DSC 190

Machine Learning: Representations

Lecture 5 | Part 1

Choosing RBF Locations

Recap

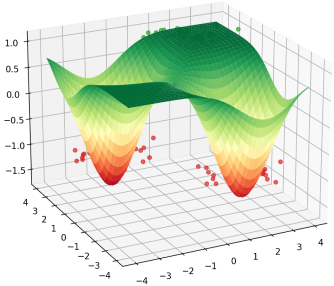
- We map data to a new representation by first choosing **basis functions**.
- Radial Basis Functions (RBFs), such as Gaussians, are a popular choice.
- Requires choosing **center** for each basis function.

Today’s Lecture

- How do we choose basis function centers automatically?

Prediction Function

- Our prediction function *H* is a surface that is made up of Gaussian “bumps”.

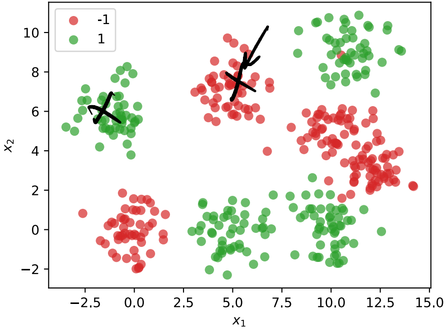


$$H(\vec{x}) = w_0 + w_1 e^{-\|\vec{x}-\vec{\mu}_1\|^2/\sigma^2} + w_2 e^{-\|\vec{x}-\vec{\mu}_2\|^2/\sigma^2}$$

Choosing Centers

- Place the centers where the value of the prediction function should be controlled.

- Intuitively: place centers where the data is.



Approaches

1. Every data point as a center
2. Randomly choose centers
3. Clustering

Clustering

- Group data points into **clusters**.
- Cluster centers are good places for RBFs.
- We'll use k-means clustering to pick k centers.

DSC 190

Machine Learning: Representations

Lecture 5 | Part 2

k-means Clustering

Digression...

- Let’s forget about RBFs for a minute...
- What is **clustering**?

Clustering

- **Clustering** is a machine learning task whose goal is to find **group structure** in data.

- Why?
  - Exploratory data analysis.
  - Representation learning (today).

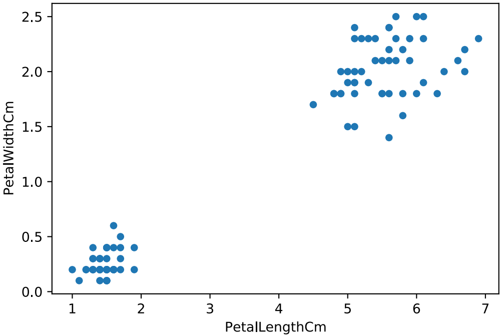
Example

- We gather measurements $\vec{x}^{(i)}$ of a bunch of flowers.
  - Petal length and petal width

- **Question**: how many species are there?

- **Goal**: **cluster** the similar flowers into groups.

Example



Supervised v. Unsupervised

- Clustering is an example of an **unsupervised** learning task.

Supervised Learning

- We tell the machine the “right answer”.
  - There is a **ground truth**.

- Data set: $\{(\vec{x}^{(i)}, y_i)\}$.

- **Goal**: learn relationship between features $\vec{x}^{(i)}$ and labels y_i.

- **Examples**: classification, regression.

Unsupervised Learning

- We don’t tell the machine the “right answer”.
  - In fact, there might not be one!
- Data set: $\{\vec{x}^{(i)}\}$ (usually no test set)
- **Goal**: learn the **structure** of the data itself.
  - To discover something, for compression, to use as a feature later.
- **Example**: **clustering**

Ground Truth

- If we don’t have labels, we can’t measure accuracy.

- Sometimes, labels don’t exist.

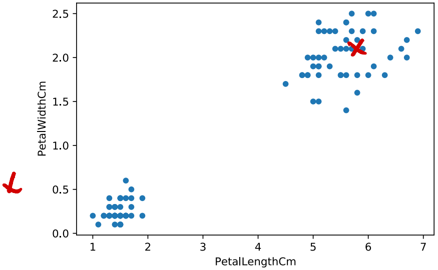
- Example: cluster customers into types by previous purchases.

Clustering Approaches

- There are many approaches to clustering.
- One of the most popular is **k-means clustering**.
- It turns clustering into an optimization problem.

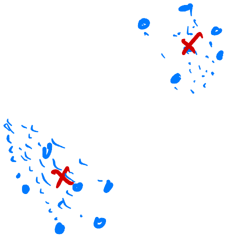
Clustering as Optimization

- **Goal**: compress each clustering into a single point while minimizing information loss.



K-Means Objective

$K=2$



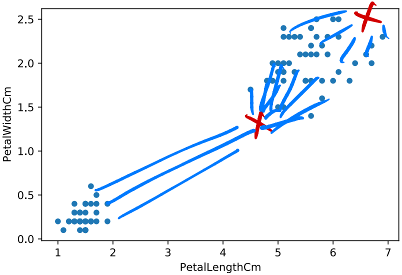
- **Given**: data, $\{\vec{x}^{(i)}\} \in \mathbb{R}^d$ and a parameter k.

- **Find**: k cluster centers $\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)}$ so that the average squared distance from a data point to nearest cluster center is small.

- The **k-means objective function**:

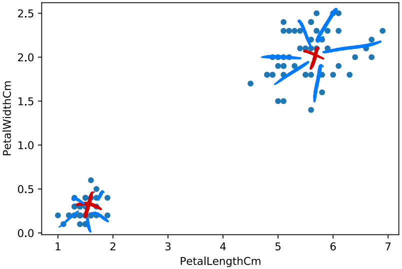
$$\text{Cost}(\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)}) = \frac{1}{n} \sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \|\vec{x}^{(i)} - \vec{\mu}^{(j)}\|^2$$

Clustering as Optimization



$$\mathrm{Cost}(\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)}) = \frac{1}{n} \sum_{i=1}^{n} \min_{j \in \{1, \ldots, k\}} \| \vec{x}^{(i)} - \vec{\mu}^{(j)} \|^2$$

Clustering as Optimization



$$\mathrm{Cost}(\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)}) = \frac{1}{n} \sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \| \vec{x}^{(i)} - \vec{\mu}^{(j)} \|^2$$

Optimization

- **Goal**: find $\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)}$ minimizing k-means objective function.

- **Problem**: this is NP-Hard.

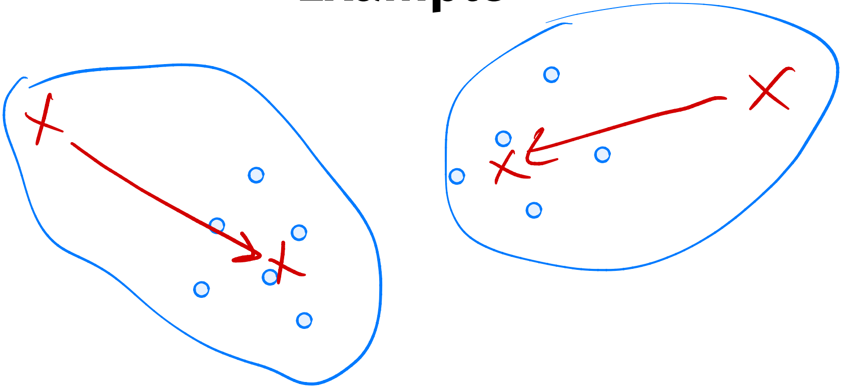
- We use a heuristic instead of solving exactly.

Lloyd’s Algorithm for K-Means

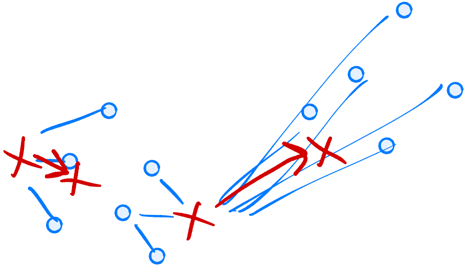
- Initialize centers, $\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(k)}$ somehow.

- Repeat until convergence:
  - Assign each point $\vec{x}^{(i)}$ to closest center
  - Update each $\vec{\mu}^{(i)}$ to be mean of points assigned to it

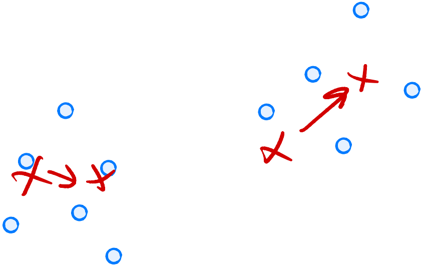
Example



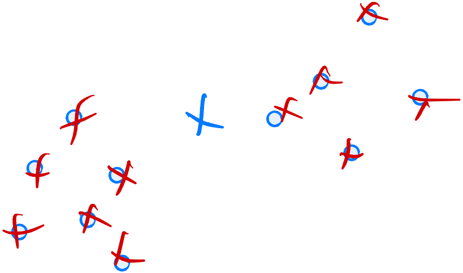
Example



Example



Example

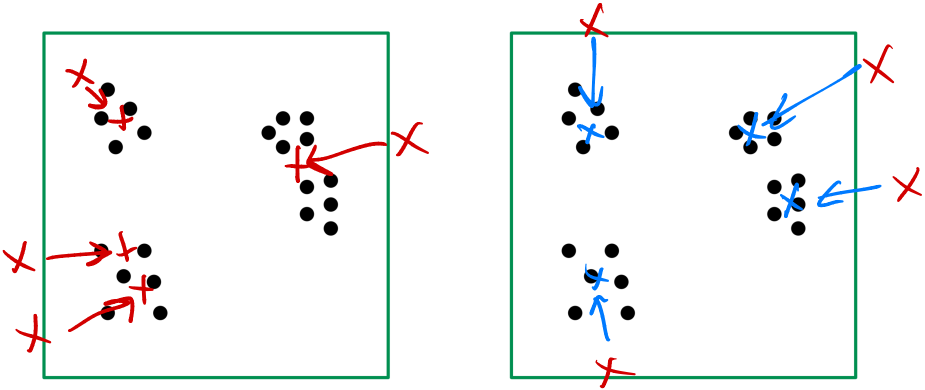


Theory

- Each iteration reduces cost.
- This guarantees convergence to a **local** min.
- Initialization is very important.

$k=4$

Example



Initialization Strategies

- Basic Approach: Pick k data points at random.

- Better Approach: **k-means++**:
  - Pick first center at random from data.
  - Let $C = \{\vec{\mu}^{(1)}\}$ (centers chosen so far)
  - Repeat $k - 1$ more times:
    - Pick random data point $\vec{x}$ according to distribution

      $$\mathbb{P}(\vec{x}) \propto \min_{\vec{\mu} \in C} \|\vec{x} - \mu\|^2$$

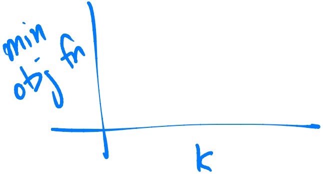
    - Add $\vec{x}$ to C

Picking k

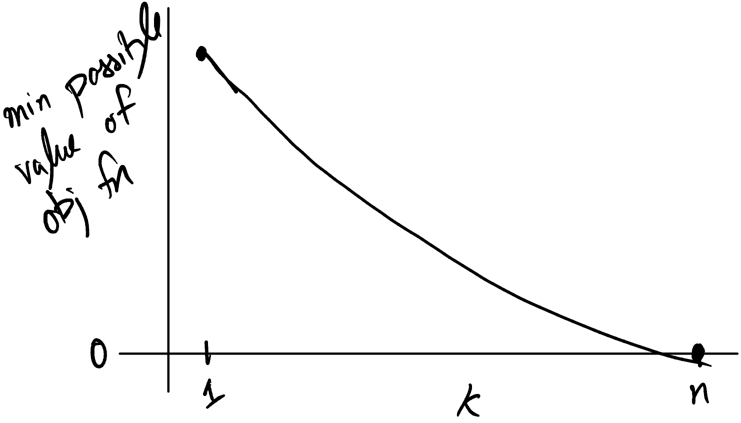
- How do we know how many clusters the data contains?

Exercise

How does the **minimum** of the k-means objective function change as k is increased (that is, as we allow more clusters)?
Hint: What is the minimum when $k = n$ (we have one cluster per data point).



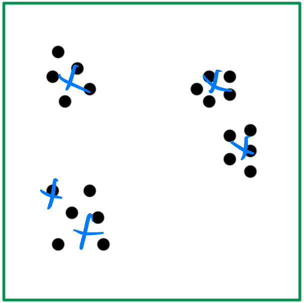
Plot of K-Means Objective

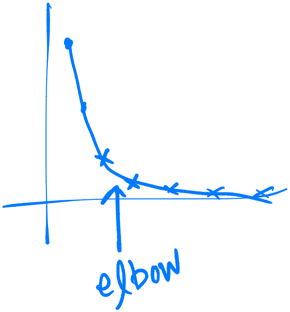


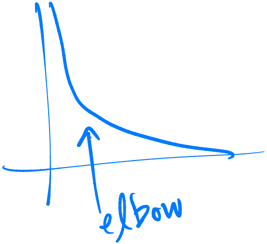
Observation

- Increasing *k* **always** decreases minimum value of objective function

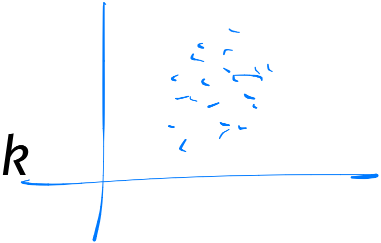
- But increasing *k* beyond “true” number of clusters has diminishing returns.



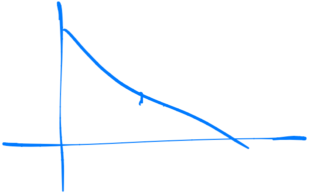




Picking *k*



- The **elbow method**:
  - Run *k*-means repeatedly with increasing values of *k*
  - Plot the value of the objective as a function of *k*
  - Find an **elbow** in the plot



Applications of K-Means

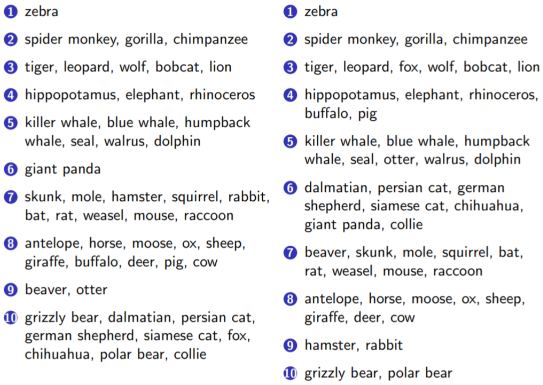
- Discovery

- Vector Quantization
  - Find a finite set of representatives of a large (possibly infinite) set.

Example

- Cluster animal descriptions.
- 50 animals: grizzly bear, dalmatian, rabbit, pig, …
- 85 attributes: long neck, tail, walks, swims, …
- 50 data points in $\mathbb{R}^{85}$. Run k-means with $k = 10$

Results



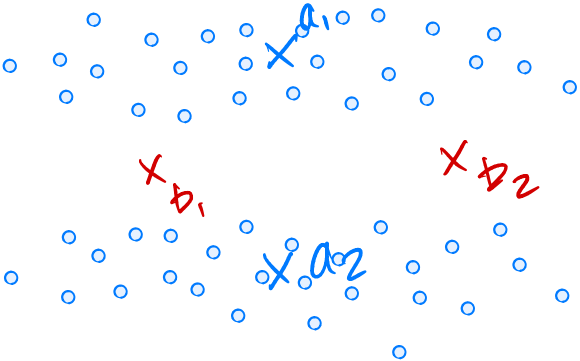
1. zebra
2. spider monkey, gorilla, chimpanzee
3. tiger, leopard, wolf, bobcat, lion
4. hippopotamus, elephant, rhinoceros
5. killer whale, blue whale, humpback whale, seal, walrus, dolphin
6. giant panda
7. skunk, mole, hamster, squirrel, rabbit, bat, rat, weasel, mouse, raccoon
8. antelope, horse, moose, ox, sheep, giraffe, buffalo, deer, pig, cow
9. beaver, otter
10. grizzly bear, dalmatian, persian cat, german shepherd, siamese cat, fox, chihuahua, polar bear, collie

1. zebra
2. spider monkey, gorilla, chimpanzee
3. tiger, leopard, fox, wolf, bobcat, lion
4. hippopotamus, elephant, rhinoceros, buffalo, pig
5. killer whale, blue whale, humpback whale, seal, otter, walrus, dolphin
6. dalmatian, persian cat, german shepherd, siamese cat, chihuahua, giant panda, collie
7. beaver, skunk, mole, squirrel, bat, rat, weasel, mouse, raccoon
8. antelope, horse, moose, ox, sheep, giraffe, deer, cow
9. hamster, rabbit
10. grizzly bear, polar bear

K-Means

- Perhaps the most popular clustering algorithm.
- **Fast, easy to understand**.
- **Assumes spherical clusters**.

Example



DSC 190

Machine Learning: Representations

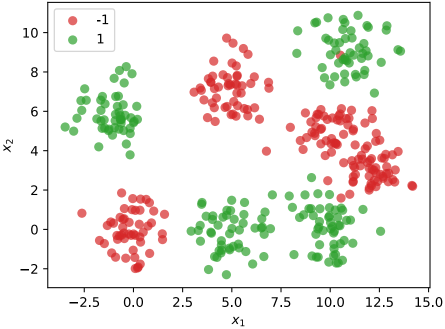
Lecture 5 | Part 3

K-Means for Finding RBF Centers

Idea

- Use *k*-means centers as RBF centers.

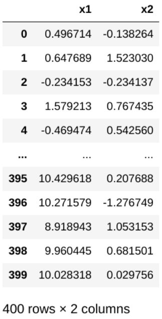
- Typically “over-cluster” by setting *k* to be large.



Workflow

- “Over-cluster” with k-means to find centers
- Create new features using k RBFs
- Fit a least squares classifier

The Data



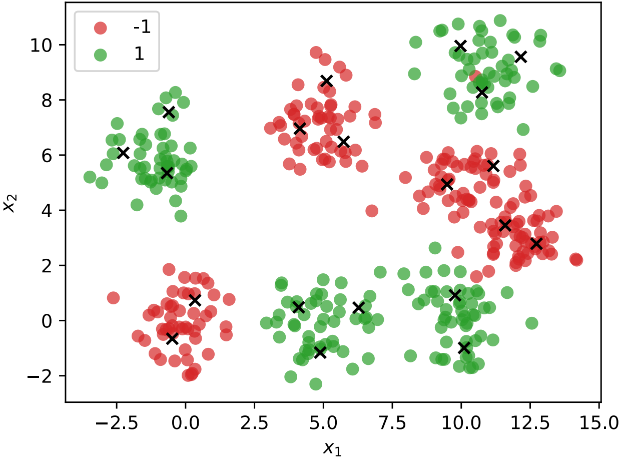
	x1	x2
0	0.496714	-0.138264
1	0.647689	1.523030
2	-0.234153	-0.234137
3	1.579213	0.767435
4	-0.469474	0.542560
...	...	...
395	10.429618	0.207688
396	10.271579	-1.276749
397	8.918943	1.053153
398	9.960445	0.681501
399	10.028318	0.029756

400 rows × 2 columns

Step 1) k-means

```
>>> import sklearn.cluster
>>> # let's start with 20 clusters
>>> kmeans = sklearn.cluster.KMeans(n_clusters=20)
>>> kmeans.fit(data)
>>> cluster_centers = kmeans.cluster_centers_
>>> cluster_centers.shape
(20, 2)
>>> cluster_centers
array([[ 4.10556507,  0.48176175],
       [ 9.48493465,  4.93921129],
       [-0.67384089,  5.34791854],
       [12.73048608,  2.78757872],
       [12.16178039,  9.56285306],
       [ 4.14585321,  6.95969919],
       ...
```

Cluster Centers



Step 2) Create New Features

- We've found $k = 20$ cluster centers, $\vec{\mu}^{(1)}, \ldots, \vec{\mu}^{(20)}$.

- Center a Gaussian RBF at each.

- Take $\sigma = 3$ for now.

- We have $k = 20$ basis functions → 20 newfeatures for every data point $\vec{x}$.

Creating the Features

```
def make_phi(center, sigma):
    def phi(x):
        return np.exp(
            -np.linalg.norm(x - center, axis=1)**2 / sigma**2
            )
    return phi

phis = [make_phi(center, 3) for center in cluster_centers]
```

Example

```
>>> phi_0 = phis[0]
>>> phi_0(np.array([[1, 2], [4, 5], [6, 7]]))
array([0.26507796, 0.10336246, 0.00597847])
```

Applying the Basis Functions

```
>>> new_features = np.column_stack([phi(data) for phi in phis])
>>> new_features.shape
(400, 20)
```

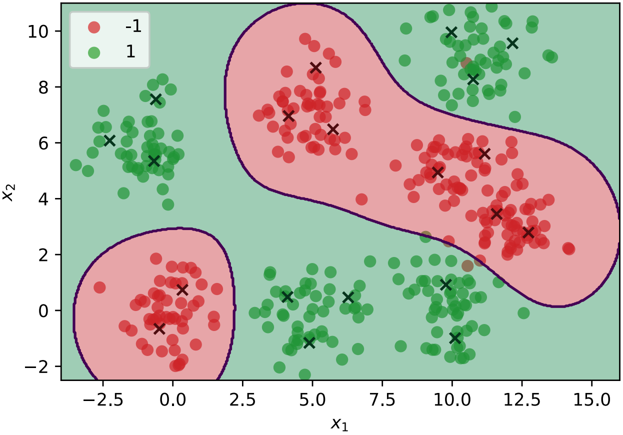
Step 3) Fitting the classifier

```python
def augment(X):
    return np.column_stack((
        np.ones(len(X)),
        X
    ))
```

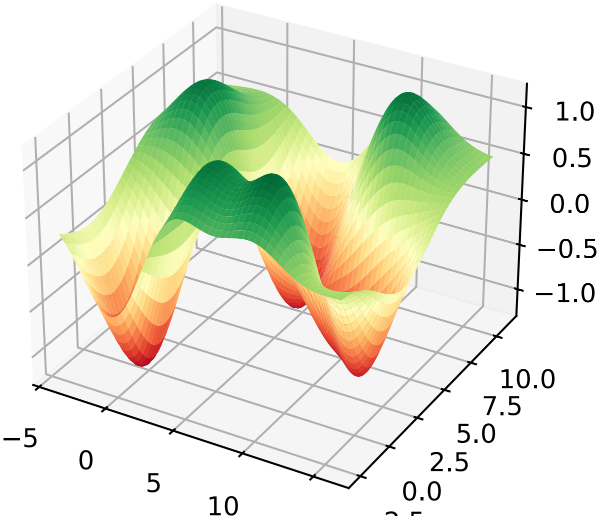
Fitting the Classifier

```
>>> X = augment(new_features)
>>> w = np.linalg.lstsq(X, y)[0]
>>> predictions = np.sign(X @ w)
>>> # training accuracy
>>> (predictions == y).mean()
0.995
```

Decision Boundary



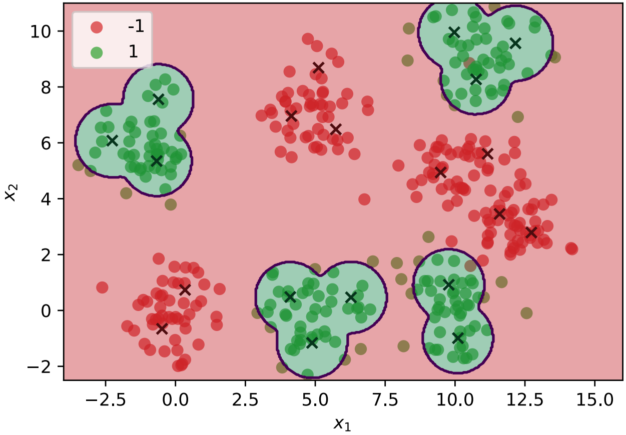
Prediction Surface



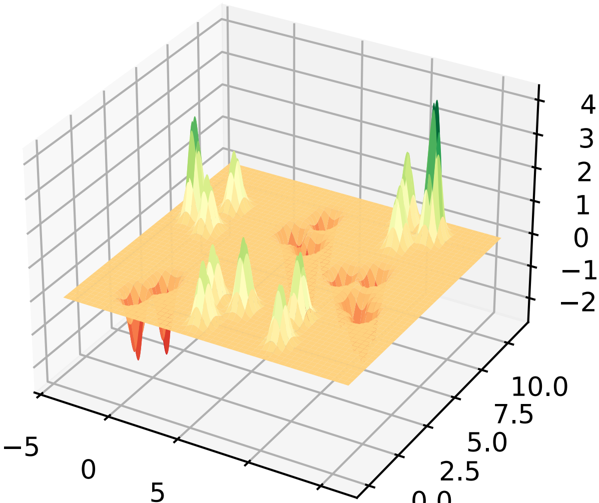
What if...

- What if we make σ smaller?
- Set $\sigma = \frac{1}{2}$

Small Sigma



Small Sigma



Model Complexity

- RBF network complexity is determined by:
  - Number of basis functions (more = more complex)
  - RBF width parameter (smaller = more complex)
- Choose via cross validation
- More complex = greater danger of overfitting

Representation Learning

- This class is about “representation learning”
- This is the first time we’ve actually **learned** a representation.
- Previously: chose basis functions by hand.
- Today: we used k-means to **learn** good basis functions using the data.

DSC 190

Machine Learning: Representations

Lecture 5 | Part 4

Linear Algebra: Linear Transformations

And now for something completely different...

- This and the next few lectures will end with linear algebra refreshers.

Vectors

- A vector $\vec{x}$ is an arrow from the origin to a point.

- We can make new arrows by:
    - scaling: $\alpha\vec{x}$
    - addition: $\vec{x} + \vec{y}$
    - both: $\alpha\vec{x} + \beta\vec{y}$

- $\|\vec{x}\|$ is the **norm** (or length) of $\vec{x}$

Linear Combinations

- We can add together a bunch of arrows:

$$\vec{y} = \alpha_1 \vec{x}^{(1)} + \alpha_2 \vec{x}^{(2)} + \ldots + \alpha_n \vec{x}^{(n)}$$

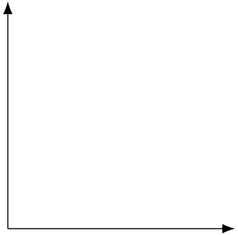
- This is a **linear combination** of $\vec{x}^{(1)}, \ldots, \vec{x}^{(n)}$

Decompositions

- Consider the two vectors, $\vec{u}^{(1)}$ and $\vec{u}^{(2)}$.

- Claim: *any* vector $\vec{x}$ in $\mathbb{R}^2$ can be **decomposed** (written) as $\vec{x} = \alpha\vec{u}^{(1)} + \beta\vec{u}^{(2)}$

- $\vec{u}^{(1)}$ and $\vec{u}^{(2)}$ form a **basis** of $\mathbb{R}^2$

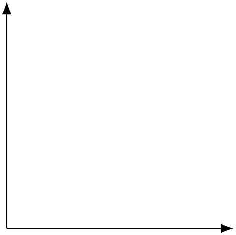


Bases

- There was nothing special about $\vec{u}^{(1)}$ and $\vec{u}^{(2)}$.
- There are **infinitely many** bases of $\mathbb{R}^2$.
- But there is one that is particularly natural…

Standard Basis Vectors

- $\hat{e}^{(1)}$ and $\hat{e}^{(2)}$ are the **standard basis vectors** in $\mathbb{R}^2$.
- We write $\vec{x} = \alpha \hat{e}^{(1)} + \beta \hat{e}^{(2)}$



Coordinate Vectors

- We often write a vector $\vec{x}$ as a **coordinate vector**:

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}$$

- Meaning: $\vec{x} = x_1 \hat{e}^{(1)} + x_2 \hat{e}^{(2)} + \ldots + x_d \hat{e}^{(d)}$

Functions of a Vector

- In ML, we often work with functions of a vector: $f : \mathbb{R}^d \to \mathbb{R}^{d'}$.

- Example: a prediction function, $H(\vec{x})$.

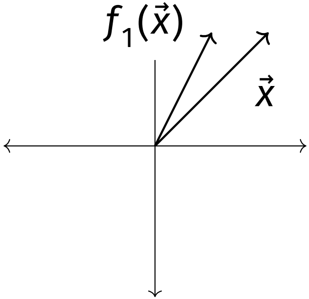
- Functions of a vector can return:
  - a number: $f : \mathbb{R}^d \to \mathbb{R}^1$
  - a vector $f : \mathbb{R}^d \to \mathbb{R}^{d'}$
  - something else?

Transformations

- A **transformation** f is a function that takes in a vector, and returns a vector *of the same dimensionality*.
- That is, $f : \mathbb{R}^d \to \mathbb{R}^d$.

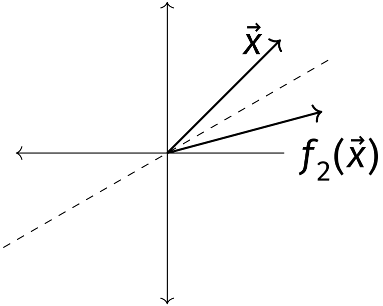
Example

- $f_1(\vec{x})$ halves horizontal component.



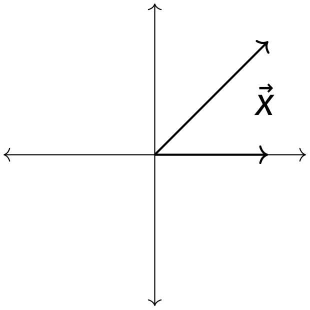
Example

- $f_2(\vec{x})$ flips $\vec{x}$ over the dashed line.



Example

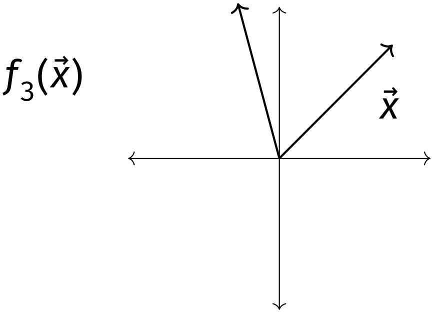
- $f_3(\vec{x})$ projects $\vec{x}$ onto the horizontal axis.



$f_3(\vec{x})$

Example

- $f_4(\vec{x})$ rotates $\vec{x}$ by 45° anticlockwise.



Linear Transformations

- An arbitrary transformation can be quite complex.

- For mathematical ease, we may decide to consider only **linear transformations**.

- A transformation f is linear if:

$$f(\alpha\vec{x} + \beta\vec{y}) = \alpha f(\vec{x}) + \beta f(\vec{y})$$

By the way...

- “Linear” functions, $f(x) = mx + b$, aren’t linear in this sense (unless $b = 0$).

- Rather call these “affine” functions.

Examples

- All of the previous four transformations are linear.

- Another example: $f(\vec{x}) = (x_1 + x_2, x_1 - x_2)^T$

- Non-example: f scales the input by the square of its length.

Main Idea

We use linear functions (and linear transformations) because they are simple and easy to work with mathematically.

The Simplicity of Linear Transformations

- Suppose f is an **arbitrary** transformation.
- I tell you $f(\hat{e}^{(1)}) = (2, 1)^T$ and $f(\hat{e}^{(2)}) = (-3, 0)^T$.
- I tell you $\vec{x} = (x_1, x_2)^T$.
- What is $f(\vec{x})$?

The Simplicity of Linear Transformations

- Suppose f is a **linear** transformation.
- I tell you $f(\hat{e}^{(1)}) = (2, 1)^T$ and $f(\hat{e}^{(2)}) = (-3, 0)^T$.
- I tell you $\vec{x} = (x_1, x_2)^T$.
- What is $f(\vec{x})$?

Exercise

- Suppose f is a **linear** transformation.
- I tell you $f(\hat{e}^{(1)}) = (2, 1)^T$ and $f(\hat{e}^{(2)}) = (-3, 0)^T$.
- I tell you $\vec{x} = (3, -4)^T$.
- What is $f(\vec{x})$?

Key Fact

- Linear functions are determined **entirely** by what they do on the basis vectors.

- I.e., to tell you what f does, I only need to tell you $f(\hat{e}^{(1)})$ and $f(\hat{e}^{(2)})$.

- This makes the math easy!