10. Other Classifiers
STA3142 Statistical Machine Learning

Kibok Lee

Assistant Professor of

Applied Statistics / Statistics and Data Science

Apr 2, 2024

연세대학교
YONSEI UNIVERSITY

Assignment 1

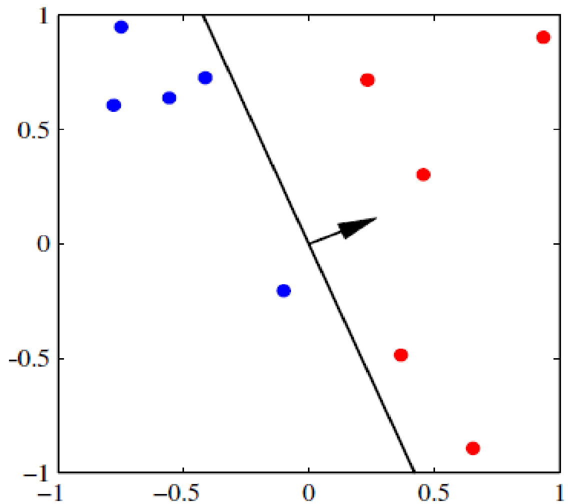
- Due ~~Friday 3/29~~ **Saturday 3/30, 11:59pm**
  - LearnUs was under maintenance around midnight.
- Topics
  - (Programming) NumPy basics
  - (Programming) Linear regression on a polynomial
  - (Math) Derivation and proof for linear regression
- Please read the instruction carefully!
  - Submit one pdf and one zip file separately
  - Write your code only in the designated spaces
  - Do not import additional libraries
  - …
- If you feel difficult, consider to take **option 2**.

Assignment 2

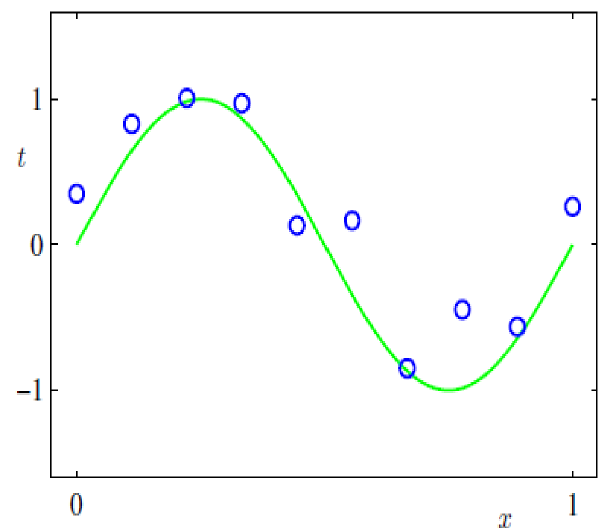
- Due **Friday 4/12, 11:59pm**
- Topics
  - (Math/Programming) Logistic Regression
  - (Math/Programming) Softmax Regression
  - (Math) Gaussian Discriminant Analysis
  - (Programming) Naïve Bayes for Spam Classification
- Please read the instruction carefully!
  - Submit one <u>pdf</u> and one <u>zip</u> file separately
  - Write your code only in the designated spaces
  - Do not import additional libraries
  - …
- If you feel difficult, consider to take **option 2**.

Recap: Supervised Learning

- Learning a function $h: \mathcal{X} \rightarrow \mathcal{Y}$
- Labels could be discrete or continuous
  - Discrete labels: **classification**
  - Continuous labels: **regression**



classification



regression

Recap: Discriminative vs. Generative

- **Probabilistic discriminative models**:
  - Logistic regression: $p(C_1|\mathbf{x}) = \sigma\left(\mathbf{w}^T\phi(\mathbf{x})\right)$
  - Softmax regression: $p(C_k|\mathbf{x}) = \frac{\exp\left(\mathbf{w}_k^T\mathbf{x}\right)}{\sum_j \exp\left(\mathbf{w}_j^T\mathbf{x}\right)}$
- **Probabilistic generative models**:
  - Gaussian discriminant analysis:
    - Prior $p(C_k)$: Bernoulli
    - Likelihood $p(\mathbf{x}|C_k)$: Gaussian
  - Naïve Bayes:
    - Prior $p(C_k)$: Bernoulli
    - Likelihood $p(\mathbf{x}|C_k) = \prod_{j=1}^{M} p\left(x_j|C_k\right)$
      - For spam mail classification, Multinomial

Classification Strategies

- Learning the distributions $p(C_k|x)$
  - Discriminative models: Directly model $p(C_k|x)$ and learn parameters from the training set.
  - Generative models: Learn class densities $p(x|C_k)$ and priors $p(C_k)$ to obtain $p(x, C_k) = p(x|C_k)p(C_k)$
- Nearest neighbor classification
  - Given query data x, find the closest training points and do majority vote.
- Discriminant functions
  - Learn a function $h(x)$ that maps x onto some C_k.

Outline

- Nearest neighbor classification
  - K-Nearest Neighbors

- Discriminant functions
  - Fisher’s Linear Discriminant
  - Perceptron

K-Nearest Neighbors

K-Nearest Neighbors

- Training: Memorize all training data and labels

$$D_{\text{train}} = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(N)}, y^{(N)})\}$$

- Test: Given a test data (or query) $\mathbf{x}^*$,
  find K training data that are closest to $\mathbf{x}^*$,
  and then **majority vote**.

$$D_{K\text{NN}}(\mathbf{x}^*) = \{(\mathbf{x}^{(1)\prime}, y^{(1)\prime}), \dots, (\mathbf{x}^{(K)\prime}, y^{(K)\prime})\}$$
$$\subset D_{\text{train}}$$

$$\hat{y}^* = \operatorname*{argmax}_t \sum_{(\mathbf{x}', y') \in D_{K\text{NN}}(\mathbf{x}^*)} 1[y' = t]$$

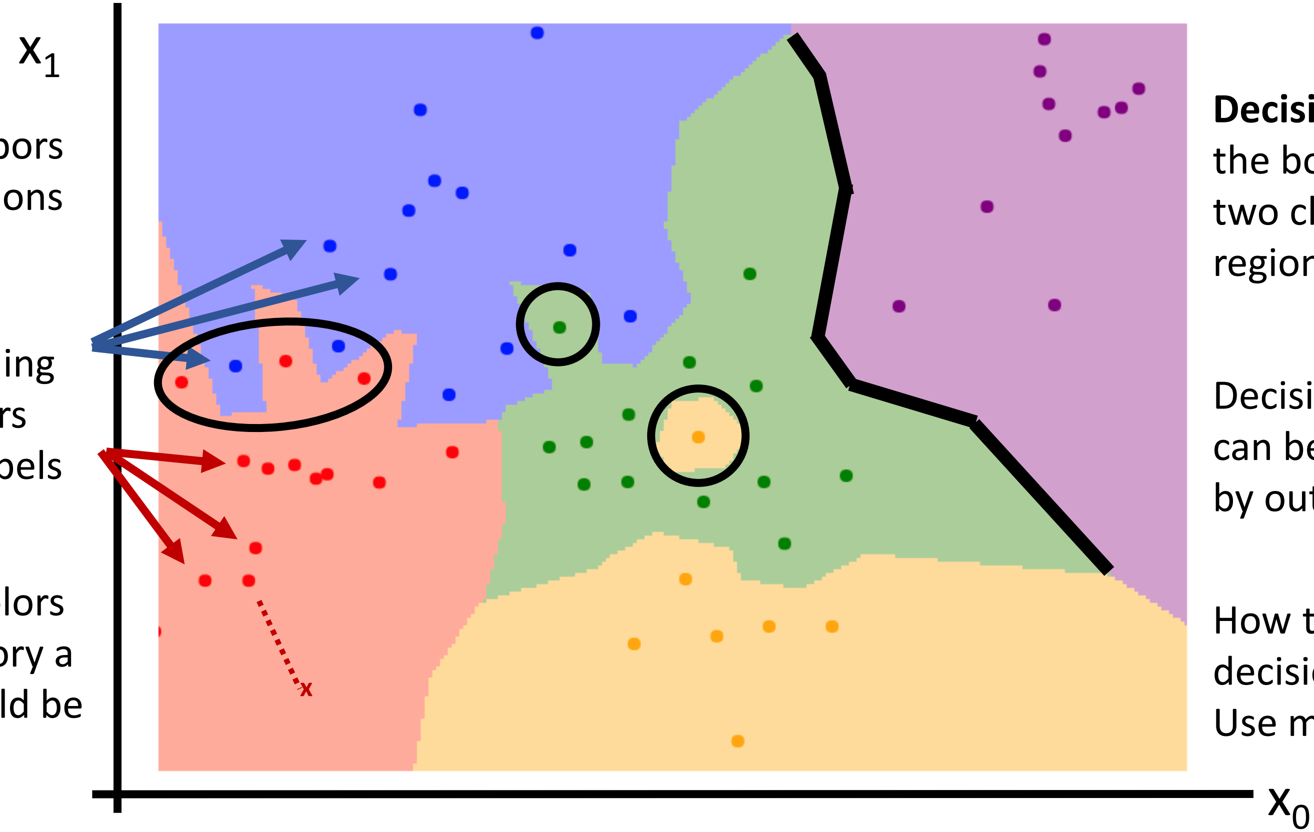
- Note: K-NN can also be used for regression.

NN 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

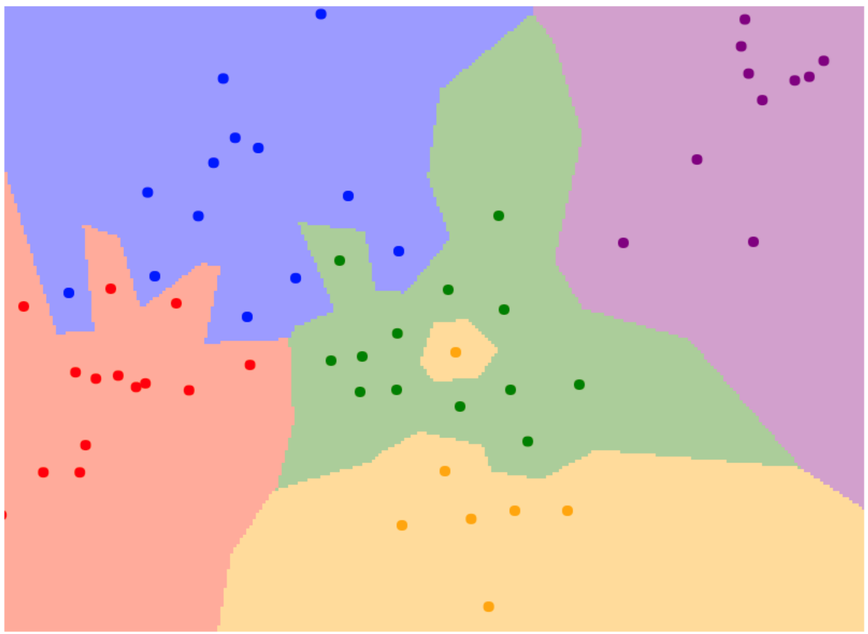
How to smooth out decision boundaries? Use more neighbors!

Slide credit: Justin Johnson

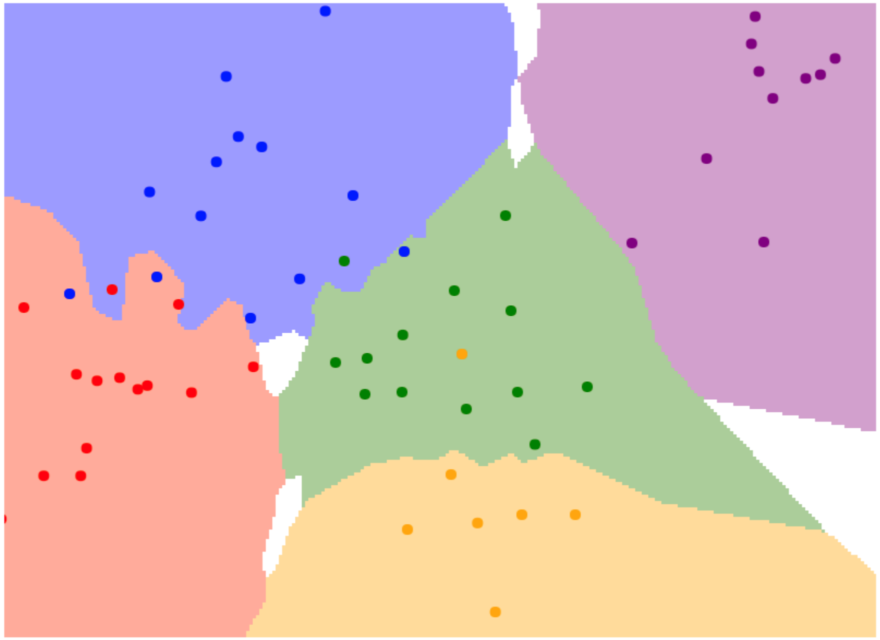
K-Nearest Neighbors

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

K = 1



K = 3

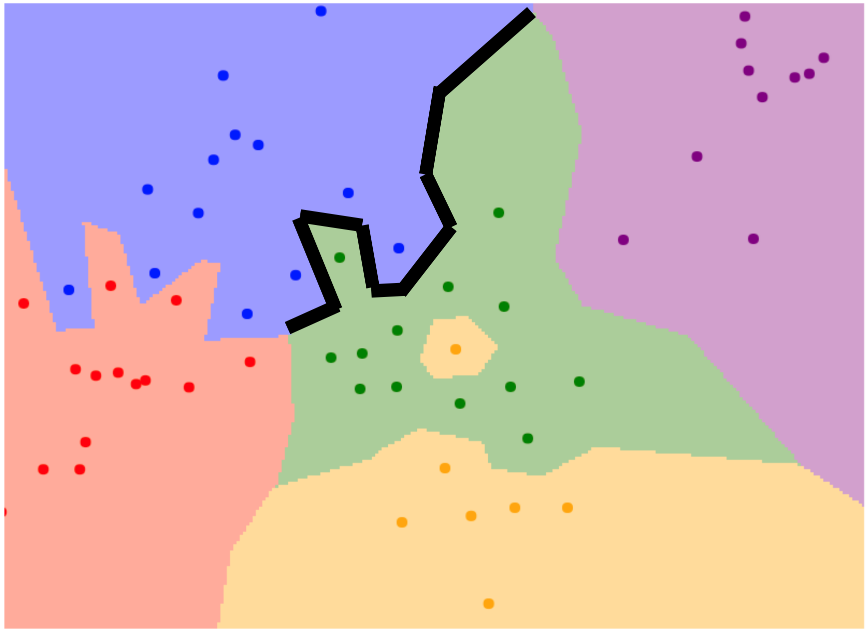


Slide credit: Justin Johnson

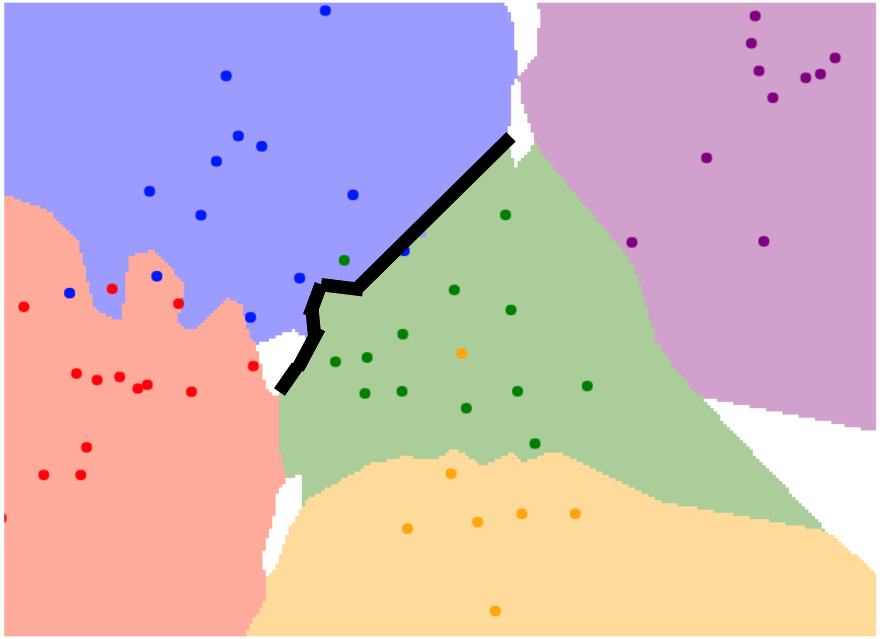
K-Nearest Neighbors

Using more neighbors helps smooth out rough decision boundaries

K = 1



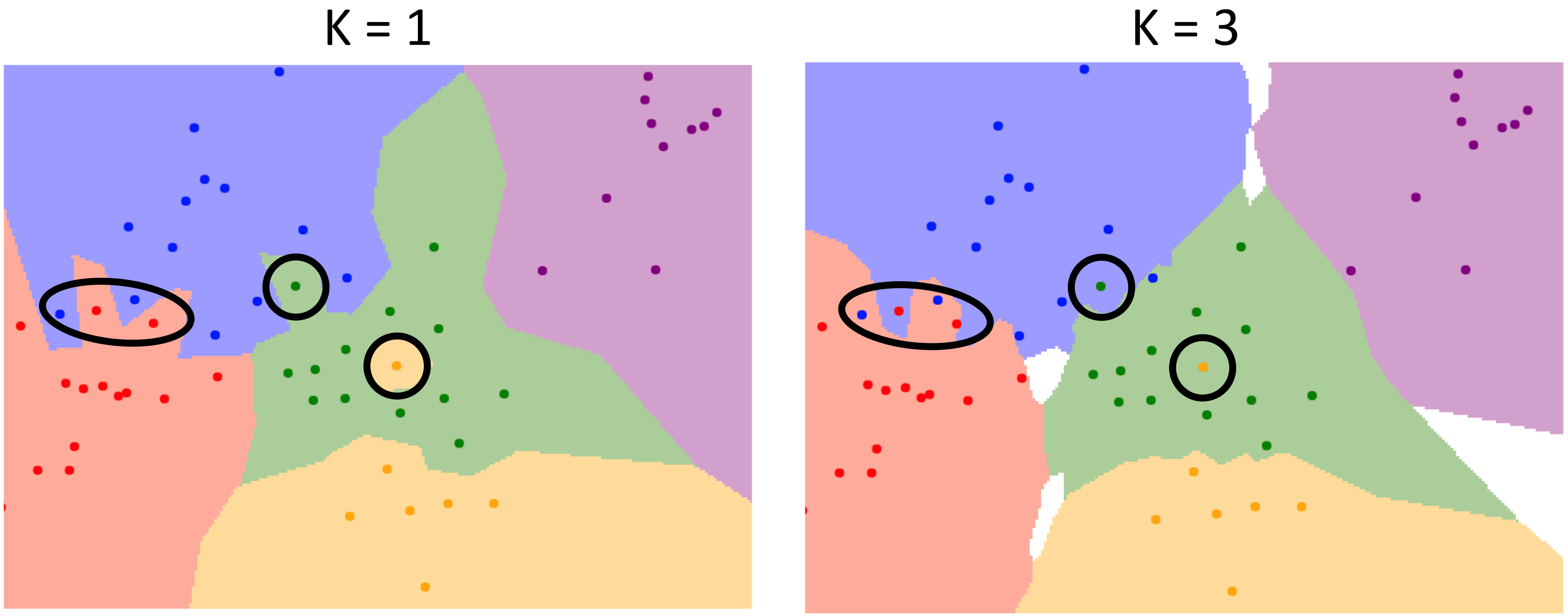
K = 3



Slide credit: Justin Johnson

K-Nearest Neighbors

Using more neighbors helps reduce the effect of outliers

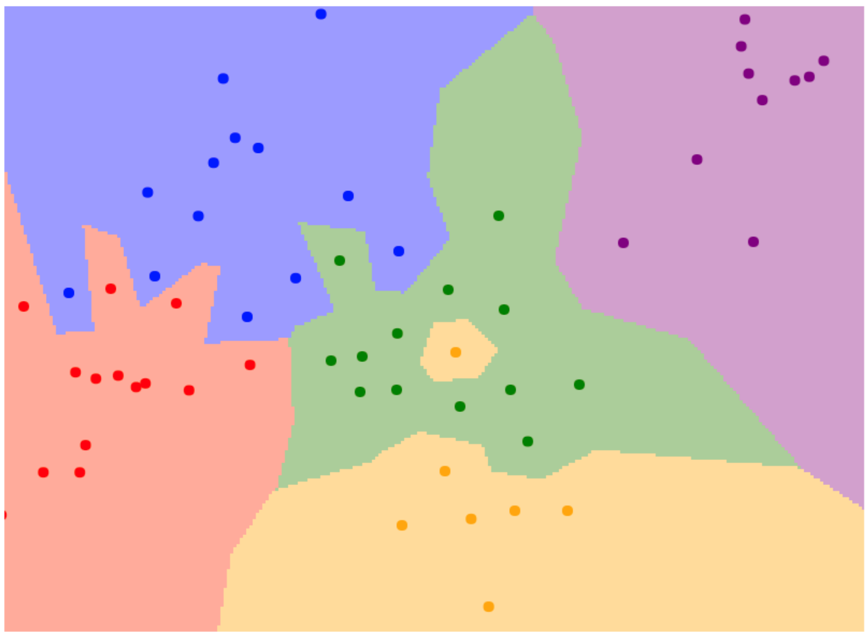


Slide credit: Justin Johnson

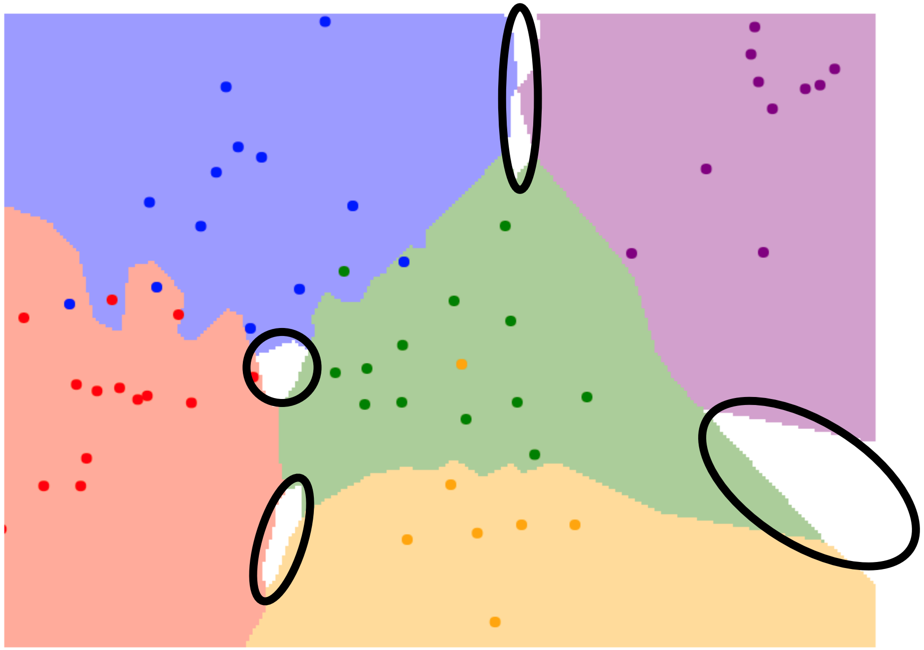
K-Nearest Neighbors

When K > 1 there can be ties between classes. Need to break somehow!

K = 1



K = 3

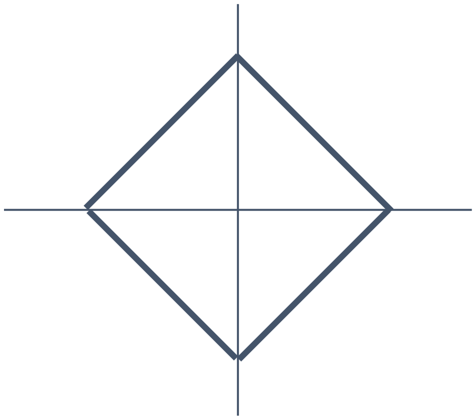


Slide credit: Justin Johnson

K-Nearest Neighbors: Distance Metric

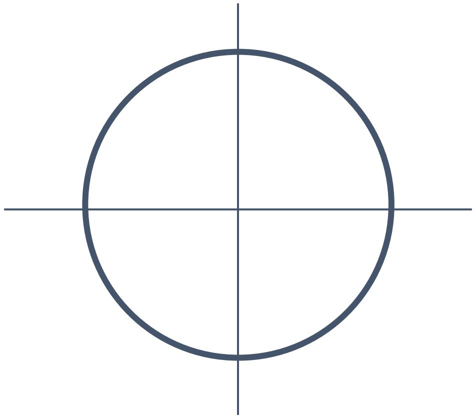
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p \left| I_1^p - I_2^p \right|$$



L2 (Euclidean) distance

$$d_1(I_1, I_2) = \left(\sum_p \left(I_1^p - I_2^p \right)^2 \right)^{\frac{1}{2}}$$

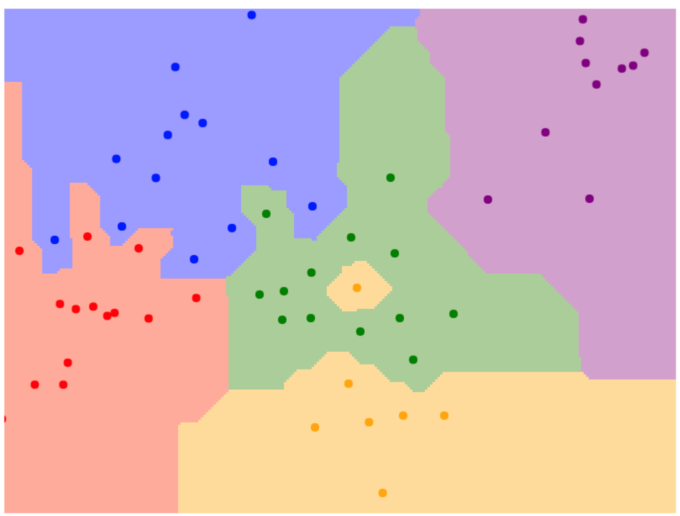


Slide credit: Justin Johnson

K-Nearest Neighbors: Distance Metric

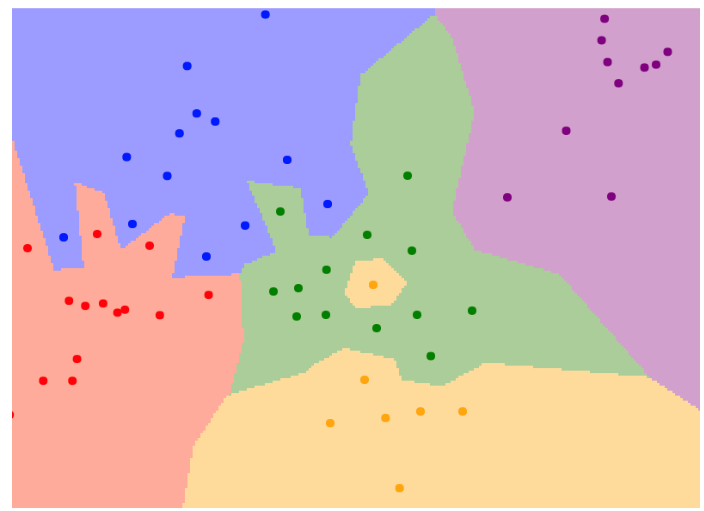
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p \left| I_1^p - I_2^p \right|$$



L2 (Euclidean) distance

$$d_1(I_1, I_2) = \left(\sum_p \left(I_1^p - I_2^p \right)^2 \right)^{\frac{1}{2}}$$



K = 1

Slide credit: Justin Johnson

K-Nearest Neighbors: Distance Metric

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

Example: Compare research papers using tf-idf similarity

Mesh R-CNN 1906.02739v1 pdf

Georgia Gkioxari, Jitendra Malik, Justin Johnson show similar discuss

6/6/2019 cs.CV

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

Slide credit: Justin Johnson

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
Xian-Feng Han, Hamid Laga, Mohammed Bennamoun
6/18/2019 (v1: 6/15/2019) cs.CV | cs.CG | cs.GR | cs.LG
show similar | discuss

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
Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, Yu-Gang Jiang
8/3/2018 (v1: 4/5/2018) cs.CV
show similar | discuss

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
Chao Wen, Yinda Zhang, Zhuwen Li, Yanwei Fu
8/16/2019 (v1: 8/5/2019) cs.CV
Accepted by ICCV 2019
show similar | discuss

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
Edward J. Smith, Scott Fujimoto, Adriana Romero, David Meger
1/31/2019 cs.CV
18 pages
show similar | discuss

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

http://www.arxiv-sanity.com/1906.02739v1

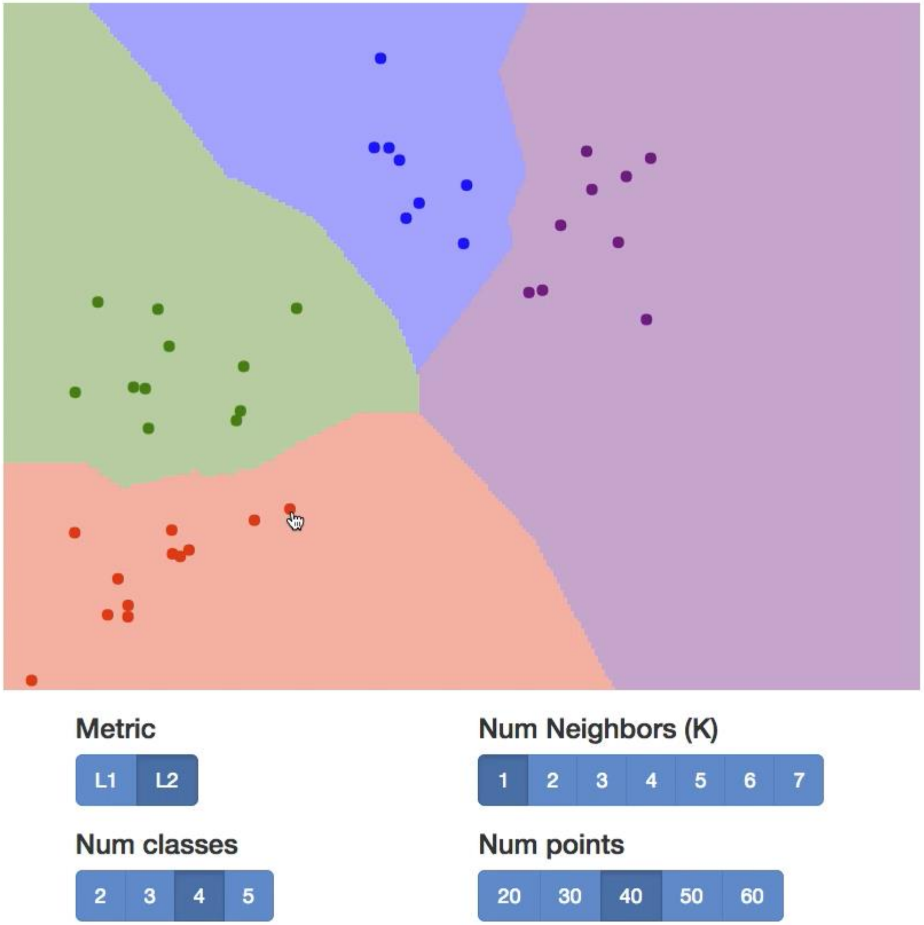
Slide credit: Justin Johnson

K-Nearest Neighbors: Web Demo

Interactively move points around and see decision boundaries change

Play with L1 vs L2 metrics

Play with changing number of training points, value of K



http://vision.stanford.edu/teaching/cs231n-demos/knn/

Slide credit: Justin Johnson

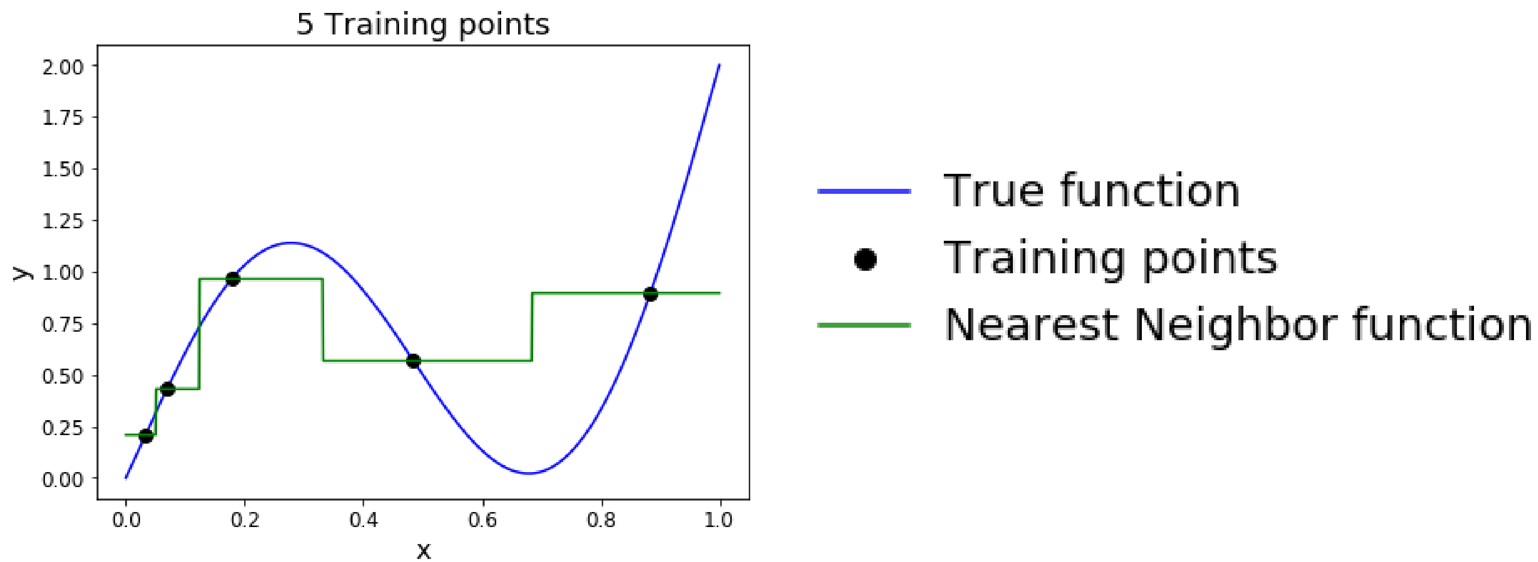
K-NN Hyperparameters

- What is the best value of K to use?
- What is the best distance metric $D(\mathbf{x}, \mathbf{x}')$ to use?
  - These are **hyperparameters**.
    - Cf. Learning rate and regularization coefficient are also hyperparameters.
  - We set them at the start of the learning process, instead of learning from the training data.
- **Answer**: Very problem-dependent. In general, we need to try them all and see what works best for our data/task.
  - Need **validation** to find the best hyperparameters
    - (We will discuss validation in the lecture on model selection)

Slide credit: Justin Johnson

K-Nearest Neighbors: Universal Approximation

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

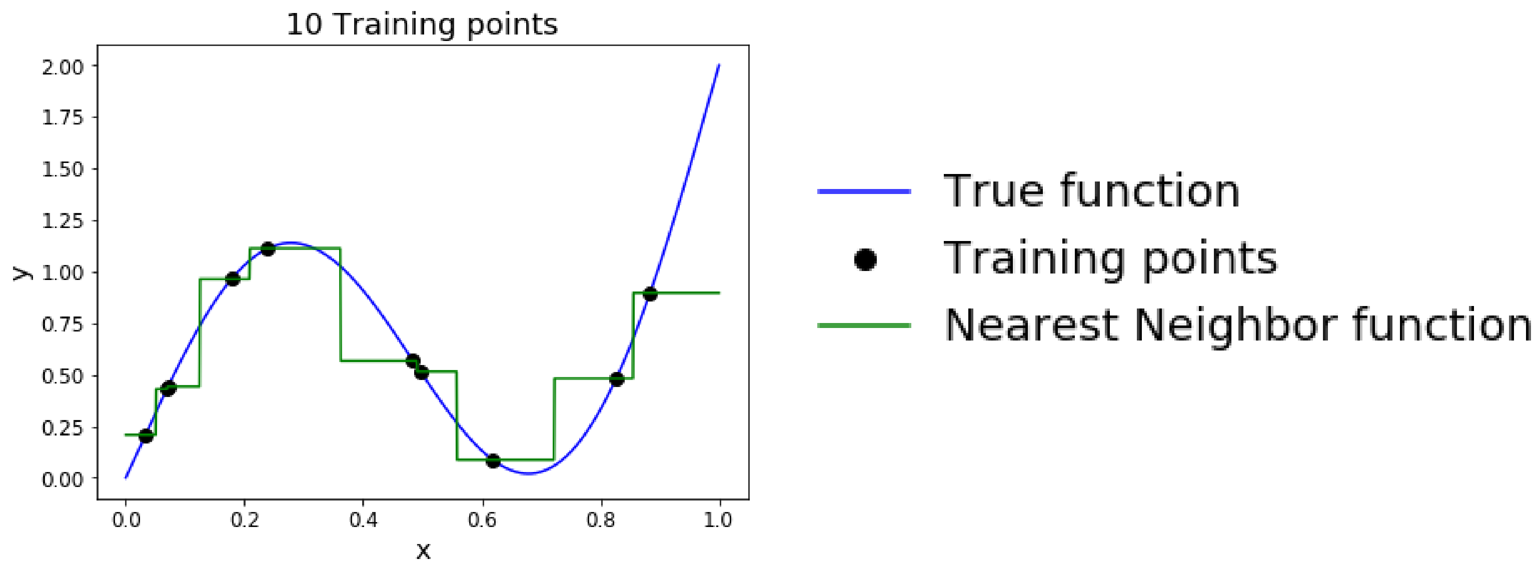


(*) Subject to many technical conditions. Only continuous functions on a compact domain; need to make assumptions about spacing of training points; etc.

Slide credit: Justin Johnson

K-Nearest Neighbors: Universal Approximation

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

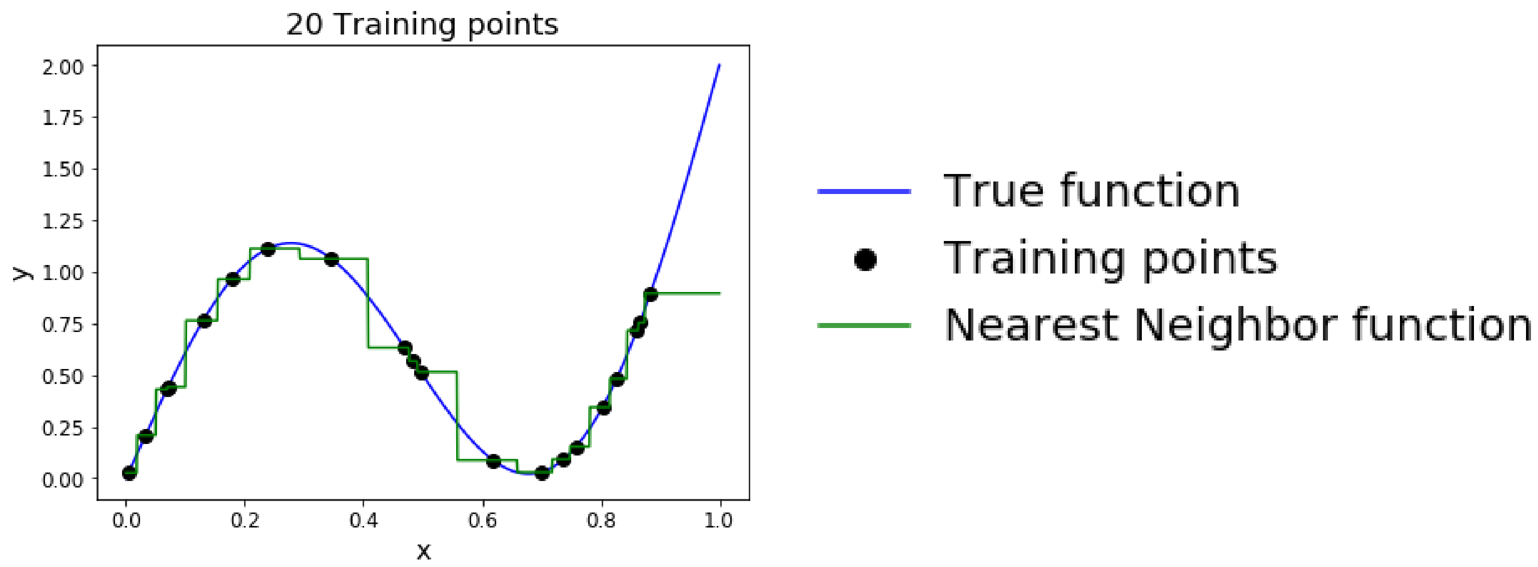


(*) Subject to many technical conditions. Only continuous functions on a compact domain; need to make assumptions about spacing of training points; etc.

Slide credit: Justin Johnson

K-Nearest Neighbors: Universal Approximation

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

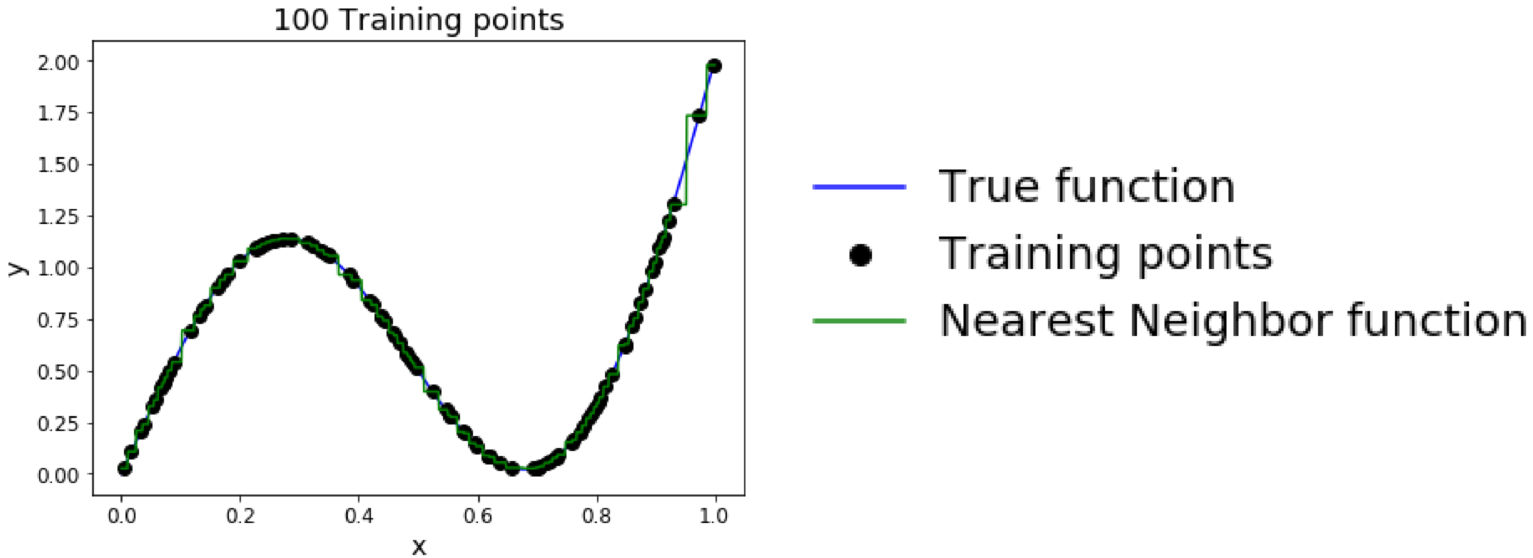


(*) Subject to many technical conditions. Only continuous functions on a compact domain; need to make assumptions about spacing of training points; etc.

Slide credit: Justin Johnson

K-Nearest Neighbors: Universal Approximation

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

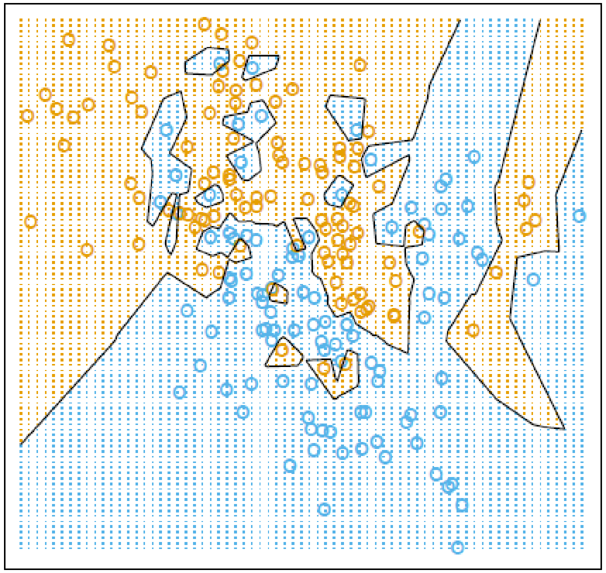


(*) Subject to many technical conditions. Only continuous functions on a compact domain; need to make assumptions about spacing of training points; etc.

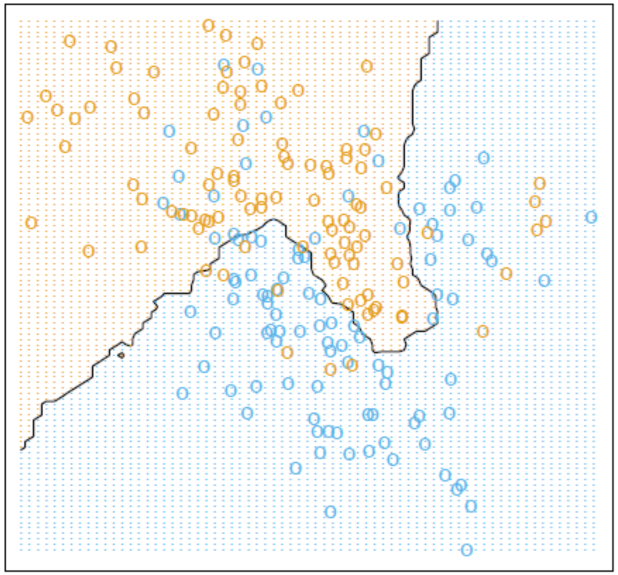
Slide credit: Justin Johnson

K-Nearest Neighbors for Classification

1–Nearest Neighbor Classifier



15-Nearest Neighbor Classifier



- K acts as a smother (bias-variance trade-off)
- Classification performance generally improves as N (training set size) increases.
- For $N \to \infty$, the error rate of the 1-nearest-neighbor classifier is never more than twice the optimal error (obtained from the true conditional class distributions).

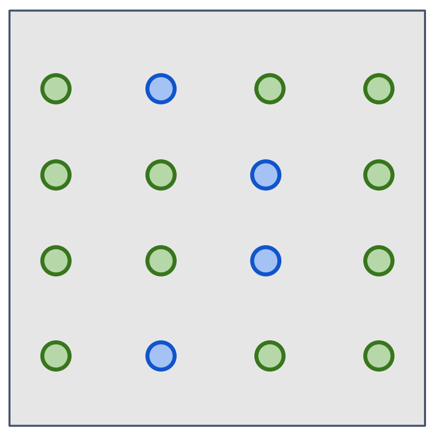
Slide credit: Ben Kuipers

Problem: Curse of Dimensionality

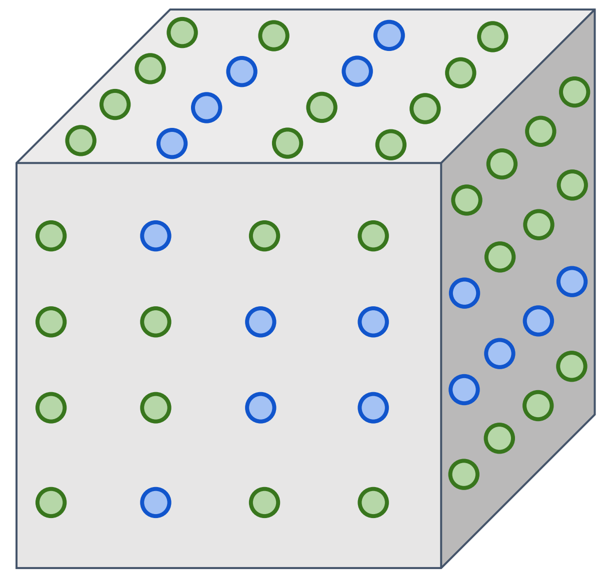
Curse of dimensionality: For uniform coverage of space, number of training points needed grows exponentially with dimension

Dimensions = 1
Points = 4

Dimensions = 2
Points = 4^2



Dimensions = 3
Points = 4^3



Slide credit: Justin Johnson

Problem: Curse of Dimensionality

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

Number of possible 32x32 binary images:

$2^{32x32} \approx 10^{308}$

Number of elementary particles in the visible universe: (source)

$\approx 10^{97}$

Slide credit: Justin Johnson

K-NN Classification vs. Regression

- Classification: Given a test data (or query) $\mathbf{x}^*$, find K training data that are closest to $\mathbf{x}^*$, and then **majority vote**.

$$D_{KNN}(\mathbf{x}^*) = \left\{\left(\mathbf{x}^{(1)\prime}, y^{(1)\prime}\right), \ldots, \left(\mathbf{x}^{(K)\prime}, y^{(K)\prime}\right)\right\} \subset D_{train}$$

$$\hat{y}^* = \operatorname*{argmax}_t \sum_{(\mathbf{x}', y') \in D_{KNN}(\mathbf{x}^*)} 1[y' = t]$$

- Regression: Take **average** over y' values.

$$\hat{y}^* = \frac{1}{K} \sum_{(\mathbf{x}', y') \in D_{KNN}(\mathbf{x}^*)} y'$$

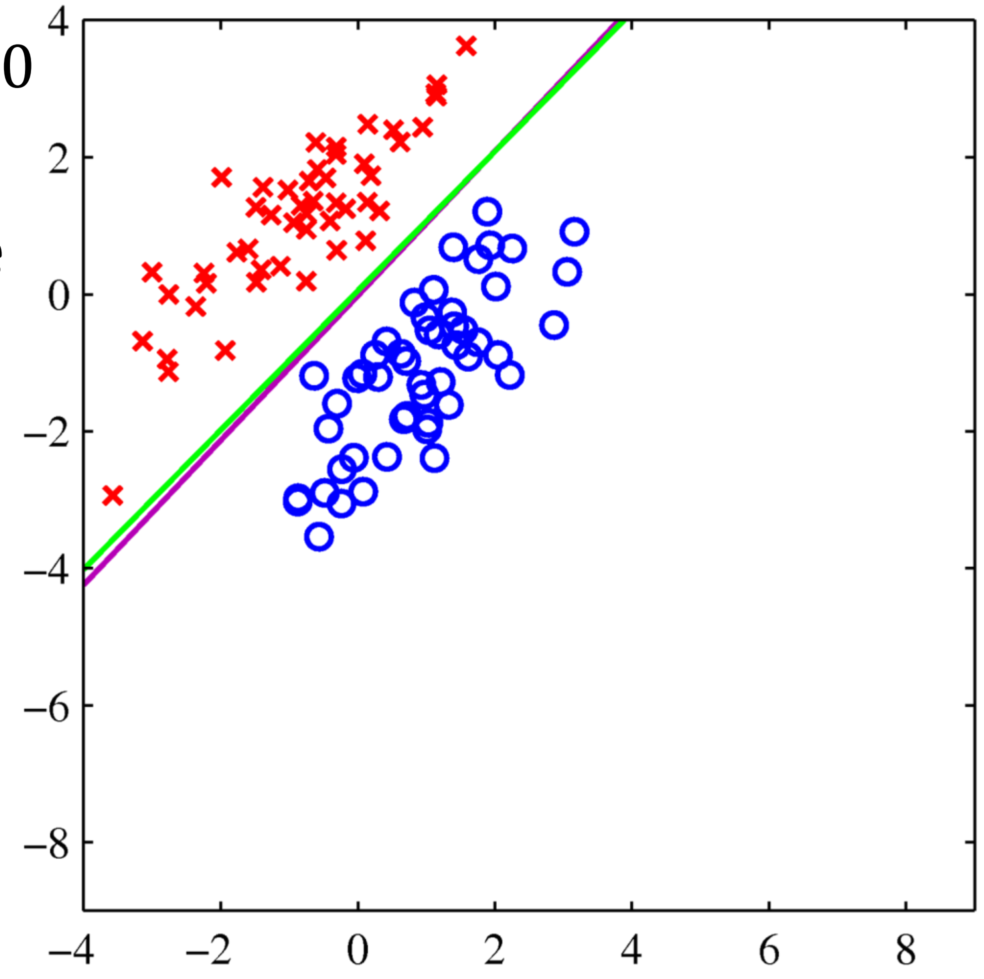
Pros and Cons of K-NN

- **Advantages**:
  - Simple and flexible (no assumption on distribution)
  - Effective for low dimensional inputs
- **Disadvantages**:
  - Expensive: need to remember (store) and search through all the training data for every prediction
  - Curse of dimensionality: the number of data must grow exponentially to keep the same density
  - Not robust to irrelevant features: If data has irrelevant/noisy features, then distance function does not reflect similarity between examples

Discriminant functions

Two-Class Linear Discriminant

- Classification rule:
  C_1 if $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 \geq 0$
  C_0 otherwise.
- Suppose we do not have any probabilistic model assumption.
  - Magenta:
    Linear regression to one-hot encoded labels
  - Cf. Green:
    Logistic regression



Multi-Class Linear Discriminant

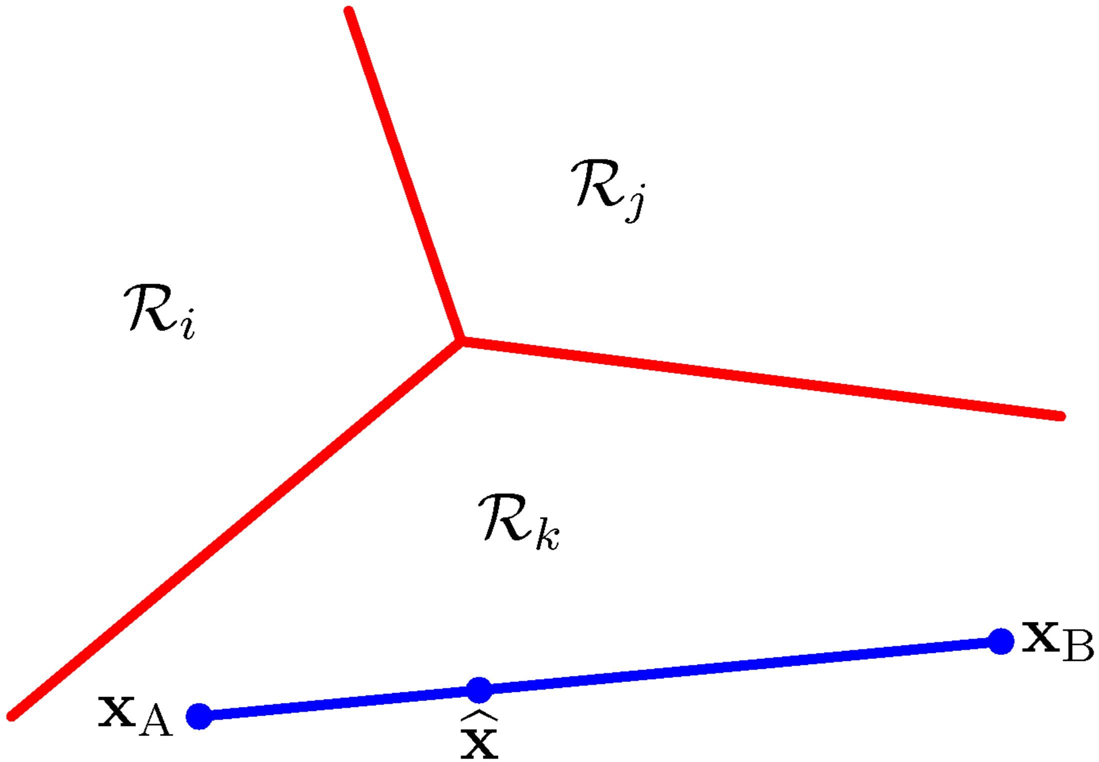
- Each class C_k gets its own function
$$h_k(\mathbf{x}) = \mathbf{w}_k^T\mathbf{x} + w_{k,0}$$

- Assign $\mathbf{x}$ to C_k if
$$h_k(\mathbf{x}) > h_k(\mathbf{x}) \text{ for all } j \neq k$$

- The decision regions are convex polyhedra.

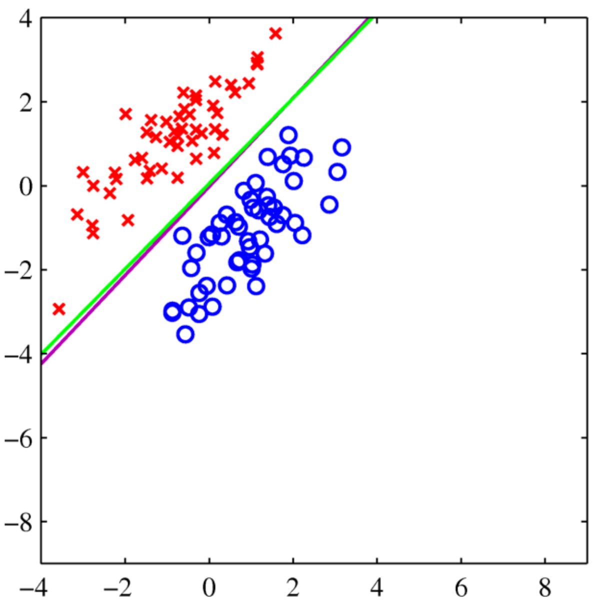
Decision Regions

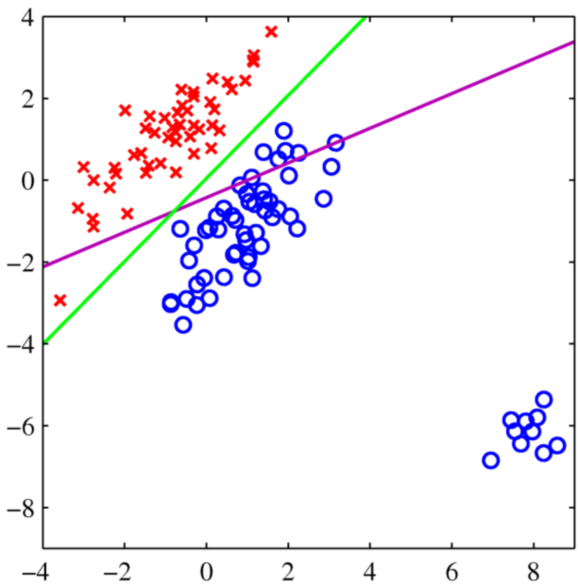
- Decision regions are convex, with piecewise linear boundaries.



Linear Discriminant Functions

- How about $\mathbf{w}$ that minimizes squared error?
  - Label y versus linear prediction $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$.
  - Least squares is too sensitive to outliers.
  - Logistic regression is robust to outliers, but requires an assumption (probabilistic discriminative model).





Classification Strategies

- Discriminant functions: Learn a function $h(x)$ that maps x onto some C_k.
  - Fisher's linear discriminant
  - Perceptron learning

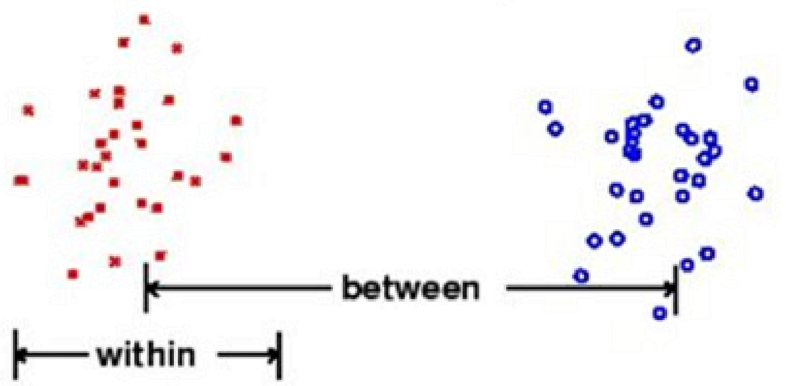
Fisher's Linear Discriminant

Fisher's Linear Discriminant

- Use $\mathbf{w}$ to project $\mathbf{x}$ onto one dimension, and then threshold the value by $-w_0$.

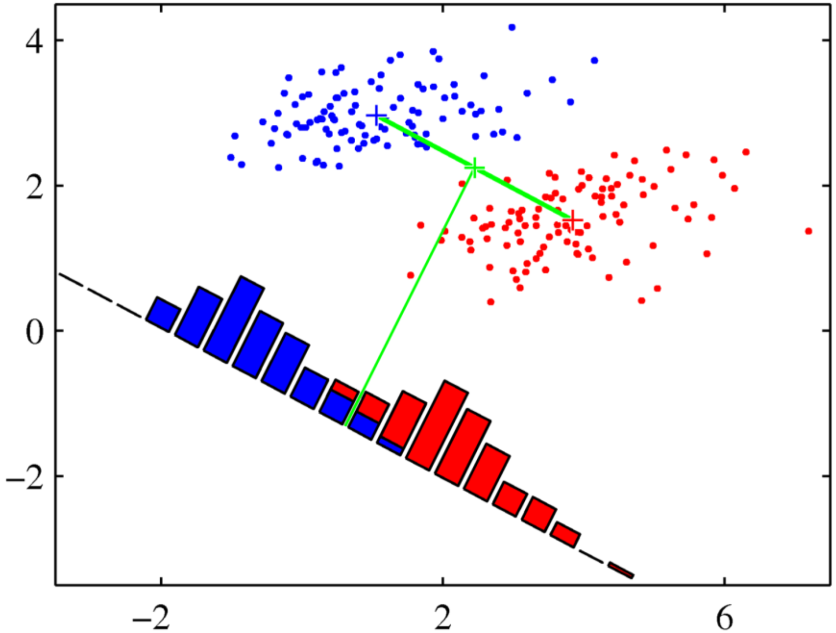
  C_1 if $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} \geq -w_0$

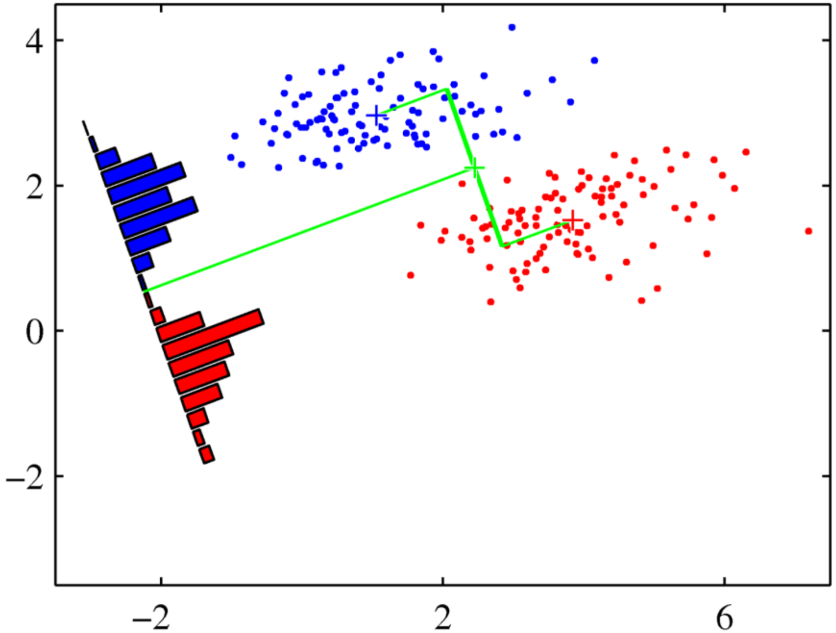
  C_0 otherwise.
- Select $\mathbf{w}$ that best **separates** the classes.
- The learning objective should simultaneously
  - Maximize between-class variances
  - Minimize within-class variances



Fisher’s Linear Discriminant

- The learning objective should simultaneously
  - Maximize between-class variances
  - Minimize within-class variances
- Because maximizing separation alone doesn’t work.
  - Minimizing class variance is a big help.





FLD: Formulation

- We want to maximize the distance between classes

$$\underline{m_2} - m_1 \equiv \mathbf{w}^T(\underline{\mathbf{m}_2} - \mathbf{m}_1) \qquad \text{where } \mathbf{m}_k = \frac{1}{N_k}\sum_{n \in C_k} \mathbf{x}_n$$

Projected mean

Mean

- While minimizing the distance within each class

$$s_1^2 + s_2^2 \equiv \sum_{n \in C_1} (\mathbf{w}^T\mathbf{x}_n - m_1)^2 + \sum_{n \in C_2} (\mathbf{w}^T\mathbf{x}_n - m_2)^2$$

- Objective function: $J(\mathbf{w}) = \dfrac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$
  - The bias w_0 is not in the learning objective; it should be determined separately after finding $\mathbf{w}$.

FLD: Derivation

- Numerator: $m_2 - m_1 = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)$

  $\Rightarrow \|m_2 - m_1\|^2 = \mathbf{w}^T \underbrace{(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T}_{S_B\text{: Between-class scatter}} \mathbf{w}$

- Denominator:

$$s_k^2 = \sum_{n \in C_k} (\mathbf{w}^T \mathbf{x}_n - \mathbf{w}^T \mathbf{m}_k)^2 = \sum_{n \in C_k} \mathbf{w}^T (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T \mathbf{w}$$

$$\Rightarrow s_1^2 + s_2^2 = \mathbf{w}^T \underbrace{\left[\sum_{k=1}^{2} \sum_{n \in C_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T \right]}_{S_W\text{: Within-class scatter}} \mathbf{w}$$

- Objective function:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

- Solution: $\mathbf{w} \propto S_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$

Generalized FLD

- Two-class
  - Learning objective: $$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$
  - Solution: $$\mathbf{w} \propto S_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

- Multi-class
  - Learning objective: $$J(W) = \mathrm{tr}(W^T S_W W)^{-1}(W^T S_B W)$$
  - Solution: $$S_B W = S_W W \Lambda$$

    - where
$$S_W = \sum_k \sum_{y^{(i)}=k} (\mathbf{x}^{(i)} - \mathbf{m}_k)(\mathbf{x}^{(i)} - \mathbf{m}_k)^T$$
$$S_B = \sum_k N_k(\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$$
$$\mathbf{m} = \frac{1}{N}\sum_i x^{(i)} = \frac{1}{N}\sum_k N_k \mathbf{m}_k$$

Generalized FLD

- Multi-class
  - Learning objective: $J(W) = \mathrm{tr}(W^T S_W W)^{-1}(W^T S_B W)$
  - Solution: $S_B W = S_W W \Lambda$

    - where
$$S_W = \sum_k \sum_{y^{(i)}=k} (\mathbf{x}^{(i)} - \mathbf{m}_k)(\mathbf{x}^{(i)} - \mathbf{m}_k)^T$$
$$S_B = \sum_k N_k(\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$$
$$\mathbf{m} = \frac{1}{N}\sum_i x^{(i)} = \frac{1}{N}\sum_k N_k \mathbf{m}_k$$
    - Derived by generalized eigenvalue decomposition
      - Not directly usable for classification, as each C_k does not get its own function; need to build another classifier
      - Up to $(K-1)$ linearly independent $\mathbf{w}_k$'s $(\mathrm{rank}(W) \leq K-1)$
      - Not unique: WZ is also a solution for nonsingular Z

Generalized FLD

- Multi-class
  - Learning objective: $J(W) = \mathrm{tr}(W^T S_T W)^{-1}(W^T S_B W)$
  - Solution: $S_B W = S_T W \Lambda$
    - where
$$S_T = \sum_i (\mathbf{x}^{(i)} - \mathbf{m})(\mathbf{x}^{(i)} - \mathbf{m})^T = S_W + S_B$$
$$S_B = \sum_k N_k(\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$$
$$\mathbf{m} = \frac{1}{N}\sum_i x^{(i)} = \frac{1}{N}\sum_k N_k \mathbf{m}_k$$
  - This is equivalent to the within-scatter version.
    - Proof sketch: Intuitively,
$$\mathrm{argmax}_W \frac{f(W)}{g(W)} = \mathrm{argmax}_W \frac{f(W)}{f(W) + g(W)}$$

Generalized FLD

- Often called linear discriminant analysis (LDA)
  - Or Fisher's LDA (FLDA)
  - Cf. For two-class classification, **GDA with shared covariance** gives us the same weight vector $\mathbf{w}$, and is also often called LDA.

- Mainly used for **dimensionality reduction**
  - From an arbitrary D to D' ($\leq K-1$)
    - Choose D' eigenvectors with largest eigenvalues
  - This is supervised dimensionality reduction; we will cover other (un)supervised dimensionality reduction methods later.

Recap: GDA with shared covariance

- Maximum likelihood estimation (MLE):

$$\phi = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = 1\}$$

$$\mu_0 = \frac{\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = 0\}\mathbf{x}^{(i)}}{\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = 0\}}$$

$\mathbf{m}_2$ in FLD formulation

$$\mu_1 = \frac{\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = 1\}\mathbf{x}^{(i)}}{\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = 1\}}$$

$\mathbf{m}_1$ in FLD formulation

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N} (\mathbf{x}^{(i)} - \mu_{y^{(i)}})(\mathbf{x}^{(i)} - \mu_{y^{(i)}})^T$$

$N \cdot S_W$ in FLD formulation

Generalized FLD (Out of Scope)

- Variations of LDA
  - Regularized LDA: $S_T \leftarrow S_T + \lambda I$
  - Uncorrelated LDA: $W^T S_T W = I$
  - Orthogonal LDA: $W \leftarrow U$ where $W = U \Sigma V^T$ is SVD
  - Kernel LDA: $\mathbf{x} \leftarrow \phi(\mathbf{x})$ and use kernel trick
    - (We will discuss the kernel method later)
  - Combination of some of the above

Generalized FLD (Out of Scope)

- Equivalent to **least squares** from **centered data** to **a family of targets** [Lee et al., 2015]
  - if and only if $\mathrm{rank}(S_B) = K - 1$
  - i.e., their solutions map data to the same latent space
    - $W^T(X - \mathbf{m}\mathbf{1}^T) \approx Y \Leftrightarrow \underset{W}{\mathrm{argmax}}\, \mathrm{tr}(W^T S_T W)^{-1}(W^T S_B W)$
  - A possible $Y = [I_{K-1}\ \mathbf{0}_K]L \in \mathbb{R}^{(K-1)\times N}$
    - $L \in \mathbb{R}^{K\times N}$ is a one-hot encoded label matrix
    - A collection of $(K-1)$ one-hot vectors and a zero vector
  - A possible $Y = B \in \mathbb{R}^{(K-1)\times N}$ where $S_B = XB^TBX^T$
    - $X \in \mathbb{R}^{D\times N}$ is the data matrix and B is a full rank factorization

[Lee et al.] On the Equivalence of Linear Discriminant Analysis and Least Squares. In *AAAI*, 2015.

Perceptron

Perceptron: Formulation

- Classification model: $h(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T \phi(\mathbf{x})\right)$
  - where $\text{sign}(a) = \begin{cases} +1 \text{ if } a \geq 0 \\ -1 \text{ if } a < 0 \end{cases}$

- Target: $y = +1$ for C_1, $y = -1$ for C_2.

- Then, we always want:

$$\mathbf{w}^T \phi(\mathbf{x}) y > 0$$

Perceptron: Formulation

- Objective function

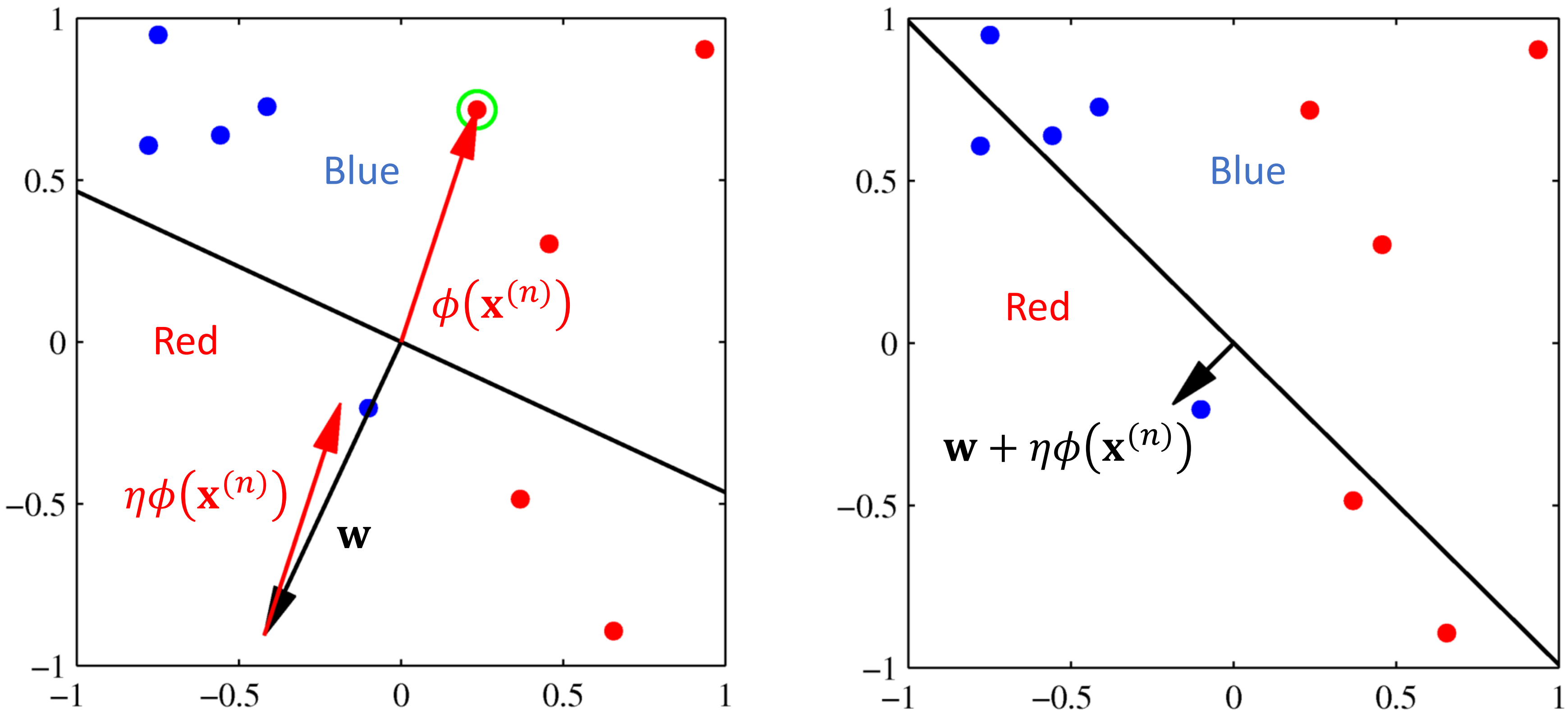
$$E_P(\mathbf{w}) = -\sum_{\mathbf{x}^{(n)} \in \mathcal{M}} \mathbf{w}^T \phi\left(\mathbf{x}^{(n)}\right) y^{(n)}$$

  - where $\mathcal{M} = \left\{\mathbf{x}^{(n)}: y^{(n)} \neq \mathbf{w}^T \phi\left(\mathbf{x}^{(n)}\right)\right\}$ is a set of misclassified points
  - It counts errors from misclassified data.
- Iterative optimization
  - By stochastic gradient descent
  - Update $\mathbf{w}$ according to the **misclassified** data:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w} + \eta \phi\left(\mathbf{x}^{(n)}\right) y^{(n)}$$

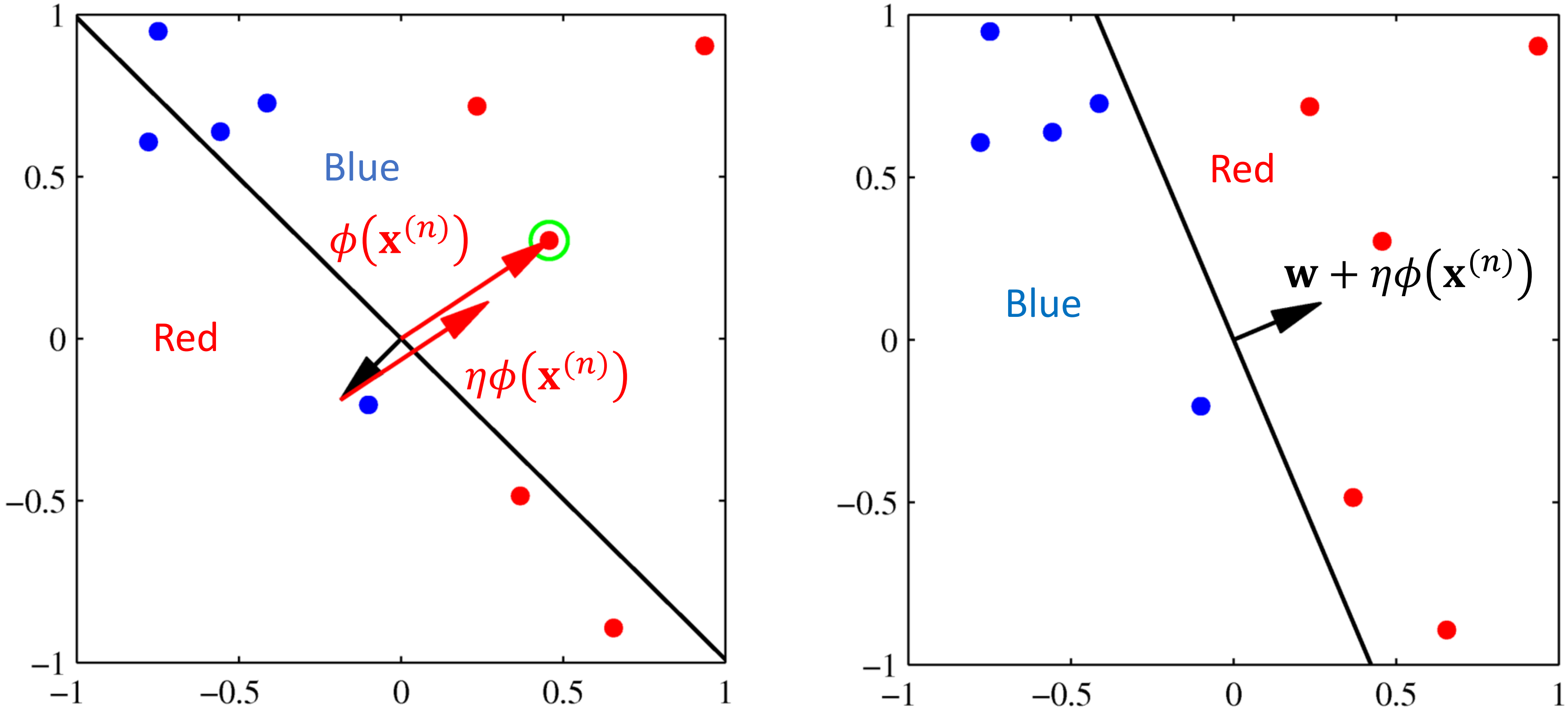
Perceptron: Learning Example

- If $\mathbf{x}^{(n)}$ is misclassified, add $\eta\phi\left(\mathbf{x}^{(n)}\right)$ into $\mathbf{w}$.



Perceptron: Learning Example

- If $\mathbf{x}^{(n)}$ is misclassified, add $\eta\phi(\mathbf{x}^{(n)})$ into $\mathbf{w}$.



Perceptron Learning

- Perceptron convergence theorem:
  - If there exists an exact solution (i.e., if the training data is linearly separable),
  - Then the learning algorithm will find it in a finite number of steps.

- Limitations of perceptron learning:
  - The convergence can be very slow.
  - If dataset is not linearly separable, it won't converge.
  - Does not generalize well to $K > 2$ classes.

Recap: Classification Strategies

- Learning the distributions $p(C_k|x)$
  - Discriminative models: Directly model $p(C_k|x)$ and learn parameters from the training set.
  - Generative models: Learn class densities $p(x|C_k)$ and priors $p(C_k)$ to obtain $p(x, C_k) = p(x|C_k)p(C_k)$
- Nearest neighbor classification
  - Given query data x, find the closest training points and do majority vote.
- Discriminant functions
  - Learn a function $h(x)$ that maps x onto some C_k.

Next: Regularization, Validation