20.08.2024

Statistical Methods in AI (CS7.403)

Lecture-6: k-Nearest Neighbours and Regression revisited, Clustering (k-means)

Ravi Kiran (ravi.kiran@iiit.ac.in)

https://ravika.github.io

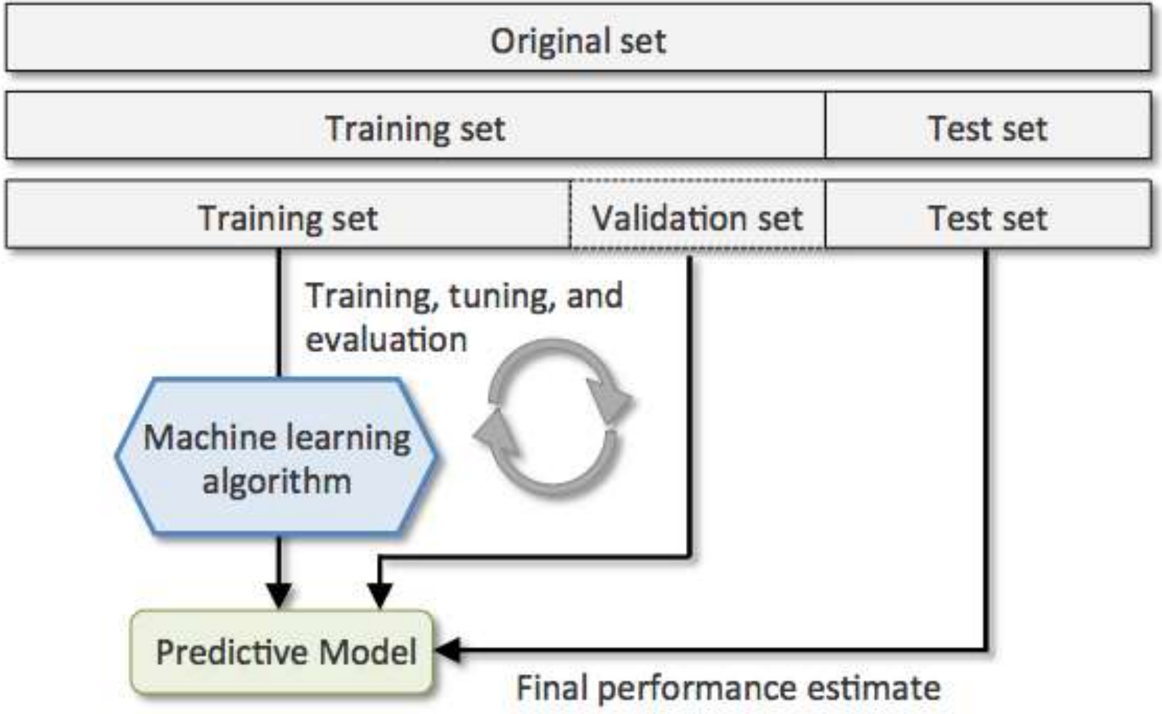
@vikataravi

Center for Visual Information Technology (CVIT)

IIIT Hyderabad

How to choose k in k-NN?

- Rule of thumb: $k < \sqrt{n}$, where n is the number of training examples



Properties and Issues with k-NN

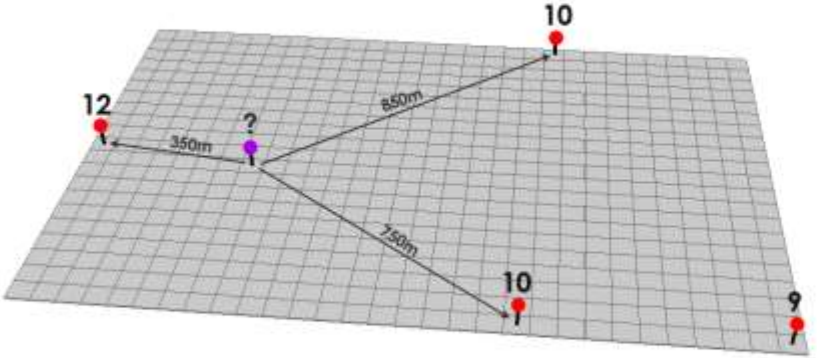
- Non-parametric
- ‘Lazy’ learner
- Simple baseline (after 0-effort baselines)
- GOOD
  - No training
  - Learns highly non-linear decision boundaries
- BAD
  - Need to keep all training points around
  - Curse of dimensionality ! (suggested #dims < 20)

Improving the k-NN Classifier

Faster, Leaner, Meaner

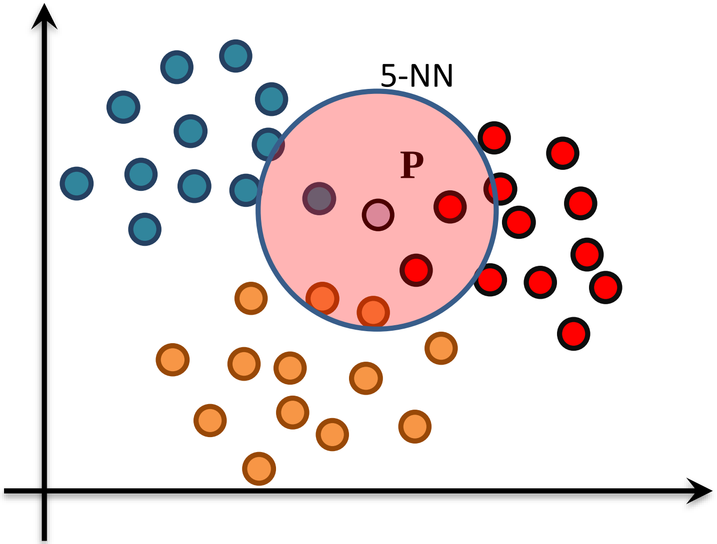
Weighted k-NN

- Helps in case of class skew
- Helps in case of even k (breaking ties)



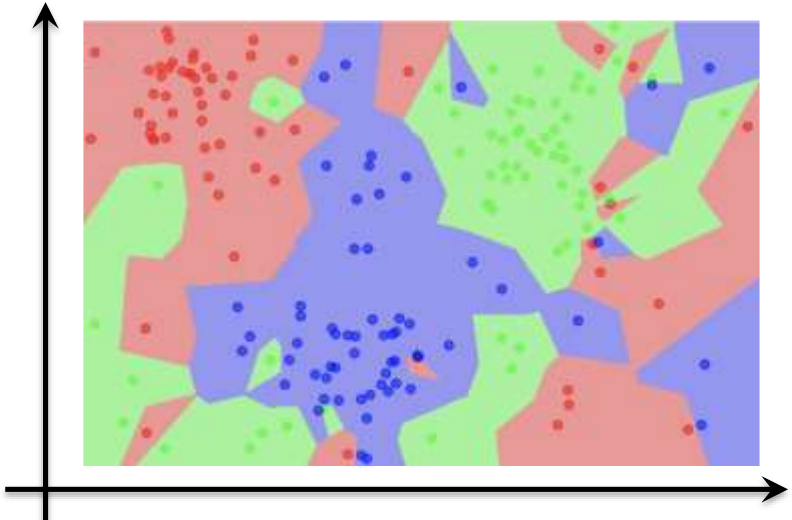
Weighted k-NN

- If there is a tie in majority labels, one can do weighted voting
  - Samples are weighted by inverse of distance to the point p
  - e.g., $w_i = \frac{1}{1+d(p,x_i)}$
  - Example:
    - Blue:
      - Distance: 1
      - Weight: 0.5
    - Orange:
      - Distances: 1.5, 1.6
      - Weight: 0.4 + 0.38 = 0.78
    - Red:
      - Distances: 1.1, 1.3
      - Weight: 0.48 + 0.43 = 0.91
  - ***Label(p)*** = **Red**

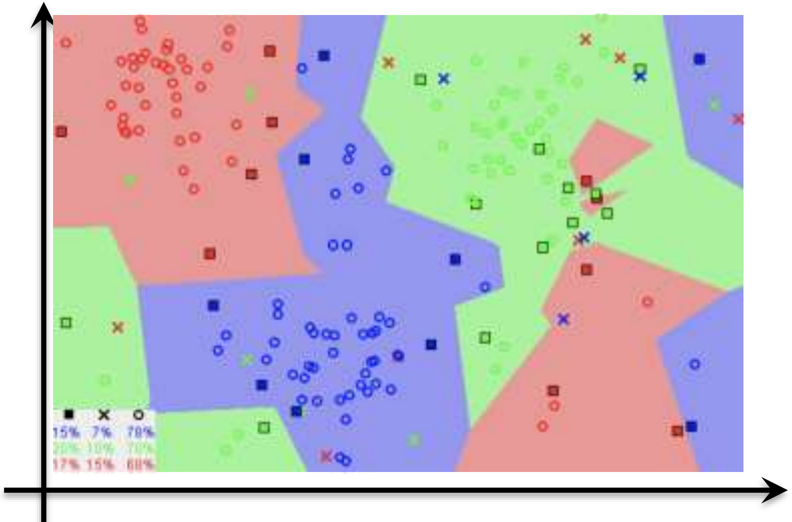


Data Reduction: Condensed NN

- Given a training set X, and the condensed set U ={},
  1. Choose an x whose nearest prototype in U has a different label than x.
  2. Move x from X to U
  3. Repeat until no more prototypes are added to U.
- Use U instead of X for classification.



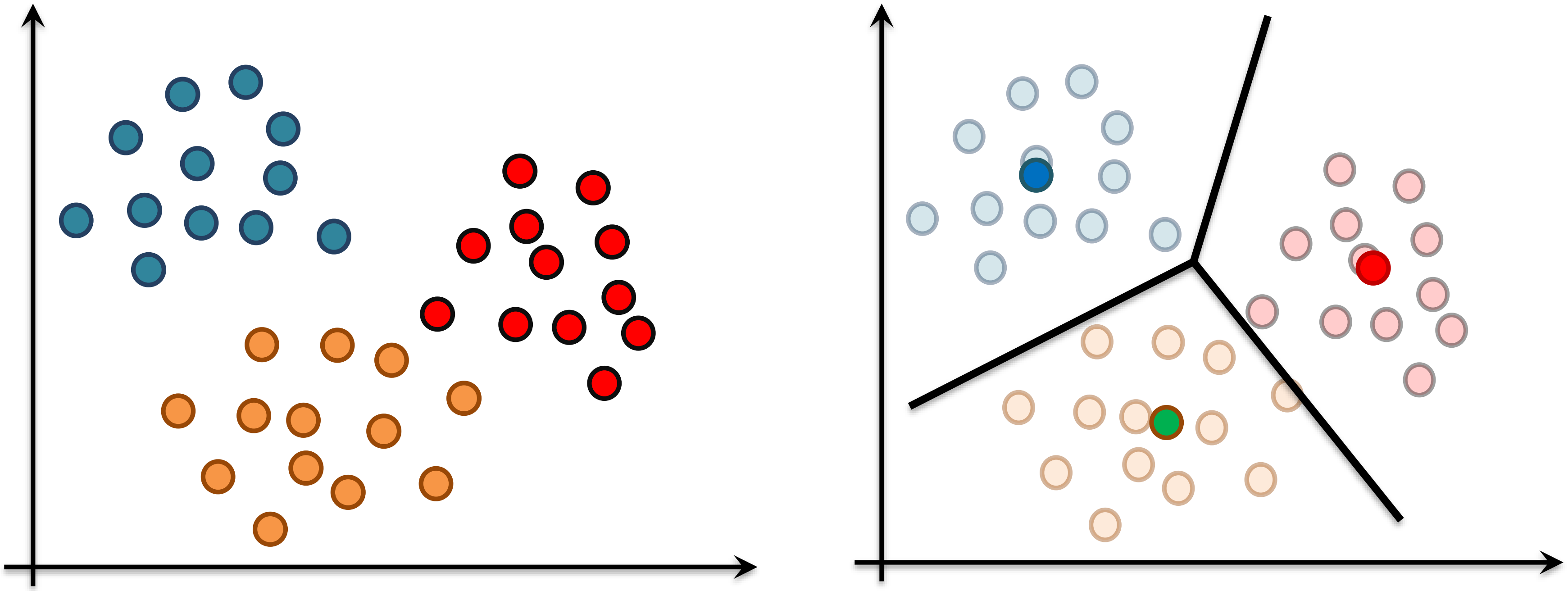
Original Dataset: X



Reduced Dataset: U

Nearest Mean Classifier

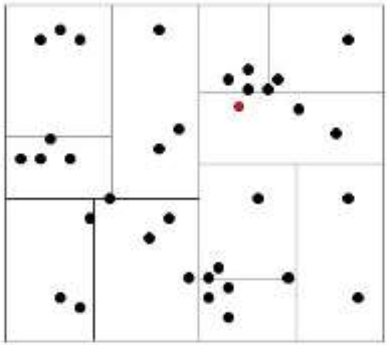
- Represent each class by a single prototype; Its Mean



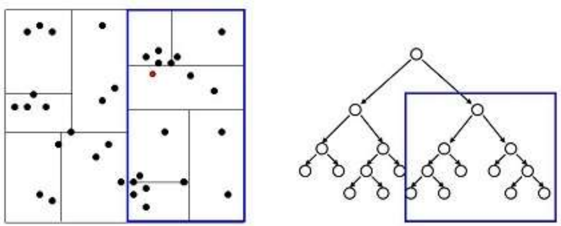
Fast / Approximate Nearest Neighbor

- Quick search for NN with possible errors
- KD Tree
  - Binary space-partitioning trees with axis-parallel splits. Each node is a hyperplane

KD-Tree



Nearest Neighbor with KD Trees

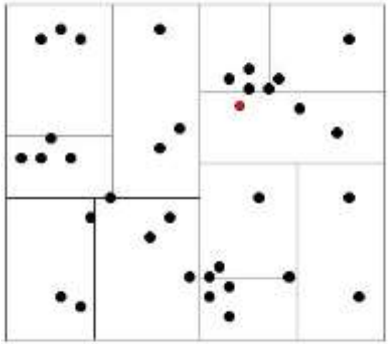


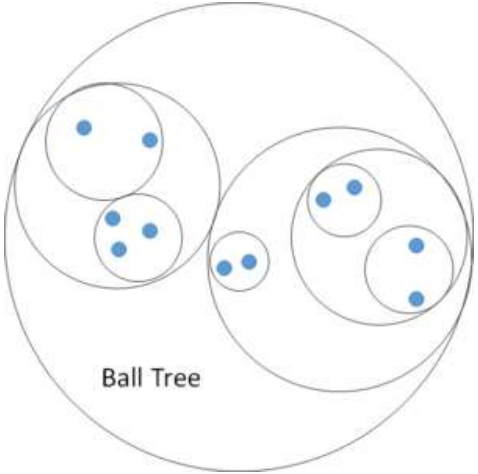
<u>Examine nearby points first</u>: Explore the branch of the tree that is closest to the query point first.

Fast / Approximate Nearest Neighbor

- Quick search for NN with possible errors
- KD Tree
  - Binary space-partitioning trees with axis-parallel splits. Each node is a hyperplane
- Ball Tree
  - Samples are grouped by spheres. Each node has a specific center and radius. Partitioning is based on per-dimension spread.

KD-Tree





scikit-learn Usage

```
>>> from sklearn.neighbors import NearestNeighbors
>>> import numpy as np
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> nbrs = NearestNeighbors(n_neighbors=2, algorithm='ball_tree').fit(X)
>>> distances, indices = nbrs.kneighbors(X)
```

The documentation contains lot of useful details and explanations

https://scikit-learn.org/stable/modules/neighbors.html

Some use cases for k-NN

- Decent performance when lots of data

- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: d = 784
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

Some use cases for k-NN

- Problem: Where (e.g., which country or GPS location) was this picture taken?

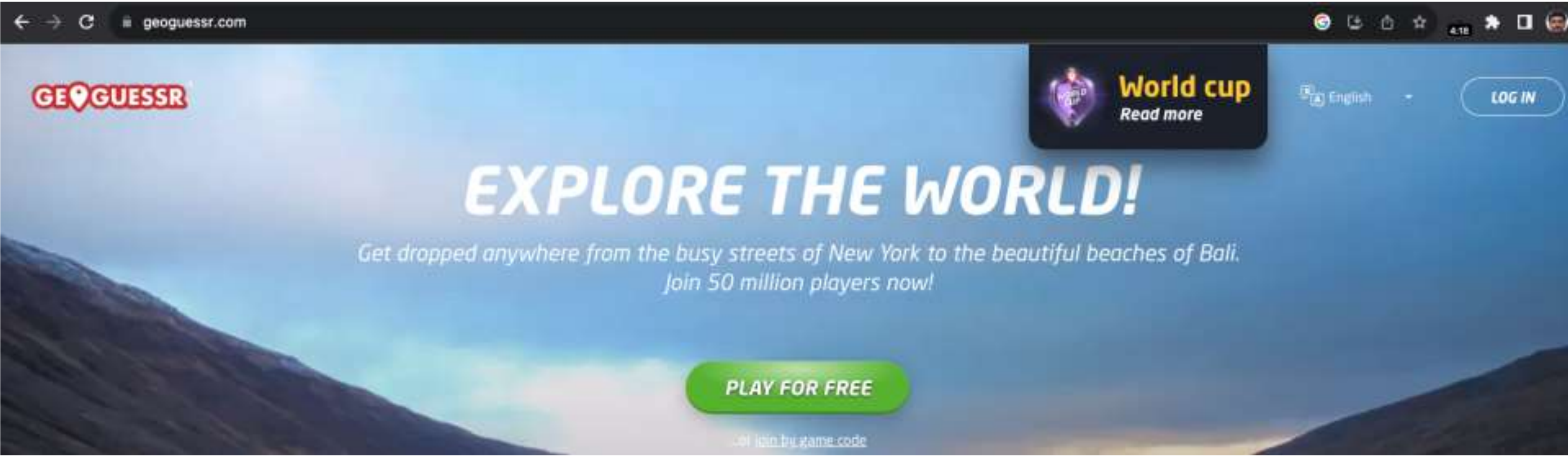
[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

Some use cases for k-NN

- Problem: Where (eg, which country or GPS location) was this picture taken?
  - Get 6M images from Flickr with gps info (dense sampling across world)
  - Represent each image with meaningful features
  - Do kNN (large k better, they use $k = 120$)!

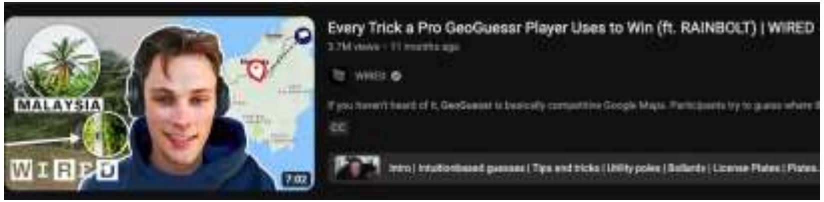
[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

GeoGuessr



geoguessr.com
GEOGUESSR
World cup
Read more
English
LOG IN
EXPLORE THE WORLD!
Get dropped anywhere from the busy streets of New York to the beautiful beaches of Bali.
Join 50 million players now!
PLAY FOR FREE
join by game code

WHO WOULD WIN?



https://www.youtube.com/watch?v=0p5Eb4OSZCs


the Return
of the
KNN

ACL 2023 (#1 NLP conference)

"Low-Resource" Text Classification: A Parameter-Free Classification Method with Compressors

Zhiying Jiang[1,2], **Matthew Y.R. Yang**[1], **Mikhail Tsirlin**[1], **Raphael Tang**[1], **Yiqin Dai**[2] and **Jimmy Lin**[1]

[1] University of Waterloo [2] AFAIK

{zhiying.jiang, m259yang, mtsirlin, r33tang}@uwaterloo.ca
quinn@afaik.io jimmylin@uwaterloo.ca

https://aclanthology.org/2023.findings-acl.426.pdf

Intuition

s1. The sun sets over the calm ocean
s2. A peaceful evening as the sun sets over the sea
s3. Jazz music fills the air as the sun rises in the morning

word	s1	s2	s3
the	2	2	3
sun	1	1	1
sets	1	1	0
over	1	1	0
calm	1	0	0
ocean	1	0	0
a	0	1	0
peaceful	0	1	0
evening	0	1	0
as	0	1	1
sea	0	1	0
jazz	0	0	1
music	0	0	1
fills	0	0	1
air	0	0	1
rises	0	0	1
morning	0	0	1

The sun sets over the calm ocean

A peaceful evening as the sun sets over the sea

Common words/phrases b/w s1 and 2
that LZ77 will compress

https://twitter.com/abhi9u/status/1688150392738385920

```
To classify a sample t:
1 compress t using gzip
2 for each sample s in the training dataset:
  2.1 compress s using gzip
  2.2 compute distance between gzip(s) and gzip(t)
3 find k-nearest neighbours for t based on distances
computed in 2.2
4 pick the majority class as the target label from the k
neighbours
```

To classify a sample t:

1 compress t using gzip

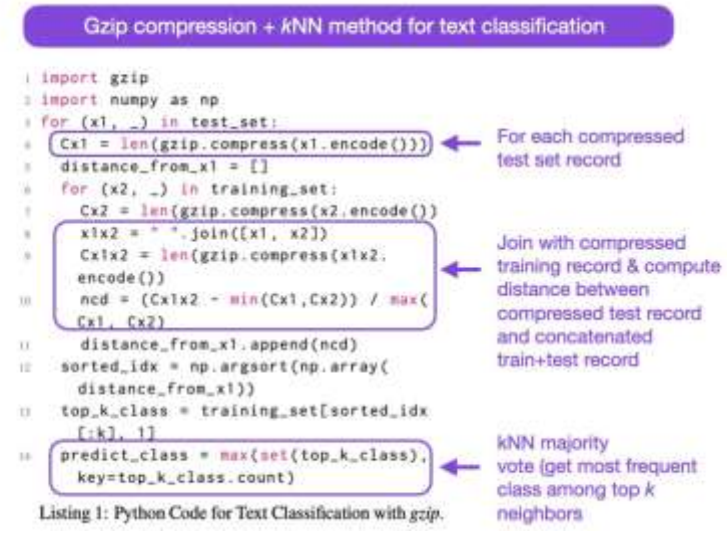
2 for each sample s in the training dataset:

  2.1 compress s using gzip

  2.2 compute distance between gzip(s) and gzip(t)

3 find k-nearest neighbours for t based on distances computed in 2.2

4 pick the majority class as the target label from the k neighbours



Listing 1: Python Code for Text Classification with *gzip*.

https://codeconfessions.substack.com/p/decoding-the-acl-paper-gzip-and-knn

https://pbs.twimg.com/media/F09sVTxWAAEjEZE?format=jpg&name=large

To classify a sample t:

1 compress t using gzip

2 for each sample s in the training dataset:

  2.1 compress s using gzip

  2.2 compute distance between gzip(s) and gzip(t)

3 find k-nearest neighbours for t based on distances computed in 2.2

4 pick the majority class as the target label from the k neighbours

Gzip compression + *k*NN method for text classification

```python
import gzip
import numpy as np
for (x1, _) in test_set:
  Cx1 = len(gzip.compress(x1.encode()))
  distance_from_x1 = []
  for (x2, _) in training_set:
    Cx2 = len(gzip.compress(x2.encode()))
    x1x2 = " ".join([x1, x2])
    Cx1x2 = len(gzip.compress(x1x2.
    encode()))
    ncd = (Cx1x2 - min(Cx1,Cx2)) / max(
    Cx1, Cx2)
    distance_from_x1.append(ncd)
  sorted_idx = np.argsort(np.array(
    distance_from_x1))
  top_k_class = training_set[sorted_idx
    [:k], 1]
  predict_class = max(set(top_k_class),
    key=top_k_class.count)
```

For each compressed test set record

Join with compressed training record & compute distance between compressed test record and concatenated train+test record

kNN majority vote (get most frequent class among top *k* neighbors

Listing 1: Python Code for Text Classification with *gzip*.

Classification accuracy across different test datasets (higher is better)

Red: outperformed by gzip method

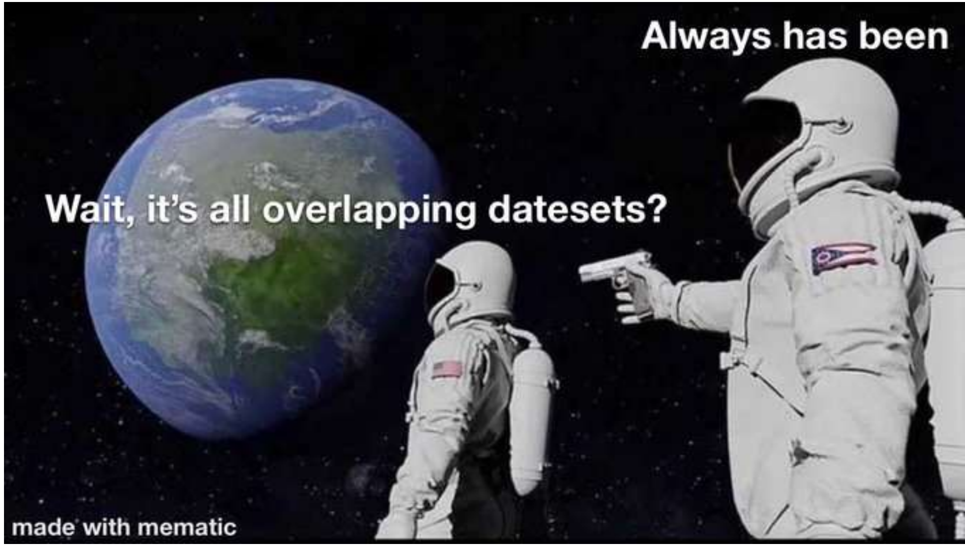
Model	Pre-training	Training	AGNews	DBpedia	YahooAnswers	20News	Ohsumed	R8	R52
TFIDF+LR	✗	✓	0.898	0.982	0.715	0.827	0.549	0.949	0.874
LSTM	✗	✓	0.861	0.985	0.708	0.657	0.411	0.937	0.855
Bi-LSTM+Attn	✗	✓	0.917	0.986	0.732	0.667	0.481	0.943	0.886
HAN	✗	✓	0.896	0.986	0.745	0.646	0.462	0.960	0.914
charCNN	✗	✓	0.914	0.986	0.712	0.401	0.269	0.823	0.724
textCNN	✗	✓	0.817	0.981	0.728	0.751	0.570	0.951	0.895
RCNN	✗	✓	0.912	0.984	0.702	0.716	0.472	0.810	0.773
VDCNN	✗	✓	0.913	0.987	0.734	0.491	0.237	0.858	0.750
fastText	✗	✓	0.911	0.978	0.702	0.690	0.218	0.827	0.571
BERT	✓	✓	0.944	0.992	0.768	0.868	0.741	0.982	0.960
W2V	✓	✗	0.892	0.961	0.689	0.460	0.284	0.930	0.856
SentBERT	✓	✗	0.940	0.937	0.782	0.778	0.719	0.947	0.910
TextLength	✗	✗	0.275	0.093	0.105	0.053	0.090	0.455	0.362
gzip (ours)	✗	✗	0.937	0.970	0.638	0.685	0.521	0.954	0.896

Proposed gzip method

https://codeconfessions.substack.com/p/decoding-the-acl-paper-gzip-and-knn

https://pbs.twimg.com/media/F09sVTxWAAEjEZE?format=jpg&name=large


Complex
problems
have simple,
easy to
understand,
wrong
answers.
Erik Spiekermann



https://twitter.com/yanndubs/status/1681325900170776578

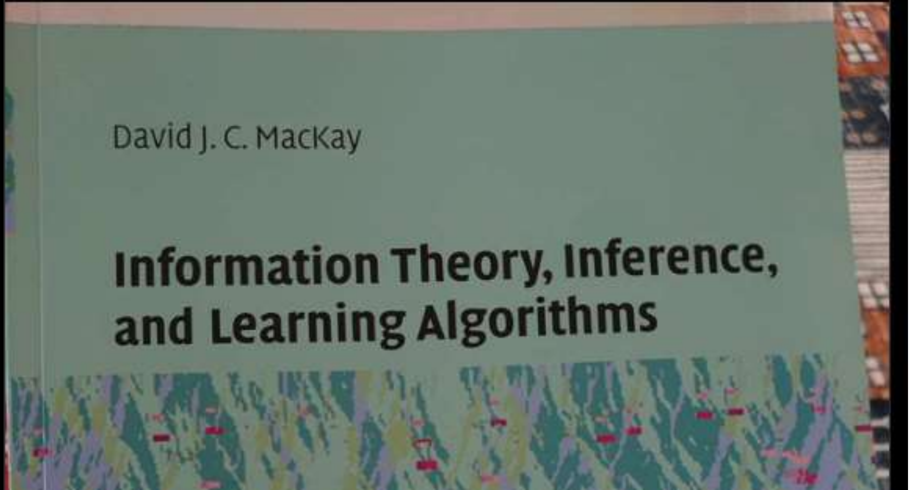
https://twitter.com/predict_addict/status/1681562362569019393

https://twitter.com/yanndubs/status/1681325900170776578

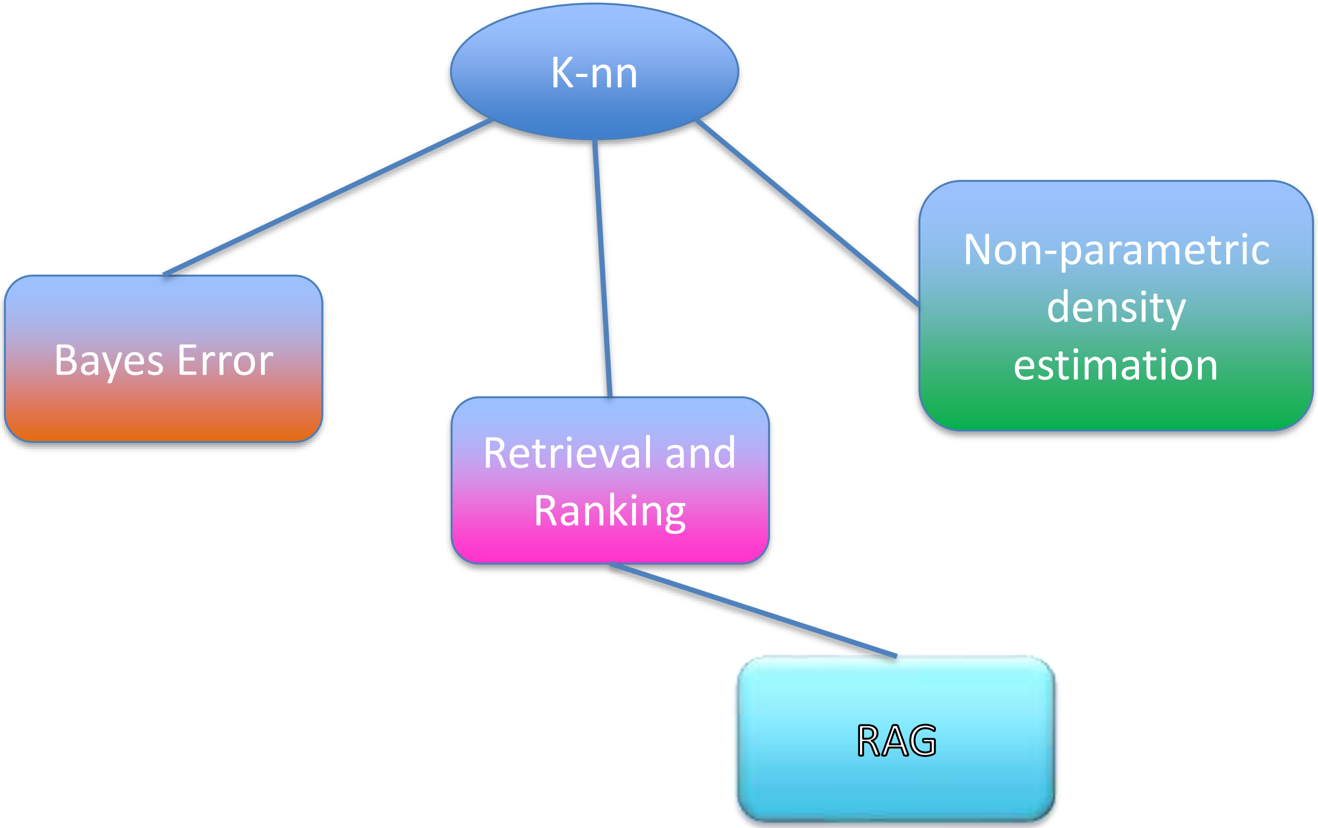
Paul Snively
@paul_snively

Should be much better known that it is. Makes explicit that compression == learning, up to and including that axioms in a logical system are compressed information, and the rules of inference are the "decompression" algorithm.

> **Frank Nielsen** @FrnkNlsn · Aug 12
>
> A very unique textbook "Information theory, inference and learning algorithms" by Sir MacKay combining Information Theory with Machine Learning.
>
> Nicely written!
>
> Book PDF freely available at:
> 👉inference.org.uk/itila/book.html



Related topics



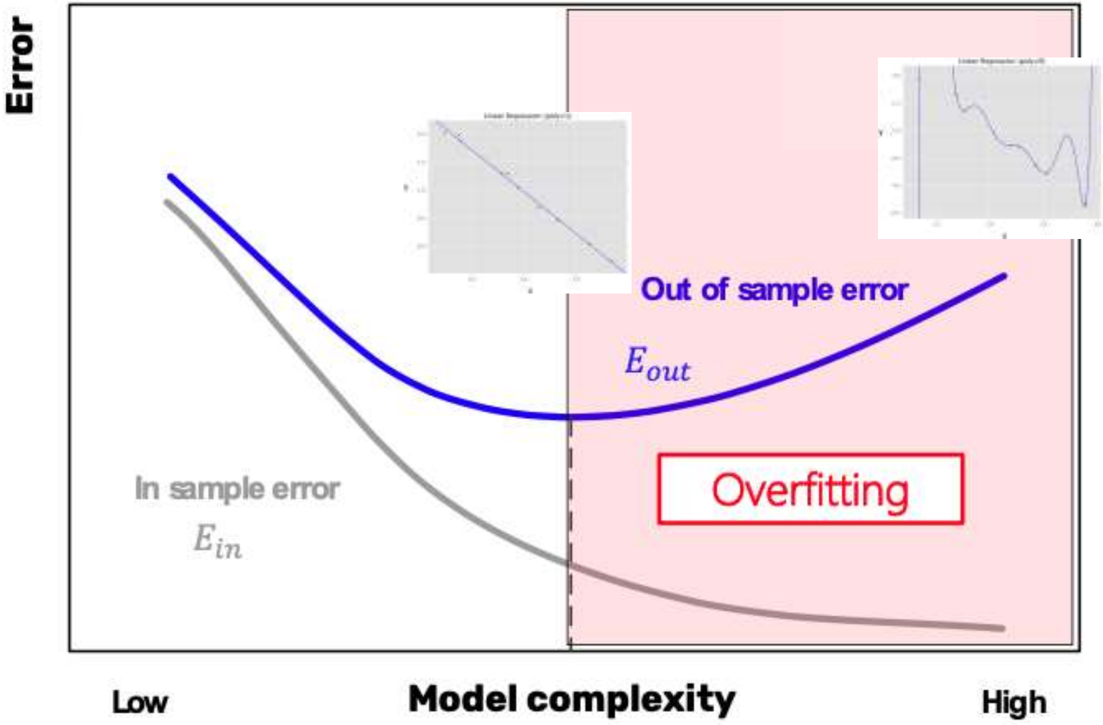
Overfitting vs. model complexity

We talk of **overfitting** when decreasing E_{in} leads to increasing E_{out}

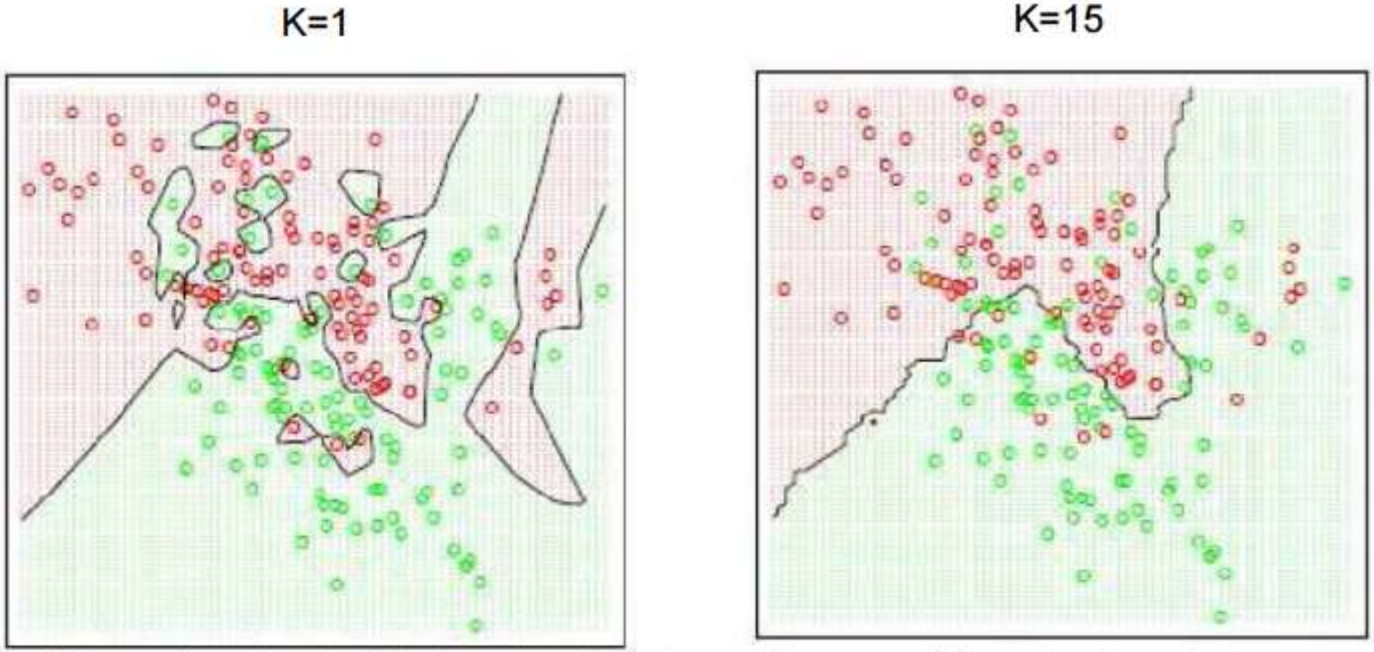
Major source of failure for machine learning systems

Overfitting leads to bad generalization

A model can exhibit bad generalization even if it does not overfit



Overfitting in k-NN



Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

Overfitting

Regularization in k-NN

- Choosing an optimal distance measure and *k*
- Weighted k-NN → smooths the decision boundary
- Feature Scaling → avoids overfitting due to over-reliance on few features
- Dimensionality reduction

Polynomial Regression

- Polynomial of degree k
- Most general case -- includes interaction terms (degree of terms up to k)

Regularization

$h(\cdot)$ is some function that represents our model

More generally, instead of minimizing the in-sample error E_{in} (i.e. the cost function $J(\boldsymbol{\theta})$), minimize the **augmented error:**

For simplicity, suppose $J(\boldsymbol{\theta})$ as squared error function

$$\boxed{E_{aug}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(y(i) - h(\boldsymbol{\varphi}(i);\boldsymbol{\theta})\right)^2 + \lambda \cdot \sum_{j=0}^{d-1}\left(\theta_j\right)^2}$$

- The term $\Omega = \sum_{j=0}^{d-1}\left(\theta_j\right)^2$ is called **regularizer**
- The term λ (**regularization hyper-parameter**) weights the importance of minimizing $J(\boldsymbol{\theta})$, with respect to minimizing Ω

Choice of the regularizer

There are many choices of possible regularizers. The most used ones are:

- L_2 **regularizer:** also called **Ridge** regression $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1}\left(\theta_j\right)^2 = \boldsymbol{\theta}^\top\boldsymbol{\theta} = \|\boldsymbol{\theta}\|_2^2$
- L_1 **regularizer:** also called **Lasso** regression $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1}\left|\theta_j\right| = \|\boldsymbol{\theta}\|_1$
- **Elastic-net regularizer:** $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1}\beta\left(\theta_j\right)^2 + (1-\beta)\sum_{j=0}^{d-1}\left|\theta_j\right|$

The different regularizers behaves differently:

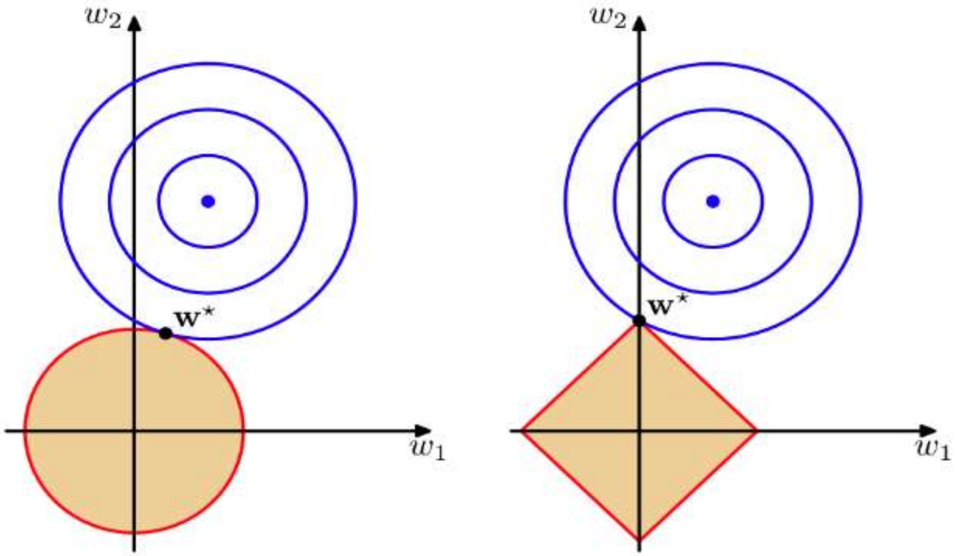
- The ridge penalty tends to shrink all coefficients to a **lower value**
- The lasso penalty tends to set more coefficients **exactly to zero**
- The elastic-net penalty is a compromise between ridge and lasso, with the β value controlling the two contributions

L2 Regularization aka weight decay

$$J_{\text{reg}}(\mathbf{w}) = J(\mathbf{w}) + \frac{\lambda}{2} \sum_{j=1}^{p} w_j^2$$

$$w_j^{(t+1)} = w_j^{(t)} - \eta \frac{\partial J(\mathbf{w})}{\partial w_j}$$

Behavior of regularizers – a visual handwavy explanation



L2 regularization

$$\mathcal{R} = \sum_i w_i^2$$

L1 regularization

$$\mathcal{R} = \sum_i |w_i|$$

Extending Linear Regression to More Complex Models

- The inputs **X** for linear regression can be:
  - Original quantitative inputs
  - Transformation of quantitative inputs
    - e.g. log, exp, square root, square, etc.
  - Polynomial transformation
    - example: $y = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2 + \beta_3 \cdot x^3$
  - Basis expansions
  - Dummy coding of categorical inputs
  - Interactions between variables
    - example: $x_3 = x_1 \cdot x_2$

This allows use of linear regression techniques to fit non-linear datasets.

Linear Basis Function Models

- Generally,

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j \underbrace{\phi_j(\boldsymbol{x})}_{\text{basis function}}$$

- Typically, $\phi_0(\boldsymbol{x}) = 1$ so that θ_0 acts as a bias
- In the simplest case, we use linear basis functions :

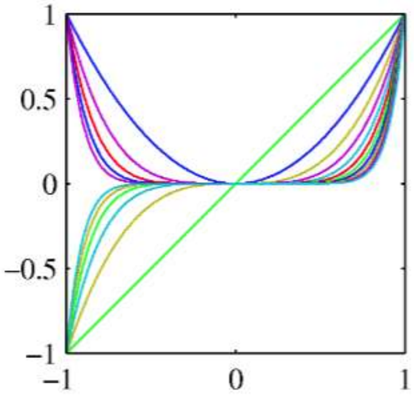
$$\phi_j(\boldsymbol{x}) = x_j$$

Linear Basis Function Models

- Polynomial basis functions:

$$\phi_j(x) = x^j$$

  - These are global; a small change in x affects all basis functions



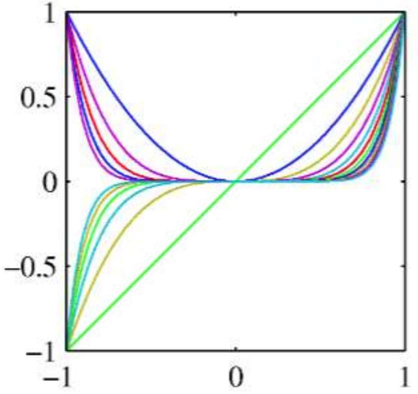
- Sensitive to even minor changes in x → Overfitting
- Unpredictable behavior near input data boundaries → affects extrapolation
- Highly local variations, piecewise functions (e.g. sin + poly) cannot be captured well
- Generally suited when data is limited, or extrapolation is not required

Linear Basis Function Models

- Polynomial basis functions:

$$\phi_j(x) = x^j$$

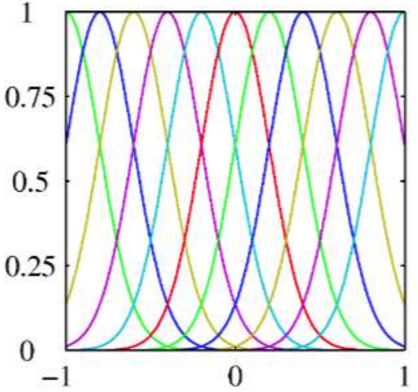
  - These are global; a small change in x affects all basis functions

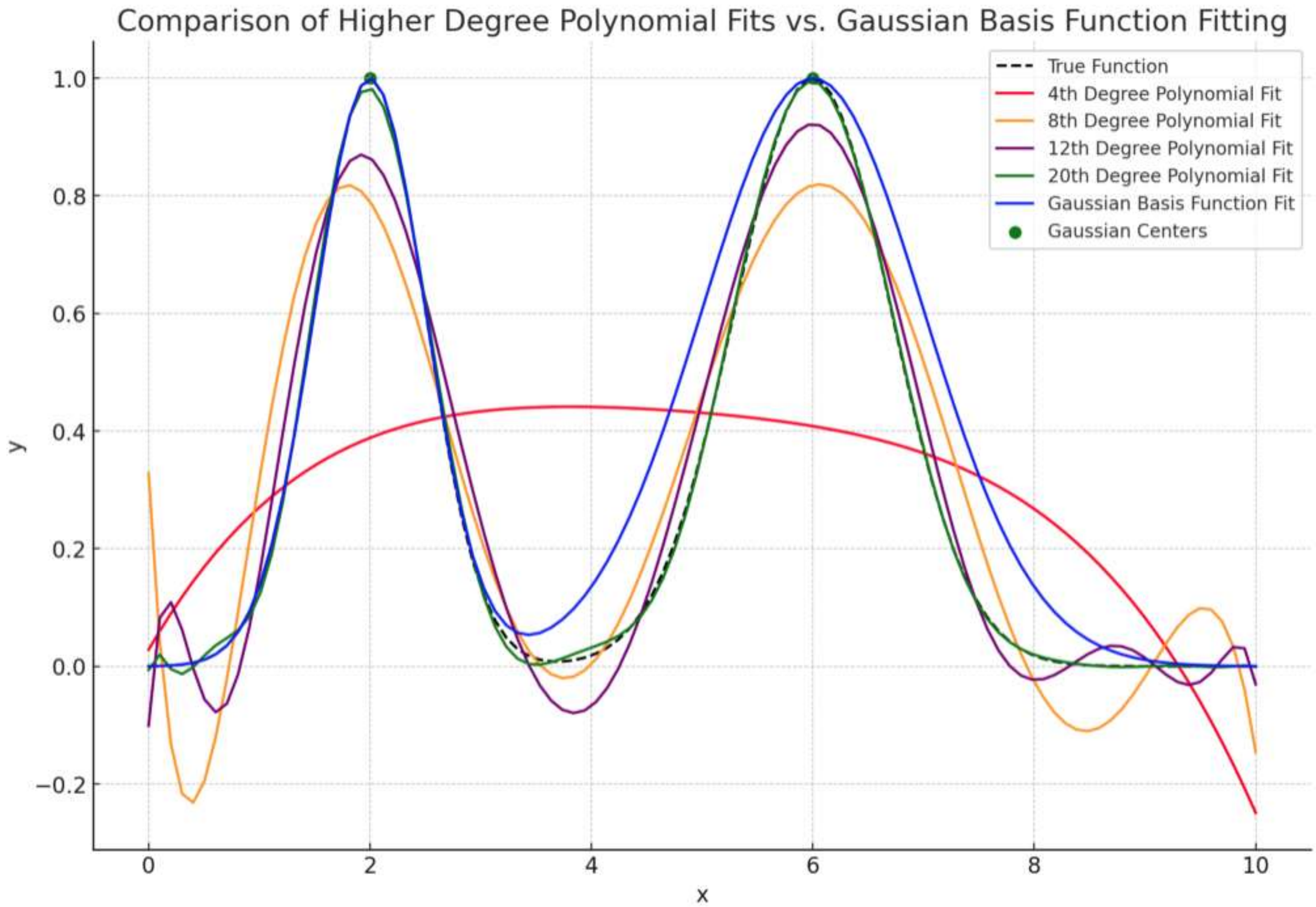


- Gaussian basis functions:

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

  - These are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (width).

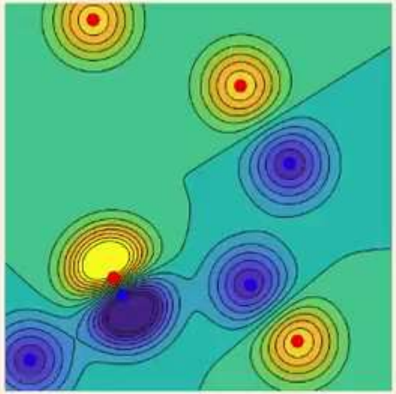



Comparison of Higher Degree Polynomial Fits vs. Gaussian Basis Function Fitting
True Function
4th Degree Polynomial Fit
8th Degree Polynomial Fit
12th Degree Polynomial Fit
20th Degree Polynomial Fit
Gaussian Basis Function Fit
Gaussian Centers
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
y
0
2
4
6
8
10
x

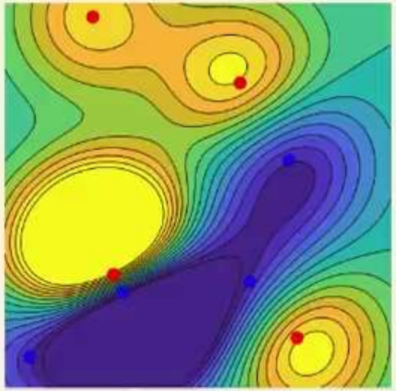
Radial basis functions interpolation: $f(x) \overset{\text{def.}}{=} \sum_i a_i \varphi(\|x - x_i\|)$

$$f(x_j) = y_j \quad \Leftrightarrow \quad \sum_i a_i \varphi(\|x_j - x_i\|) = y_i$$

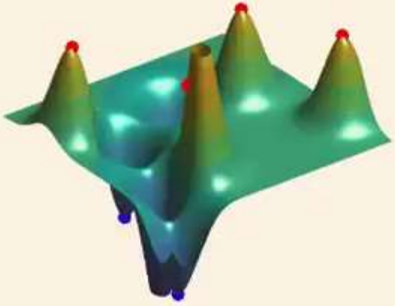
Gaussian: $\varphi(r) = \exp(-\frac{r^2}{2s^2})$.

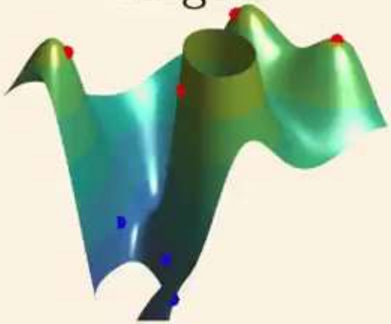


small s

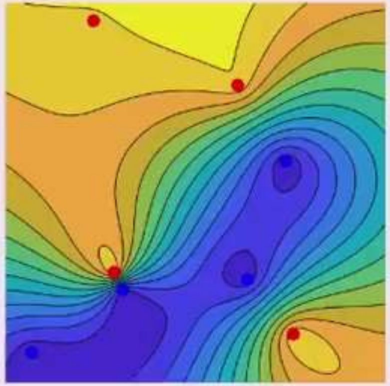


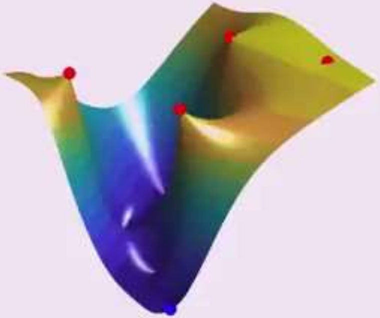
large s





$\varphi(r) = -r$

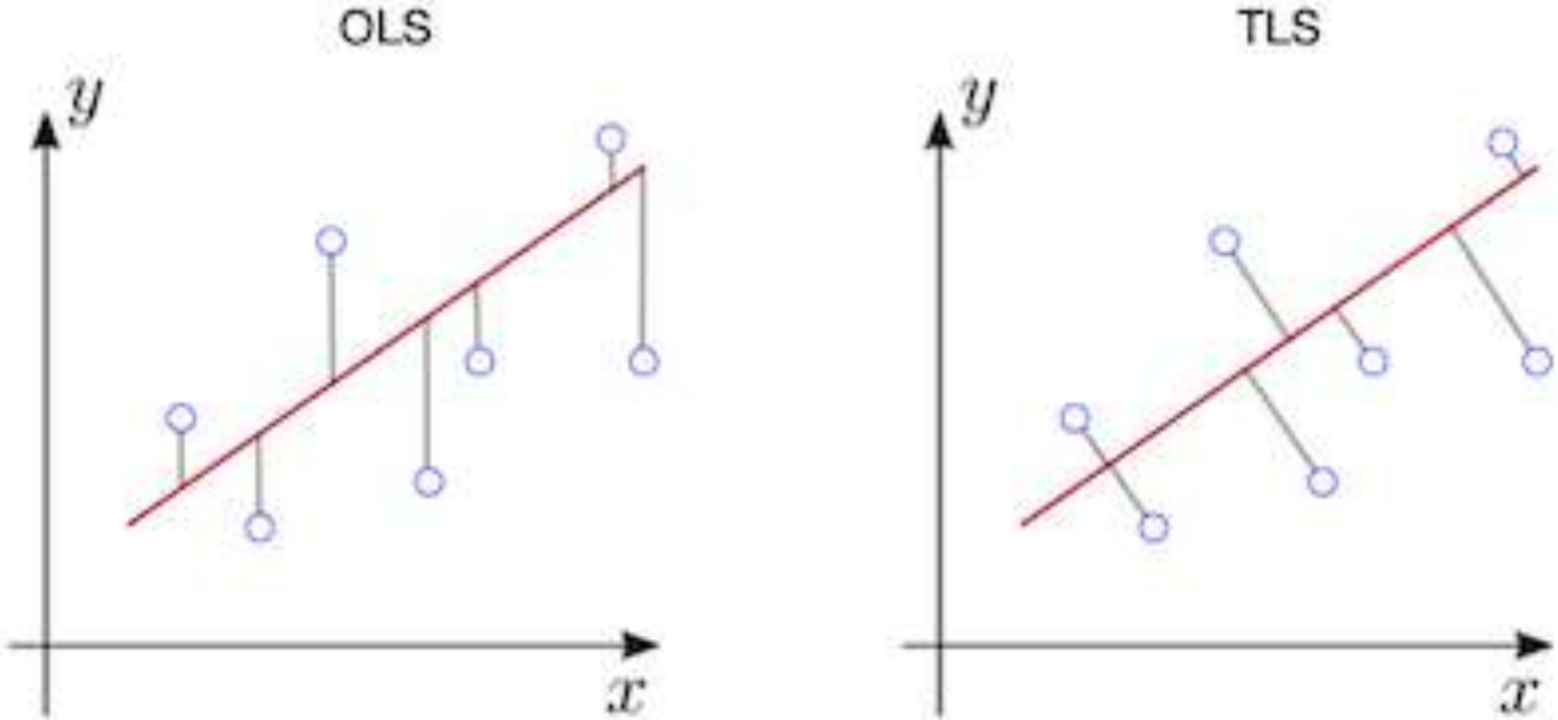




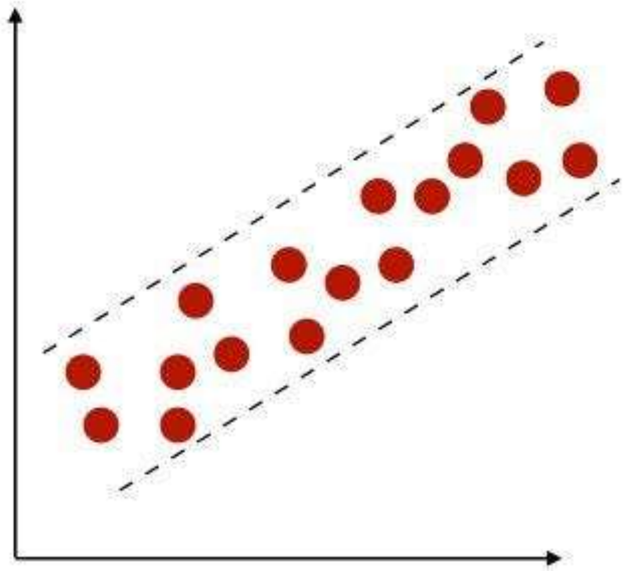
Custom Regularizers

- Weights
  - Sum to 1 (Lagrange Multiplier, Projected Gradient)
  - Lie in [0,1] (Penalty, Clipping, Projected)

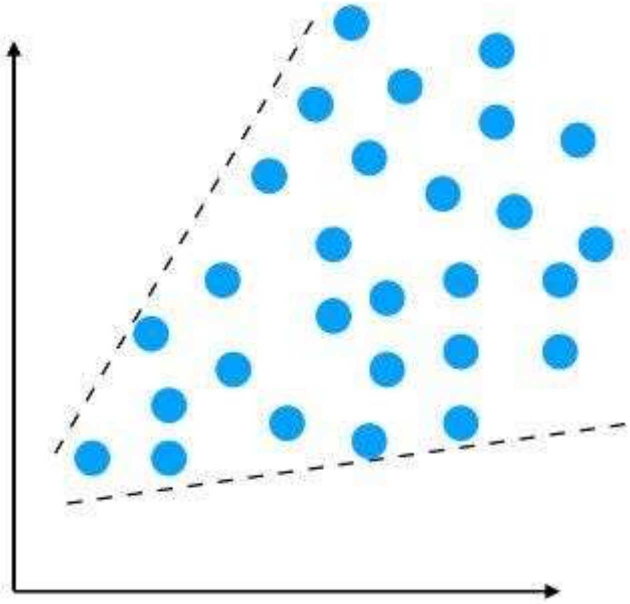
OLS and TLS



Homo/Heteroscedasticity



Homoscedasticity



Heteroscedasticity