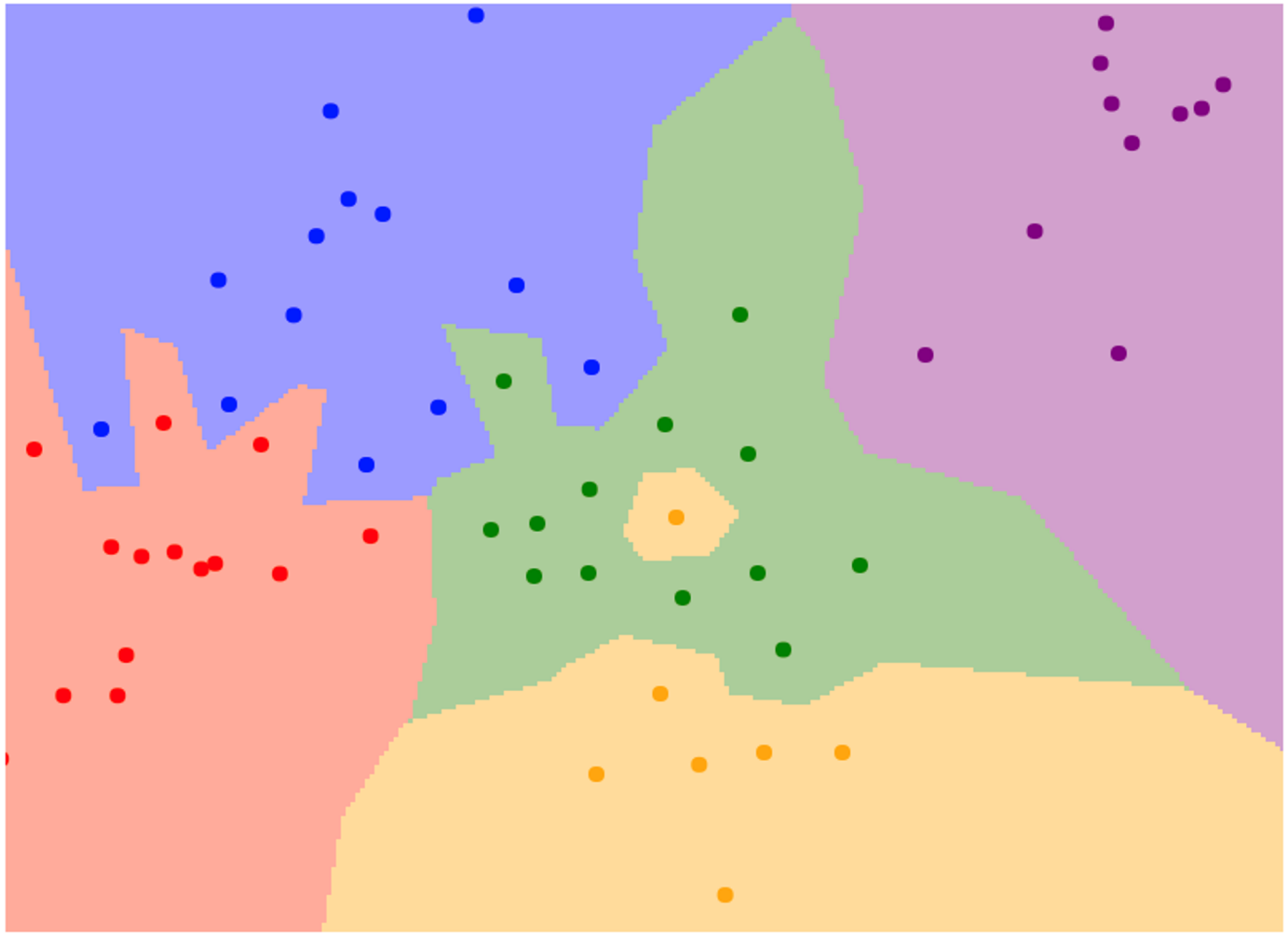


Credit: Justin

What does this look like?

Credit: Justin

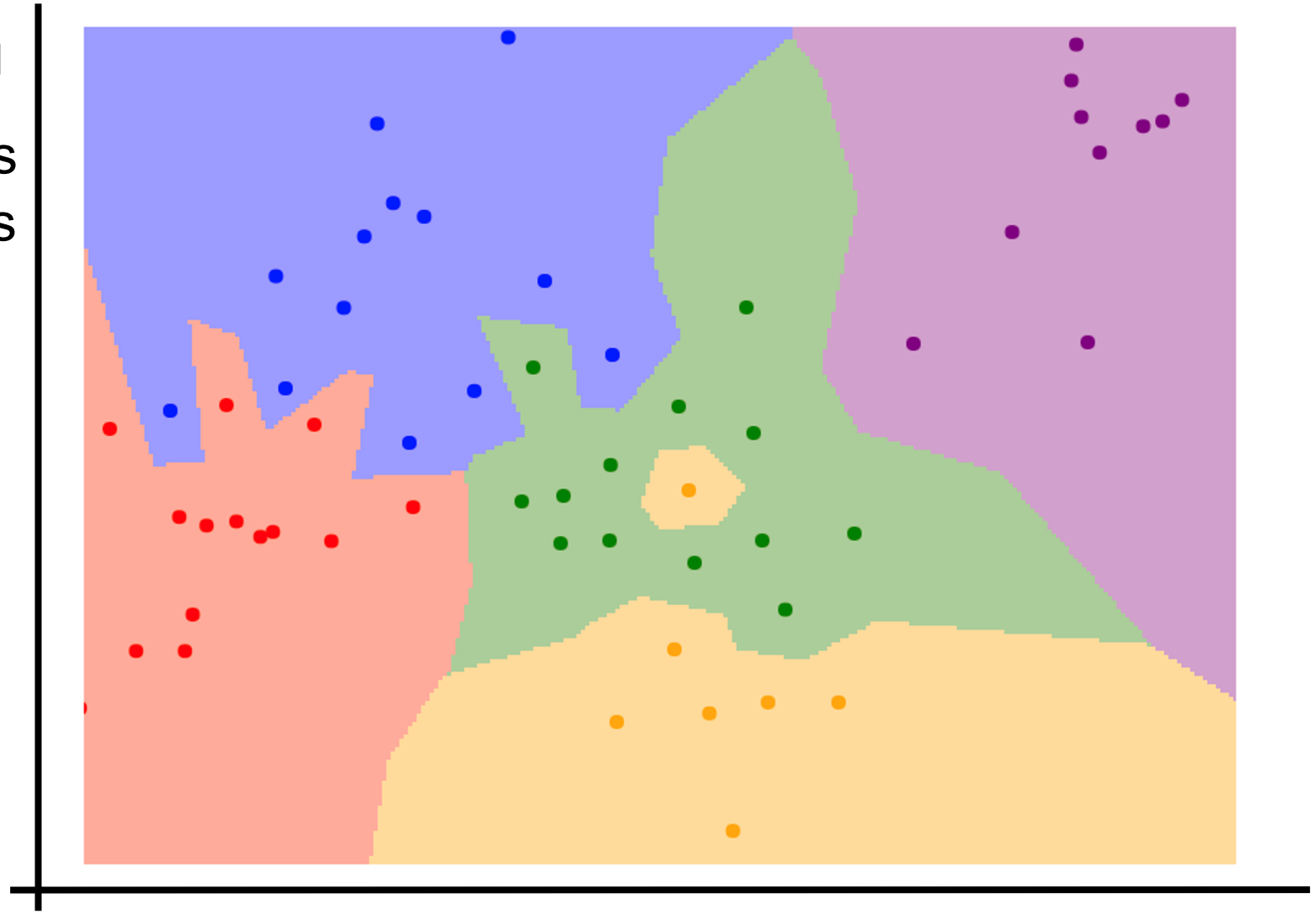
Nearest Neighbor Decision Boundaries



Credit: Justin

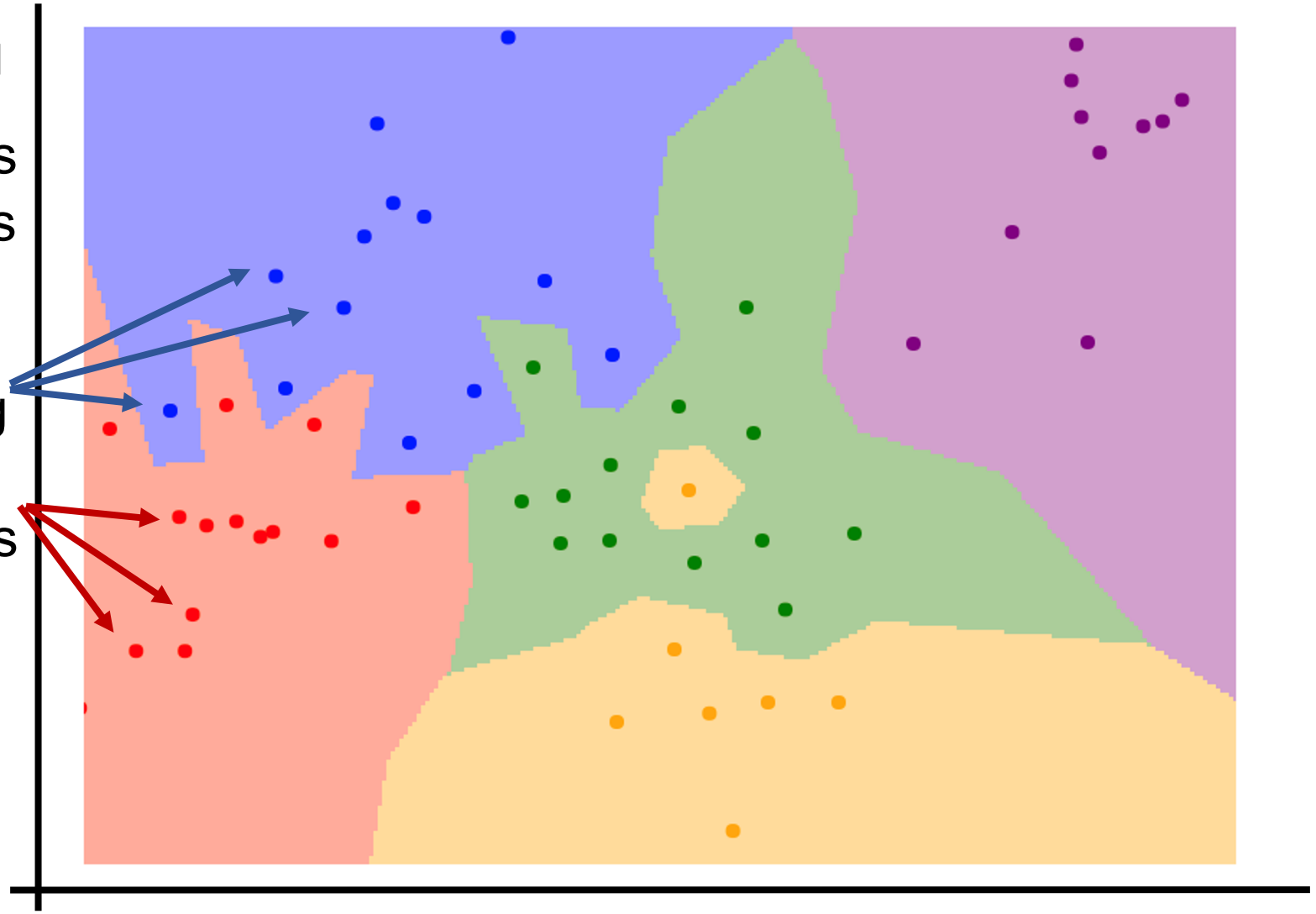
Nearest Neighbor Decision Boundaries

Nearest neighbors in two dimensions



Credit: Justin

Nearest Neighbor Decision Boundaries

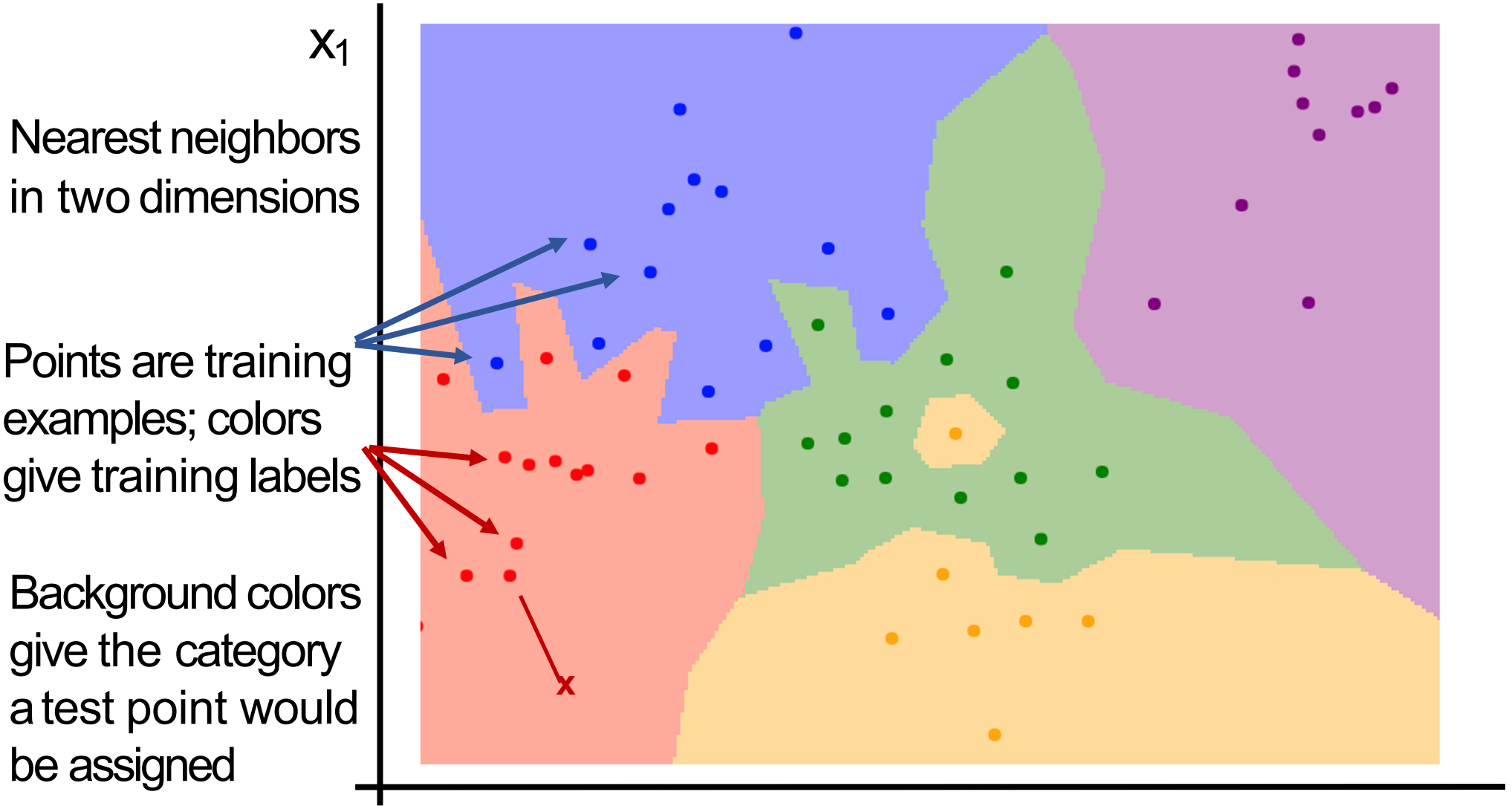


Nearest neighbors in two dimensions

Points are training examples; colors give training labels

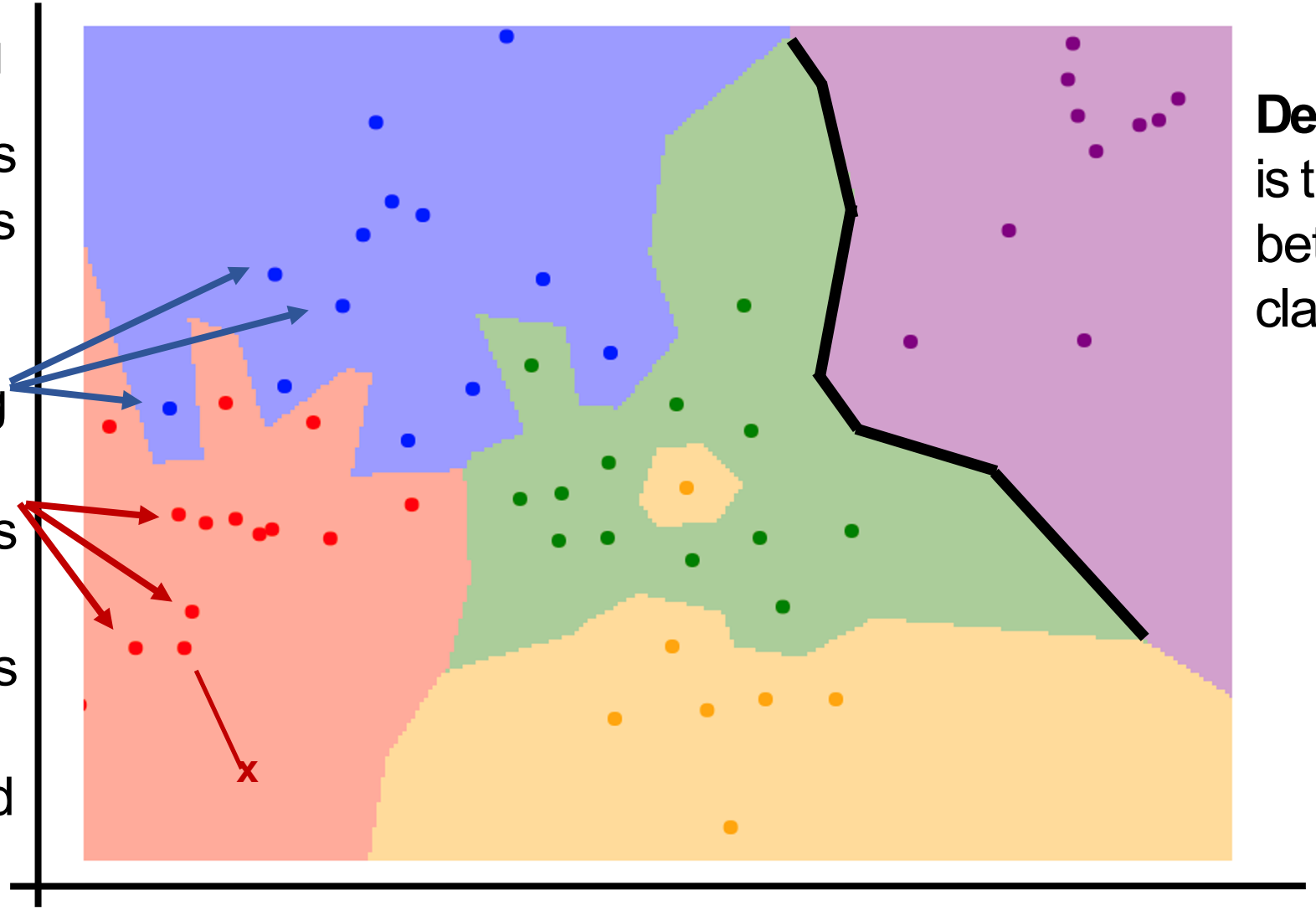
Credit: Justin

Nearest Neighbor Decision Boundaries



Credit: Justin

Nearest Neighbor Decision Boundaries



Nearest neighbors in two dimensions

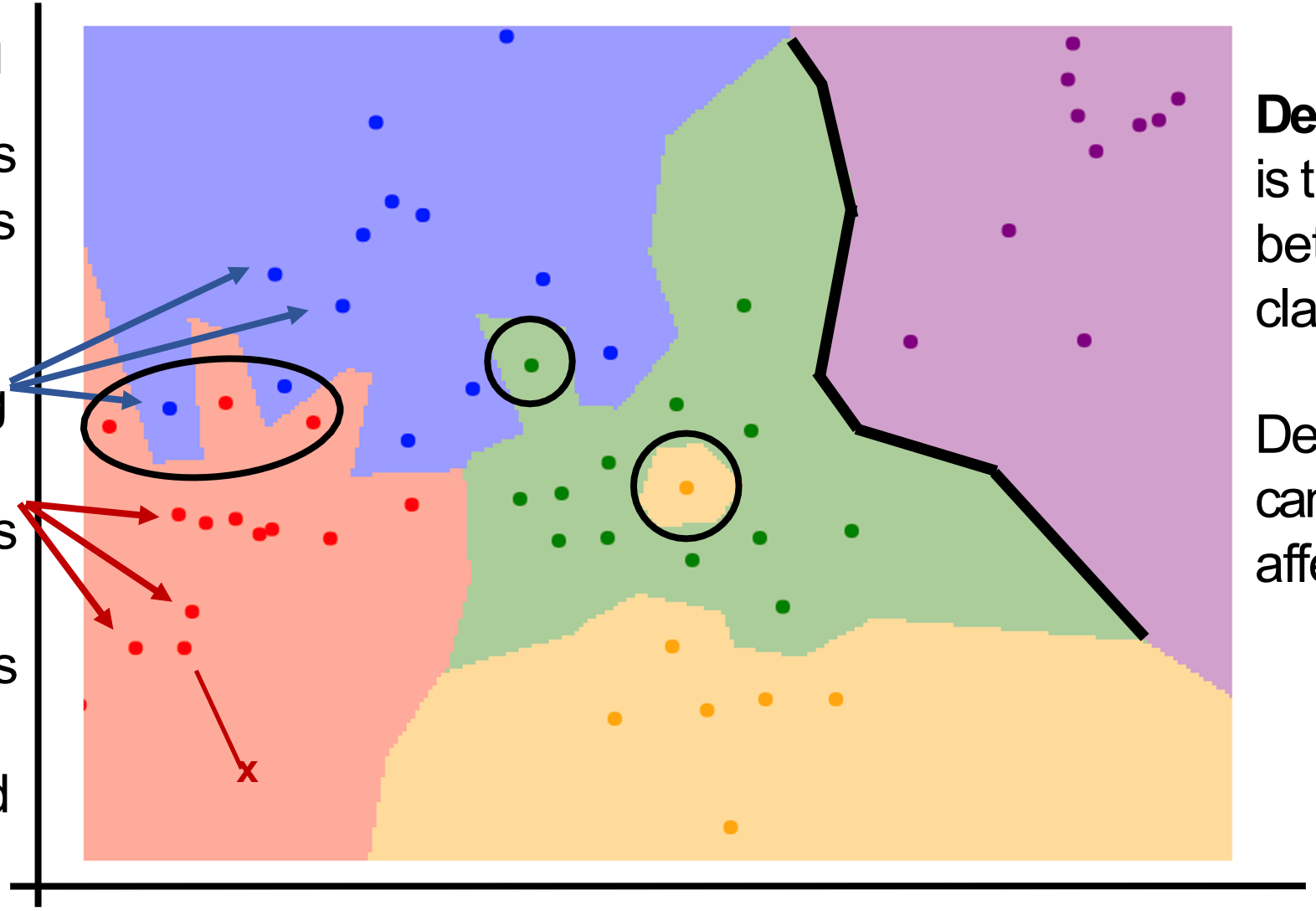
Points are training examples; colors give training labels

Background colors give the category a test point would be assigned

Decision boundary is the boundary between two classification regions

Credit: Justin

Nearest Neighbor Decision Boundaries



Nearest neighbors in two dimensions

Points are training examples; colors give training labels

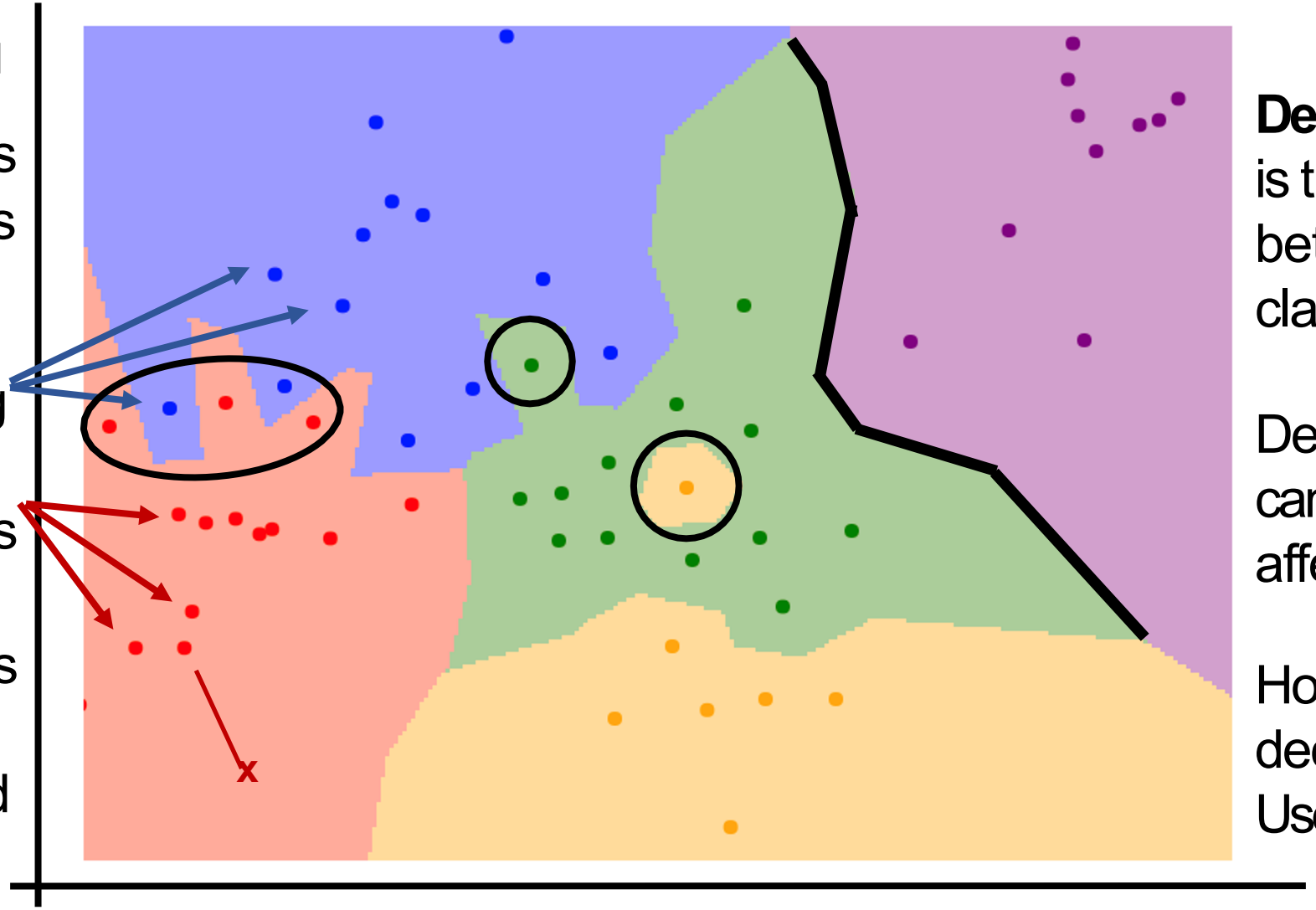
Background colors give the category a test point would be assigned

Decision boundary is the boundary between two classification regions

Decision boundaries can be noisy; affected by outliers

Credit: Justin

Nearest Neighbor Decision Boundaries



Nearest neighbors in two dimensions

Points are training examples; colors give training labels

Background colors give the category a test point would be assigned

Decision boundary is the boundary between two classification regions

Decision boundaries can be noisy; affected by outliers

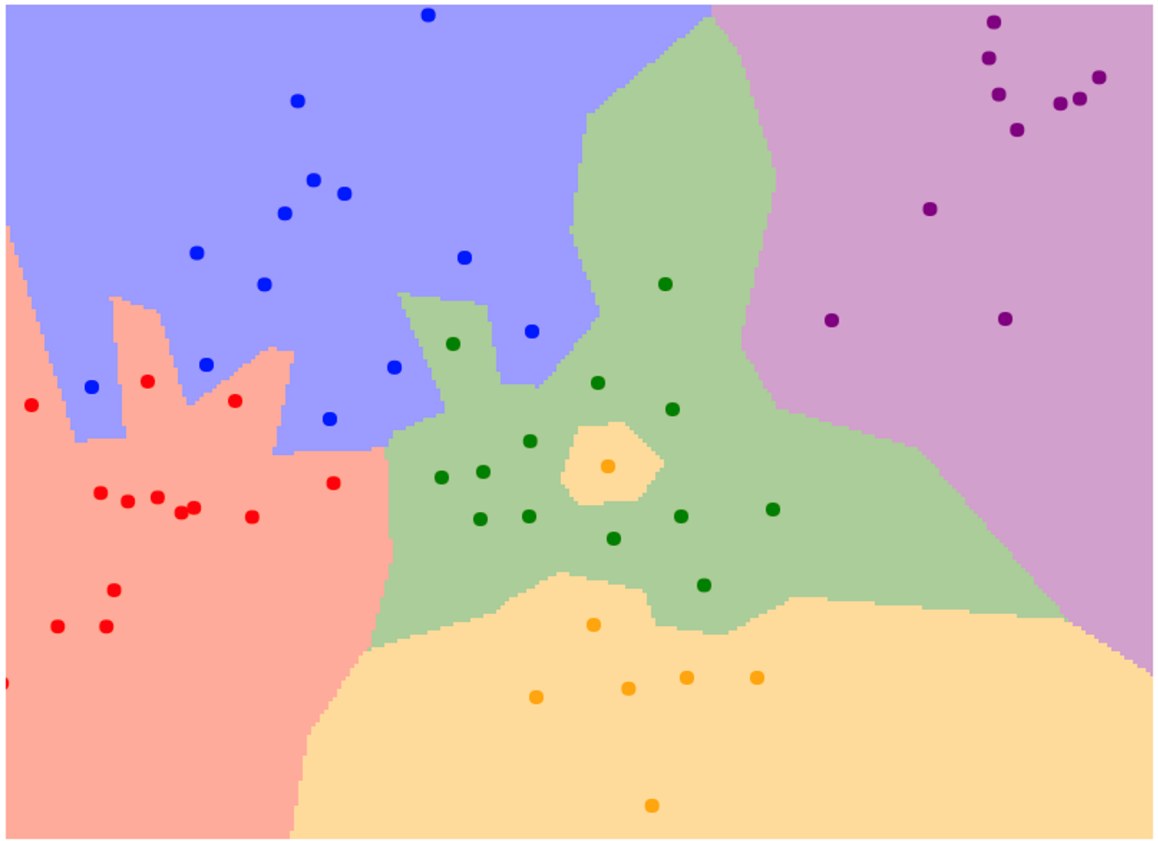
How to smooth out decision boundaries? Use more neighbors!

Credit: Justin

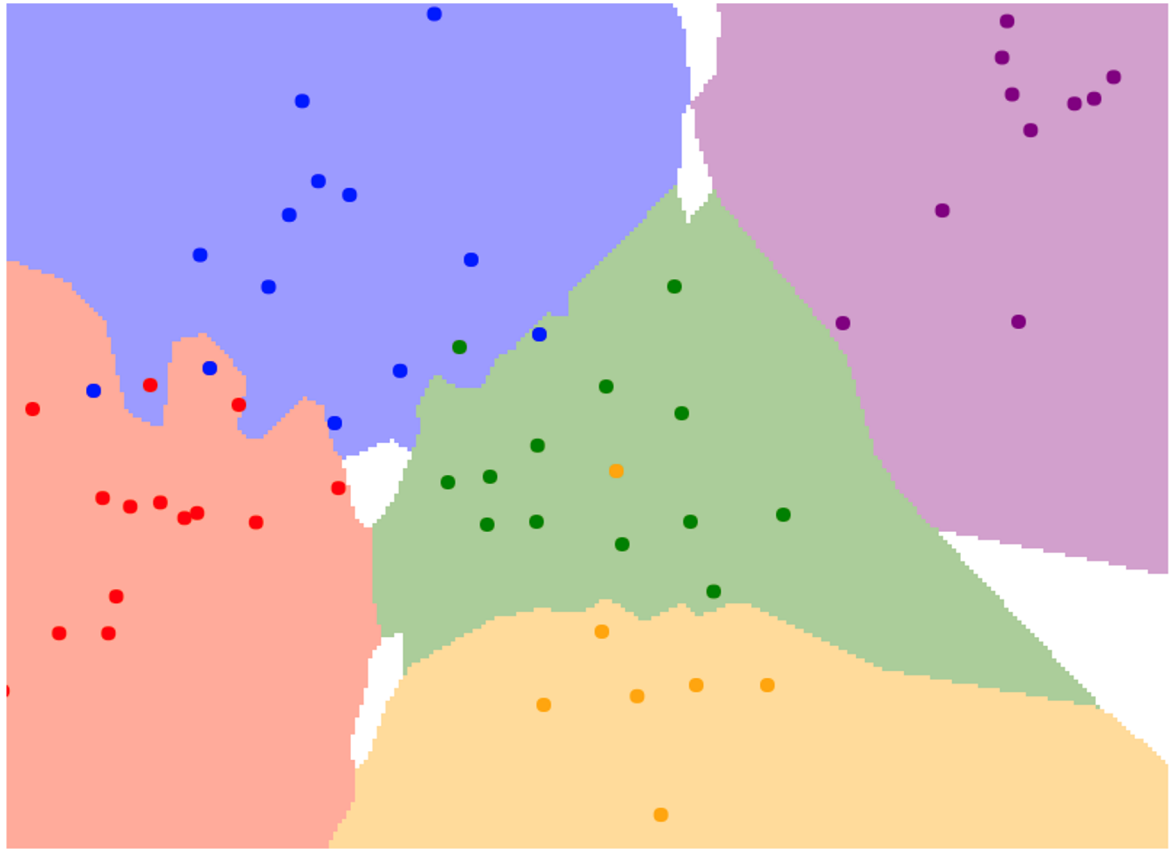
K-Nearest Neighbors

Instead of copying label from nearest neighbor, take **majority vote** from K closest points

K = 1



K = 3

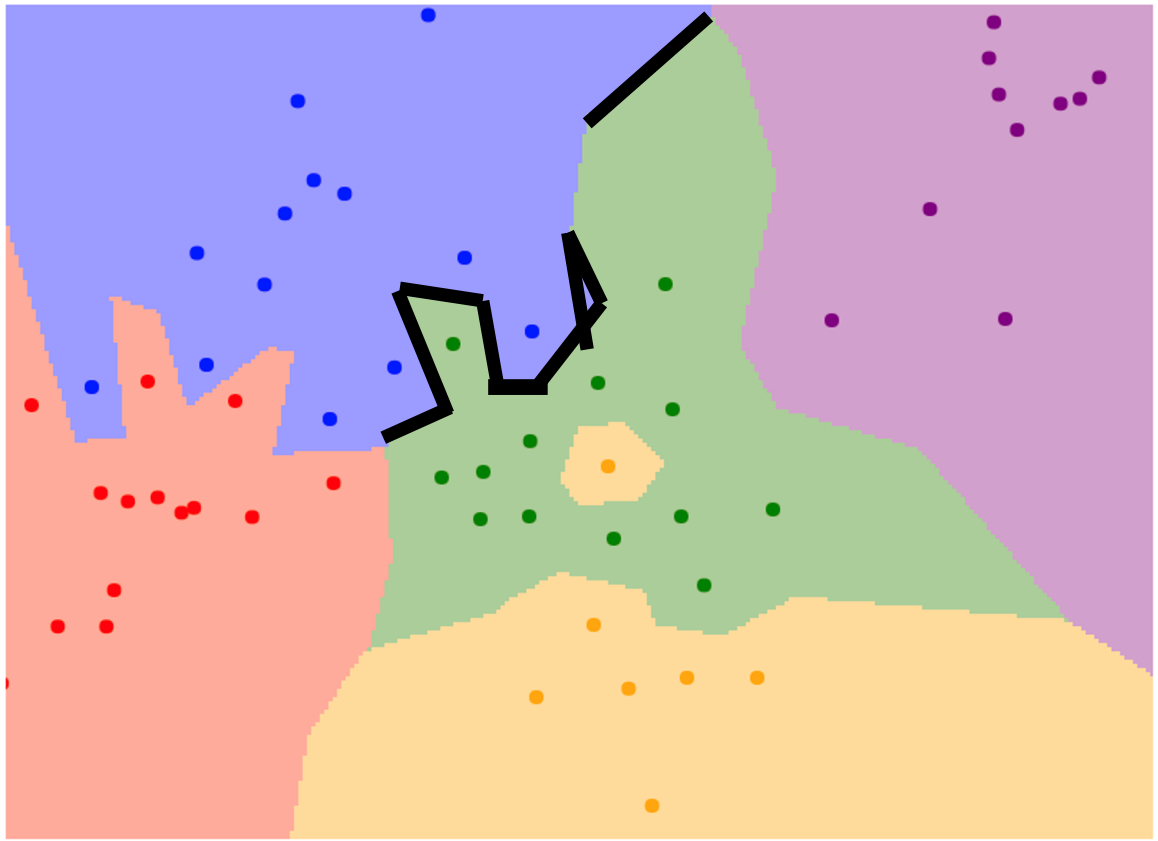


Credit: Justin

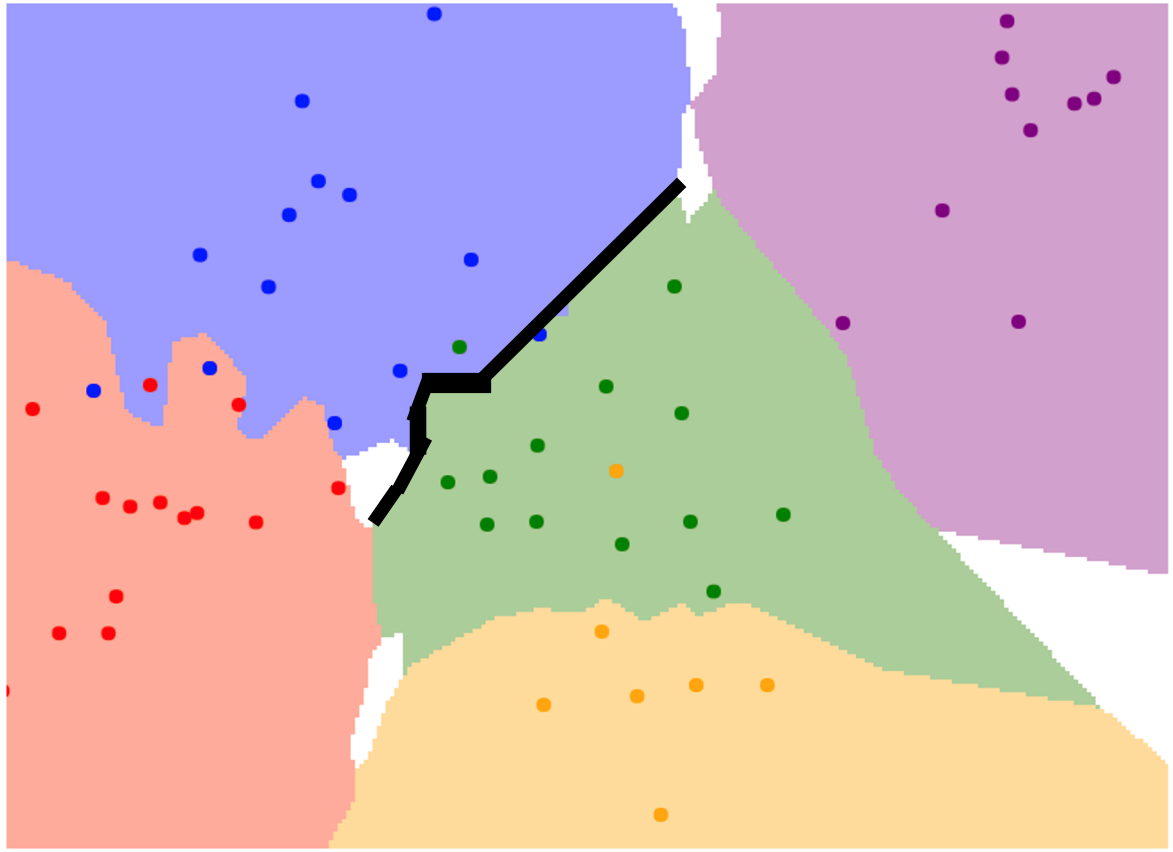
K-Nearest Neighbors

Using more neighbors helps smooth out rough decision boundaries

K = 1



K = 3



Credit: Justin

K-Nearest Neighbors: Distance Metric

With the right choice of distance metric, we can apply K-Nearest Neighbor to any type of data!

Credit: Justin

K-Nearest Neighbors: Distance Metric

With the right choice of distance metric, we can apply K-Nearest Neighbor to any type of data!

Example: Compare research papers using tf-idf similarity

Mesh R-CNN
Georgia Gkioxari, Jitendra Malik, Justin Johnson
6/6/2019 cs.CV

[illegible]02739v1 pdf
show similar discuss

Rapid advances in 2D perception have led to systems that accurately detect objects in real-world images. However, these systems make predictions in 2D, ignoring the 3D structure of the world. Concurrently, advances in 3D shape prediction have mostly focused on synthetic benchmarks and isolated objects. We unify advances in these two areas. We propose a system that detects objects in real-world images and produces a triangle mesh giving the full 3D shape of each detected object. Our system, called Mesh R-CNN, augments Mask R-CNN with a mesh prediction branch that outputs meshes with varying topological structure by first predicting coarse voxel representations which are converted to meshes and refined with a graph convolution network operating over the mesh's vertices and edges. We validate our mesh prediction branch on ShapeNet, where we outperform prior work on single-image shape prediction. We then deploy our full Mesh R-CNN system on Pix3D, where we jointly detect objects and predict their 3D shapes.

http://www.arxiv-sanity.com/search?q=mesh+r-cnn

Credit: Justin

K-Nearest Neighbors: Distance Metric

Most similar papers:

Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era
1906.06543v2 pdf
show similar | discuss
Xian-Feng Han, Hamid Laga, Mohammed Bennamoun
6/18/2019 (v1: 6/15/2019) cs.CV | cs.CG | cs.GR | cs.LG

3D reconstruction is a longstanding ill-posed problem, which has been explored for decades by the computer vision, computer graphics, and machine learning communities. Since 2015, image-based 3D reconstruction using convolutional neural networks (CNN) has attracted increasing interest and demonstrated an impressive performance. Given this new era of rapid evolution, this article provides a comprehensive survey of the recent developments in this field. We focus on the works which use deep learning techniques to estimate the 3D shape of generic objects either from a single or multiple RGB images. We organize the literature based on the shape representations, the network architectures, and the training mechanisms they use. While this survey is intended for methods which reconstruct generic objects, we also review some of the recent works which focus on specific object classes such as human body shapes and faces. We provide an analysis and comparison of the performance of some key papers, summarize some of the open problems in this field, and discuss promising directions for future research.

Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images
1804.01654v2 pdf
show similar | discuss
Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, Yu-Gang Jiang
8/3/2018 (v1: 4/5/2018) cs.CV

We propose an end-to-end deep learning architecture that produces a 3D shape in triangular mesh from a single color image. Limited by the nature of deep neural network, previous methods usually represent a 3D shape in volume or point cloud, and it is non-trivial to convert them to the more ready-to-use mesh model. Unlike the existing methods, our network represents 3D mesh in a graph-based convolutional neural network and produces correct geometry by progressively deforming an ellipsoid, leveraging perceptual features extracted from the input image. We adopt a coarse-to-fine strategy to make the whole deformation procedure stable, and define various of mesh related losses to capture properties of different levels to guarantee visually appealing and physically accurate 3D geometry. Extensive experiments show that our method not only qualitatively produces mesh model with better details, but also achieves higher 3D shape estimation accuracy compared to the state-of-the-art.

Pixel2Mesh++: Multi-View 3D Mesh Generation via Deformation
1908.01491v2 pdf
show similar | discuss
Chao Wen, Yinda Zhang, Zhuwen Li, Yanwei Fu
8/16/2019 (v1: 8/5/2019) cs.CV
Accepted by ICCV 2019

We study the problem of shape generation in 3D mesh representation from a few color images with known camera poses. While many previous works learn to hallucinate the shape directly from priors, we resort to further improving the shape quality by leveraging cross-view information with a graph convolutional network. Instead of building a direct mapping function from images to 3D shape, our model learns to predict series of deformations to improve a coarse shape iteratively. Inspired by traditional multiple view geometry methods, our network samples nearby area around the initial mesh's vertex locations and reasons an optimal deformation using perceptual feature statistics built from multiple input images. Extensive experiments show that our model produces accurate 3D shape that are not only visually plausible from the input perspectives, but also well aligned to arbitrary viewpoints. With the help of physically driven architecture, our model also exhibits generalization capability across different semantic categories, number of input images, and quality of mesh initialization.

GEOMetrics: Exploiting Geometric Structure for Graph-Encoded Objects
1901.11461v1 pdf
show similar | discuss
Edward J. Smith, Scott Fujimoto, Adriana Romero, David Meger
1/31/2019 cs.CV
18 pages

Mesh models are a promising approach for encoding the structure of 3D objects. Current mesh reconstruction systems predict uniformly distributed vertex locations of a predetermined graph through a series of graph convolutions, leading to compromises with respect to performance or resolution. In this paper, we argue that the graph representation of geometric objects allows for additional structure, which should be leveraged for enhanced reconstruction. Thus, we propose a system which properly benefits from the advantages of the geometric structure of graph encoded objects by introducing (1) a graph convolutional update preserving vertex information; (2) an adaptive splitting heuristic allowing detail to emerge; and (3) a training objective operating both on the local surfaces defined by vertices as well as the global structure defined by the mesh. Our proposed method is evaluated on the task of 3D object reconstruction from images with the ShapeNet dataset, where we demonstrate state of the art performance, both visually and numerically, while having far smaller space requirements by generating adaptive meshes

Credit: Justin

Hyperparameters

What is the best value of **K** to use?
What is the best **distance metric** to use?

These are examples of **hyperparameters**: choices about our learning algorithm that we don't learn from the training data; instead we set them at the start of the learning process

Credit: Justin

Hyperparameters

What is the best value of **K** to use?
What is the best **distance metric** to use?

These are examples of **hyperparameters**: choices about our learning algorithm that we don't learn from the training data; instead we set them at the start of the learning process

Very problem-dependent. In general need to try them all and see what works best for our data / task.

Credit: Justin

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

Credit: Justin

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

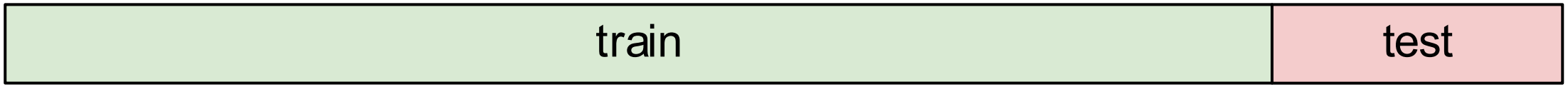
Credit: Justin

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data



Credit: Justin

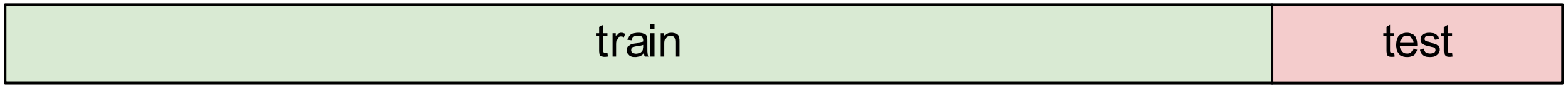
Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data



Credit: Justin

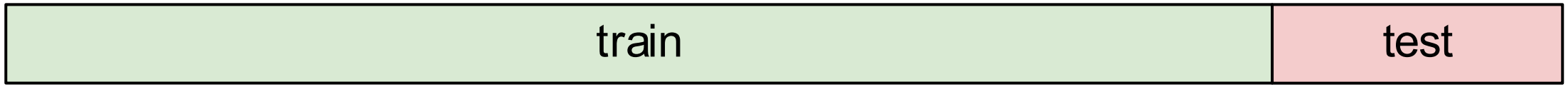
Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

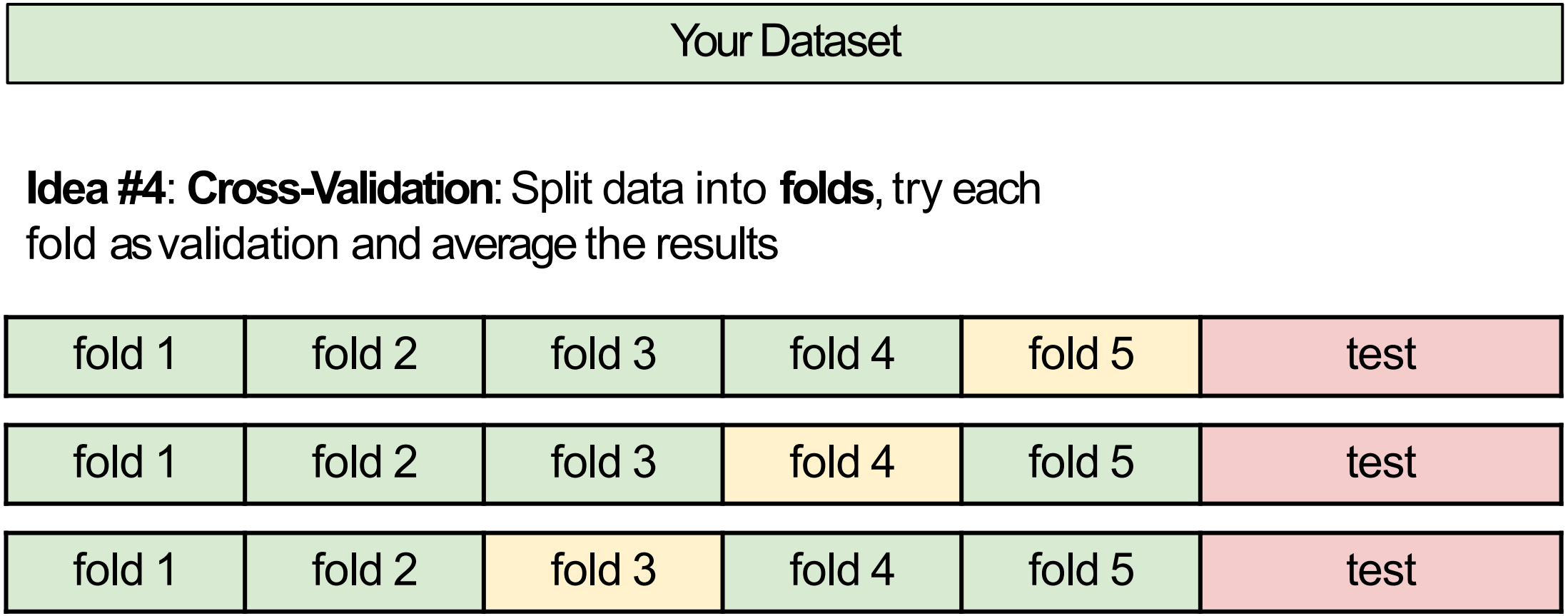


Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

Credit: Justin

Setting Hyperparameters



Idea #4: **Cross-Validation**: Split data into **folds**, try each fold as validation and average the results

Useful for small datasets, but (unfortunately) not used too frequently in deep learning

Credit: Justin

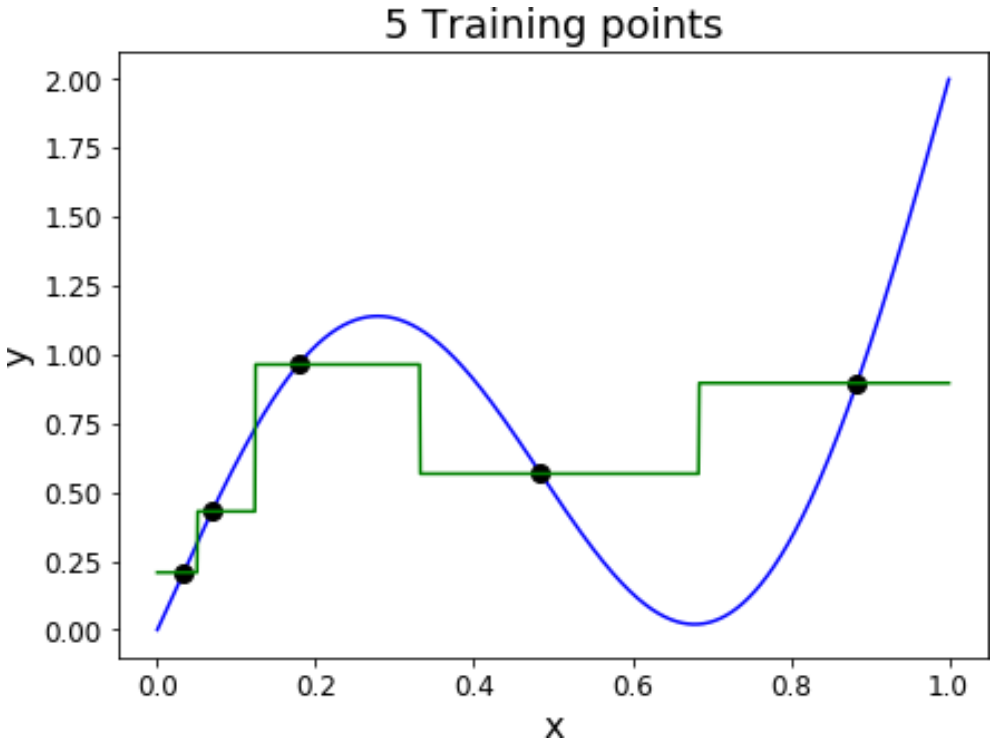
K-Nearest Neighbor: Universal Approximation

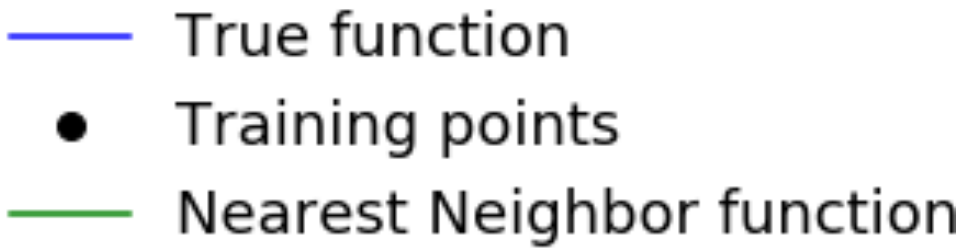
As the number of training samples goes to infinity, nearest neighbor can represent any(*) function!

Credit: Justin

K-Nearest Neighbor: Universal Approximation

As the number of training samples goes to infinity, nearest neighbor can represent any(*) function!

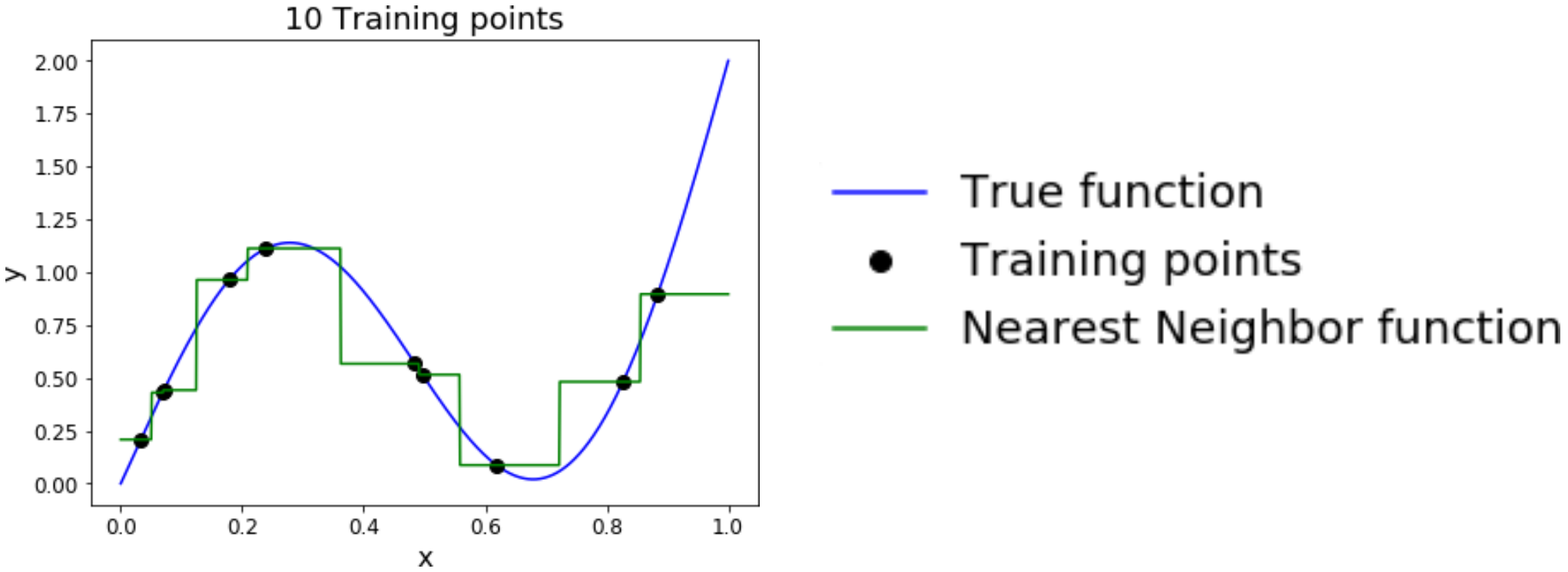




Credit: Justin

K-Nearest Neighbor: Universal Approximation

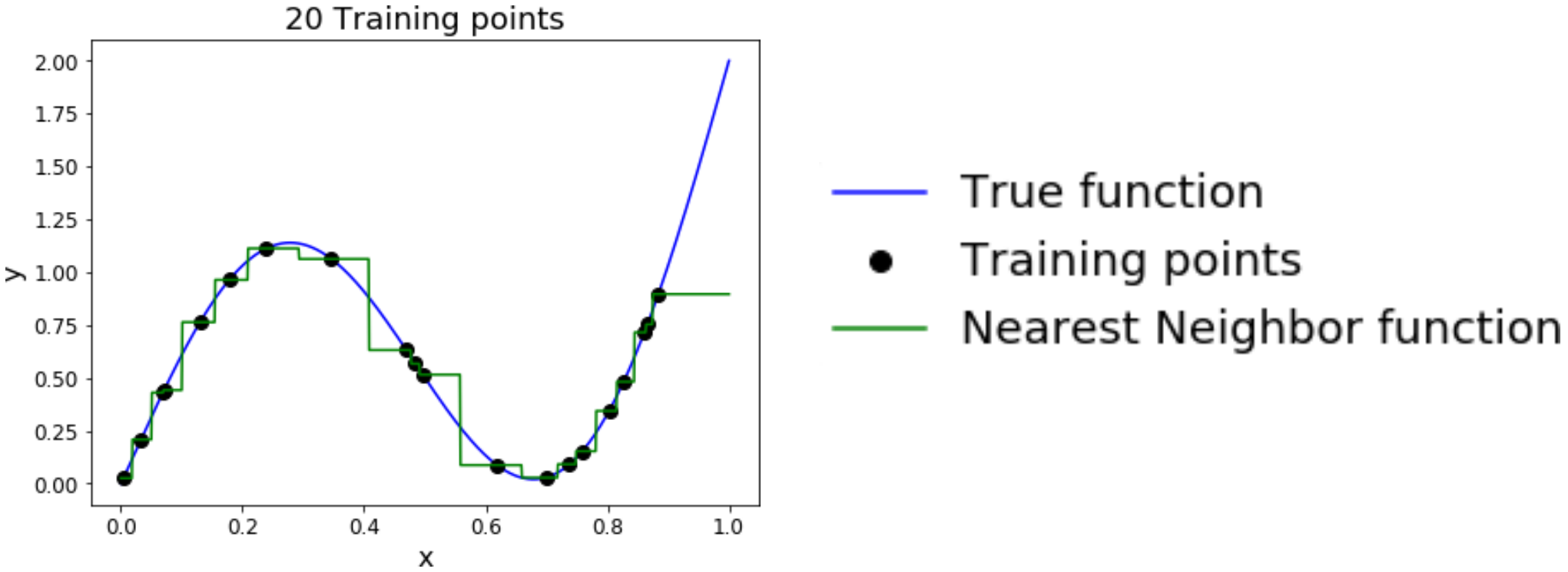
As the number of training samples goes to infinity, nearest neighbor can represent any(*) function!



Credit: Justin

K-Nearest Neighbor: Universal Approximation

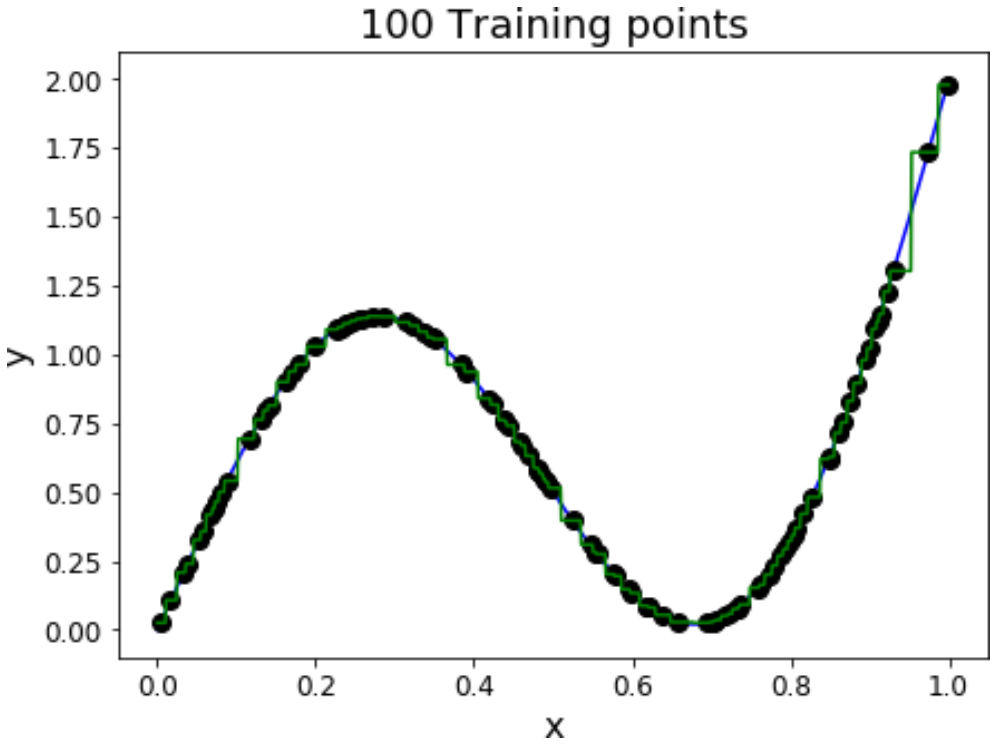
As the number of training samples goes to infinity, nearest neighbor can represent any(*) function!

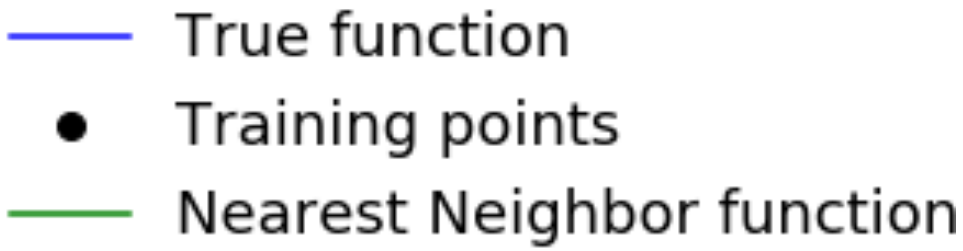


Credit: Justin

K-Nearest Neighbor: Universal Approximation

As the number of training samples goes to infinity, nearest neighbor can represent any(*) function!

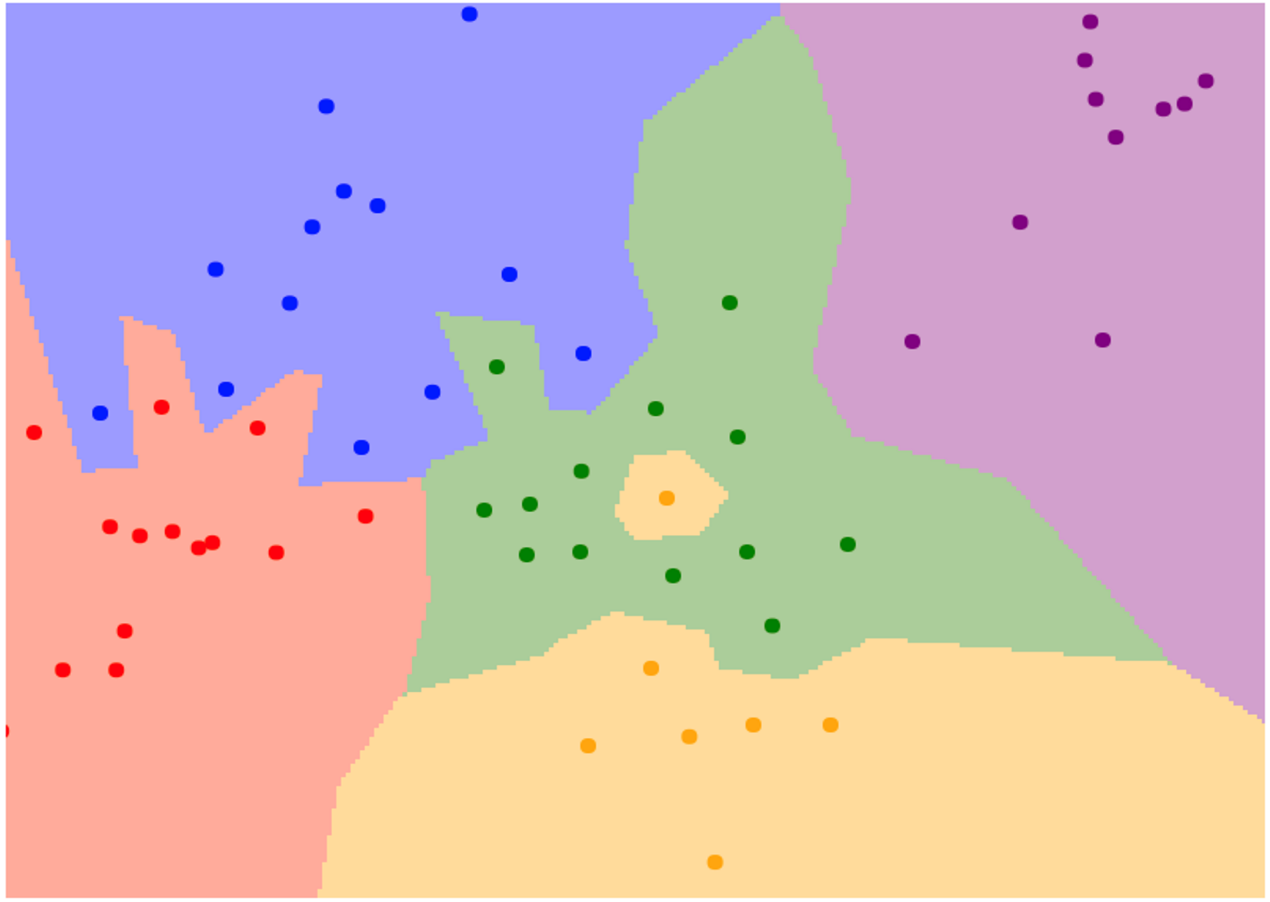




Credit: Justin

Problem: No Function

No function: you need to carry all the data



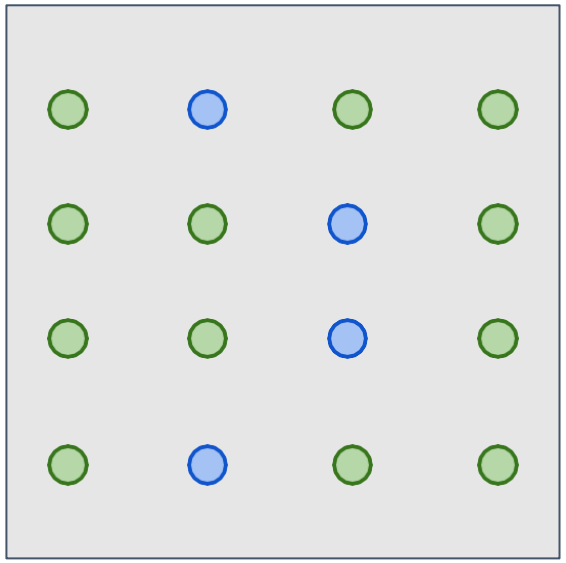
Credit: Justin

Problem: Curse of Dimensionality

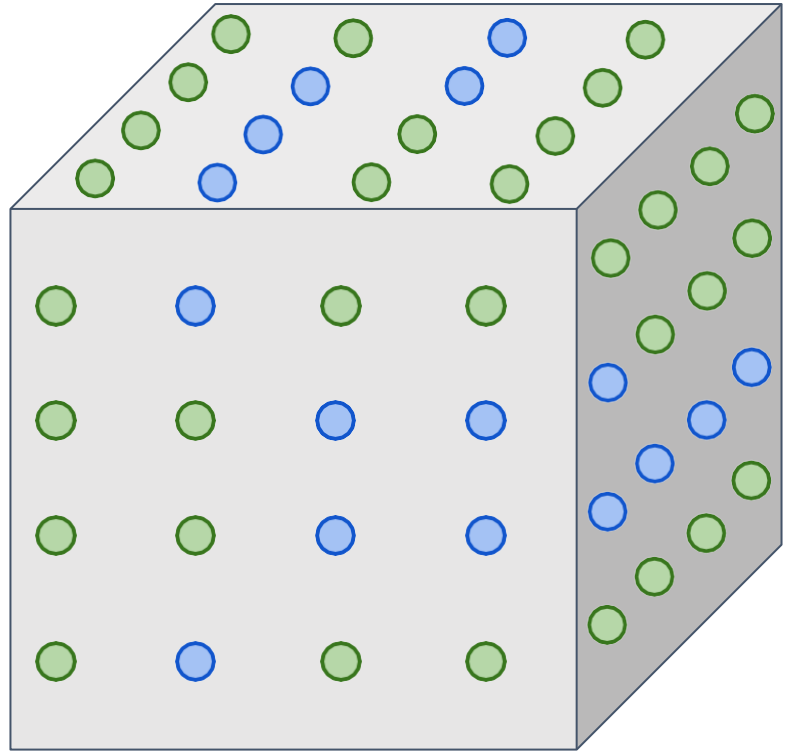
Curse of dimensionality: For uniform coverage of space, number of training points needed grows exponentially with dimension

Dimensions = 1
Points = 4

Dimensions = 2
Points = 4^2



Dimensions = 3
Points = 4^3



Credit: Justin

Any Question ?