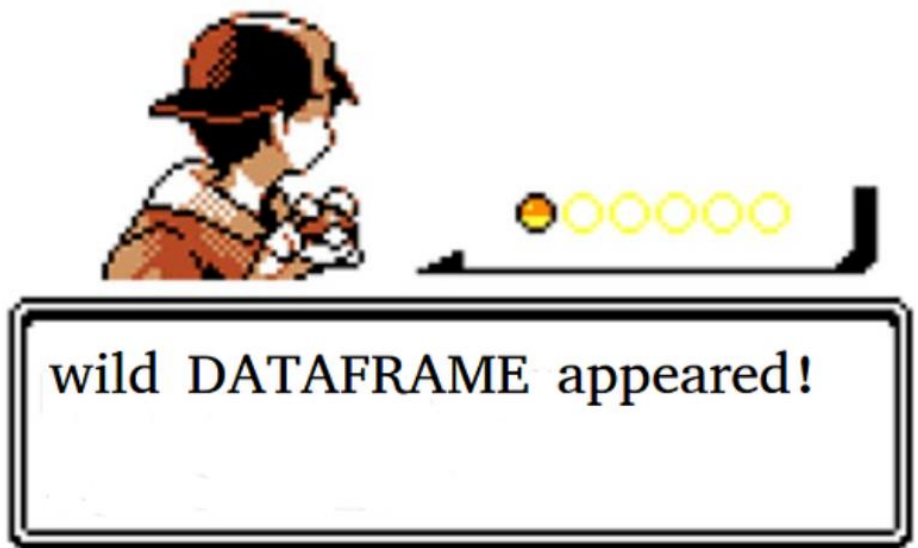
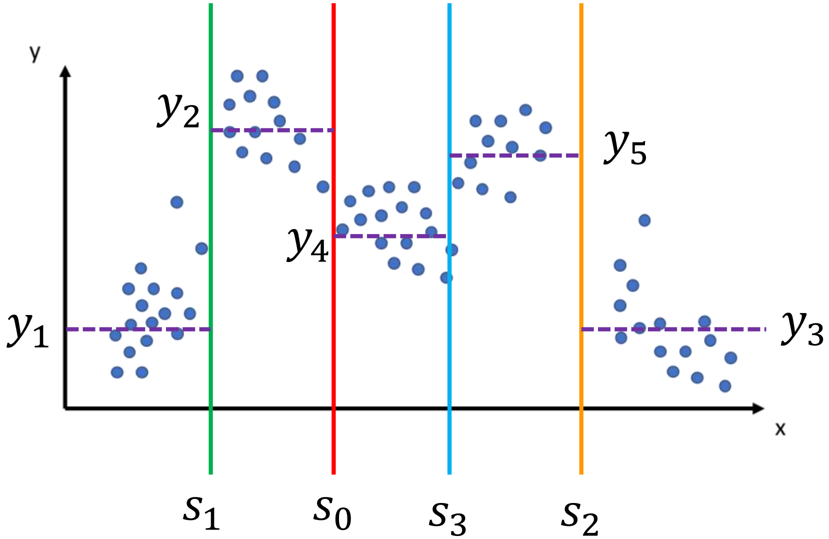
CISC 372
Advanced Data Analytics
Instance-based Learning

	name	age	state	num_children	num_pets
0	john	23	iowa	2	0
1	mary	78	dc	2	4
2	peter	22	california	0	0
3	jeff	19	texas	1	5
4	bill	45	washington	2	0
5	lisa	33	dc	1	0



Tree[s]

- Tree Induction
- Information Gain/Gain Ratio/Gini Index
- ID3, CART, C4.5
- Splitting Numeric Attribute
- Feature Selection (is difficult)
- Random Forest (the easy way)
  - Built-in bootstrap sampling
- Regression Tree
- XGBoost



Monday

- AutoDiff
- Neural Network
  - Nonlinearity
  - Learn feature mapping
- Convolutional Neural Network
  - Computational complexity
  - Position Invariance
  - Receptive Field

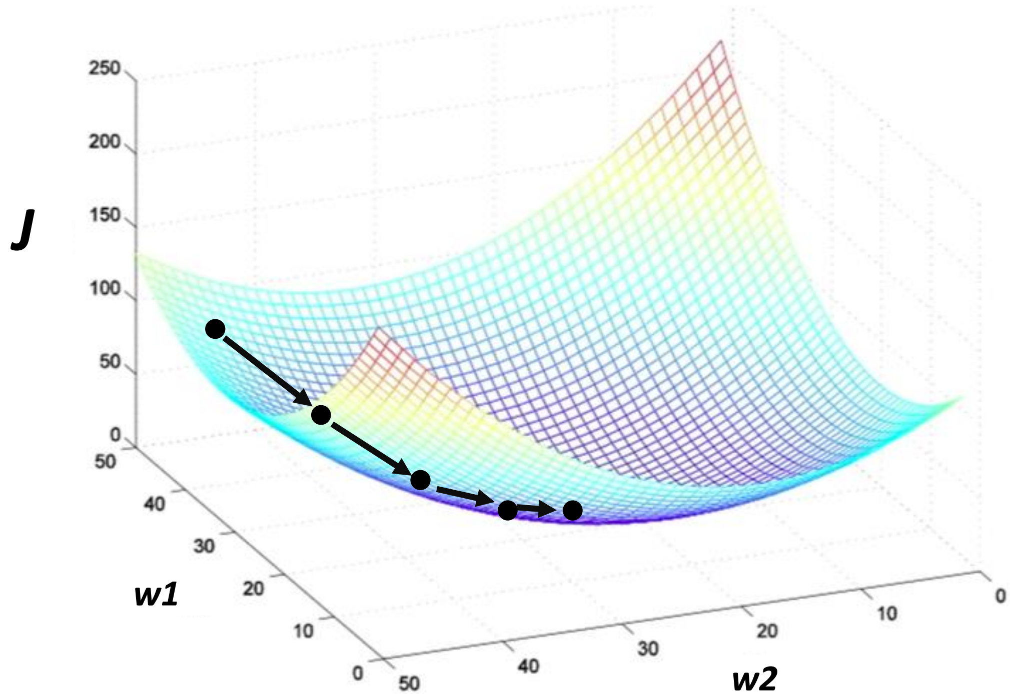
Today

- NN Optimizers
- Parametric vs. Non-parametric model
- Instance-based Learning
- Lazy Learner vs. Eager Learner
- Nearest Neighbor Lookup
- Bayesian Learning

Optimizers

- Gradient Descend (GD)
- Stochastic Gradient Descend (SGD)
- SGD with Momentum
- Adadelta
- Adagrad
- Adam
- RSMProp
- Early Stopping

GD



Total cost:

$$\boldsymbol{J} = \Sigma \, (y' - y)^2$$

Optimizers

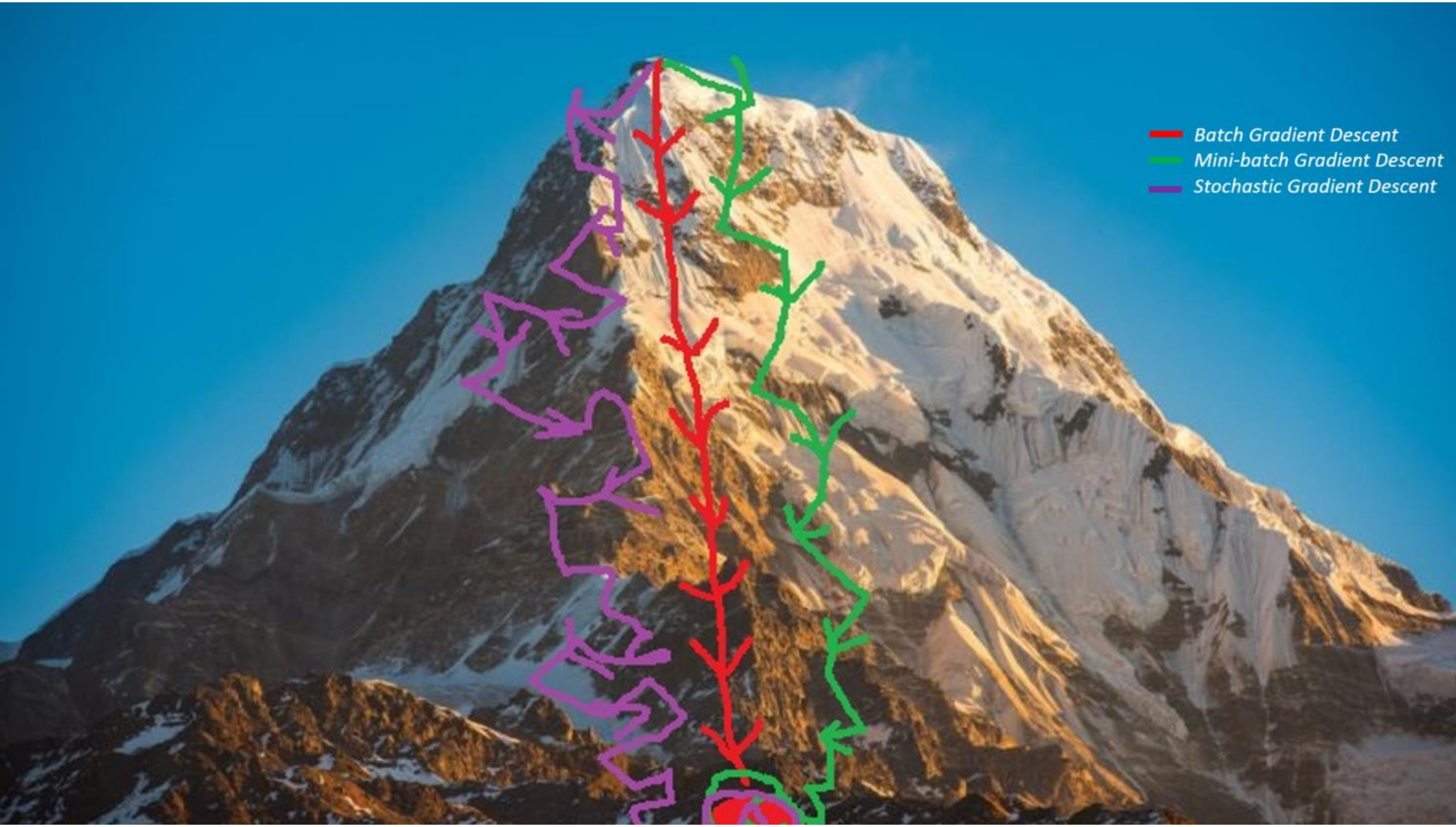
- Gradient Descend (GD)
  - 1 forward-backward pass **with the whole dataset**
- Stochastic Gradient Descend (SGD)
  - SGD
    - 1 forward-backward pass with **1 sample**
    - 1 forward-backward pass with **a mini-batch of sample**
      - A subset that is small enough to fit the memory

Optimizers

- Every backward pass:
  - Calculate Gradient
  - Multiplied by a learning rate (\alpha)

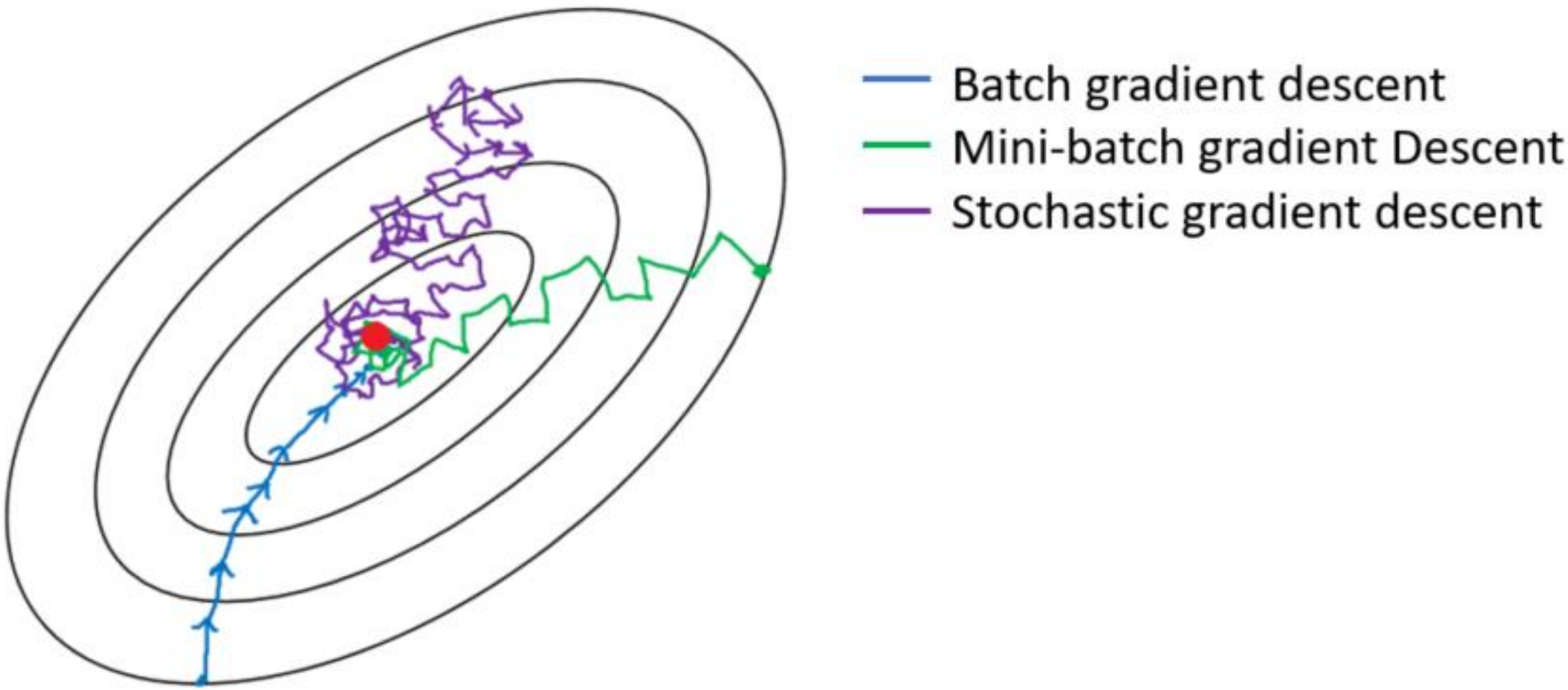
- GD
  - Gradient based on the whole training set => most accurate
  - Mini-batch: subset of samples to calculate gradient => sort-of accurate
  - 1 sample SGD: hm.. A bit high variance. Still can get the job done but just takes a bit longer.

Optimizers



https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

Optimizers



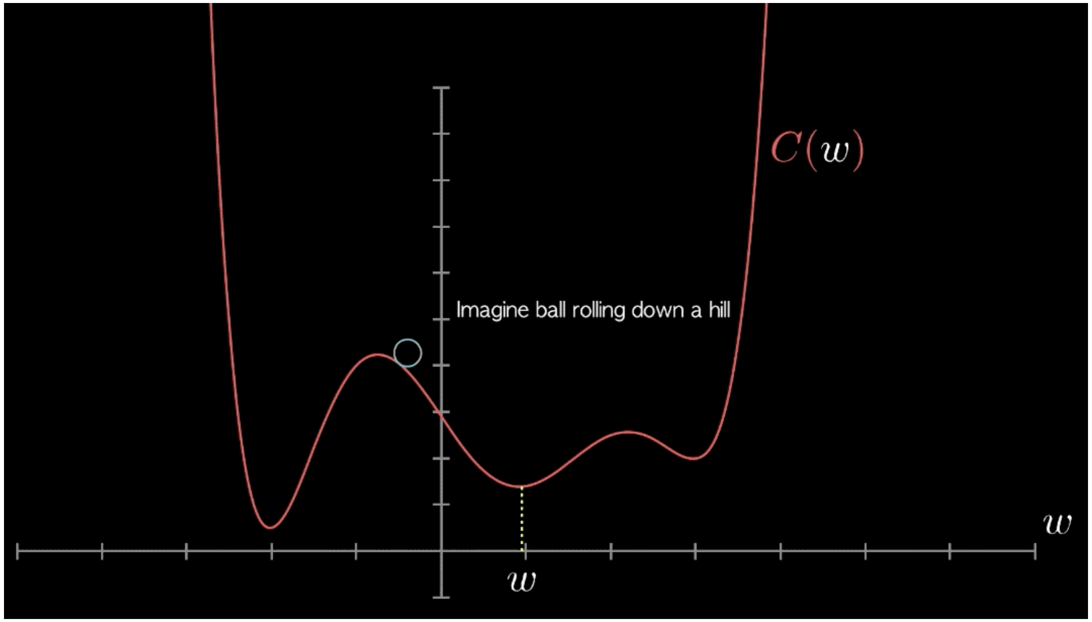
https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

Optimizers – when?

- Gradient Descend (GD)
  - <u>Convex/smooth loss function</u>
  - <u>Small dataset/model</u>
- Stochastic Gradient Descend (SGD)
  - SGD
    - 1 forward-backward pass with **1 sample**
      - <u>Fast. Low memory requirement</u>
    - 1 forward-backward pass with **a mini-batch of sample**
      - Optimal subset that fits the memory

SGD+momentum

- Accelerate!!
  - With the concept of `speed`



- Try this:
  - https://distill.pub/2017/momentum/

SGD+momentum

- How?
  - Use a moving average of gradient.

The modification of the weight @ time/step t

Error/Loss function

$$\Delta \mathbf{w}_t = -\epsilon \nabla_{\mathbf{w}} E(\mathbf{w}) + p \Delta \mathbf{w}_{t-1}$$

Learning Rate

Gradient w.r.t. w

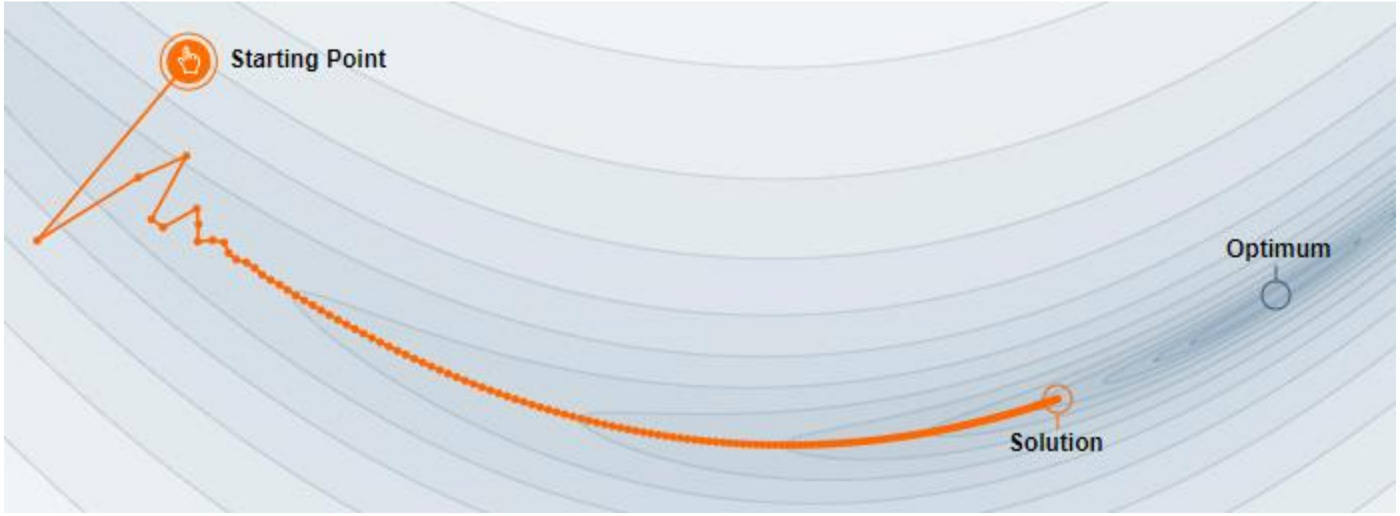
Momentum parameter

$$w_t = w_t + \Delta w_t$$

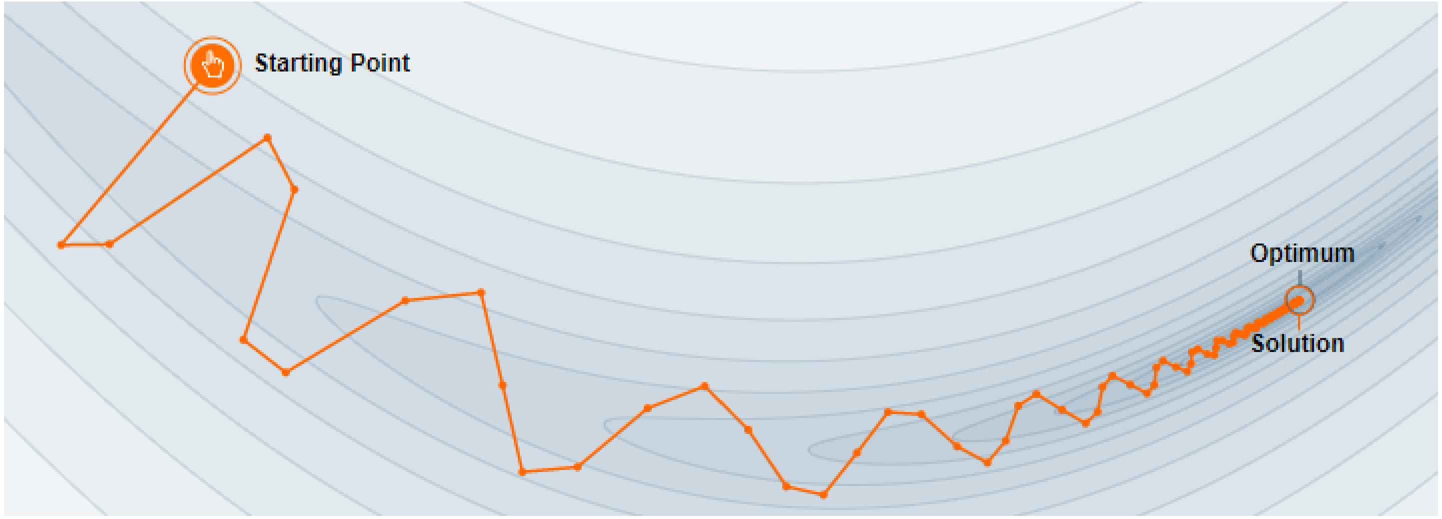
Ning Qian. On the momentum term in gradient descent learning algorithms. Neural networks : the official journal of the International Neural Network Society, 12(1):145–151, 1999

SGD+momentum

Momentum=0.5



Momentum=0.8



Ning Qian. On the momentum term in gradient descent learning algorithms. Neural networks : the official journal of the International Neural Network Society, 12(1):145–151, 1999

Adagrad

- Use the idea of Momentum
- Scale learning rate according to the history of gradient
- Learning rate is reduced if the gradient is very large
- Different learning rate for different parameter
- Normalized by exponentially decaying average of past squared gradients
- **Eliminate the need to manually tune the learning rate**

$$g_t = g_{t-1} + \nabla_w E(w)^2$$

$$w_t = w_t - \frac{\varepsilon}{\sqrt{g_t} + \beta} \nabla_w E(w)^2$$

AdaDelta

- Use the idea of Momentum
- Less aggressive than Adagrad
- Adagrad issues:
  - **Accumulation of the squared gradients**
  - **learning rate is always decreasing**
- AdaDelta tries to solve the above issue
- Restrict the history into a specific size
- **No need to set initial learning rate**

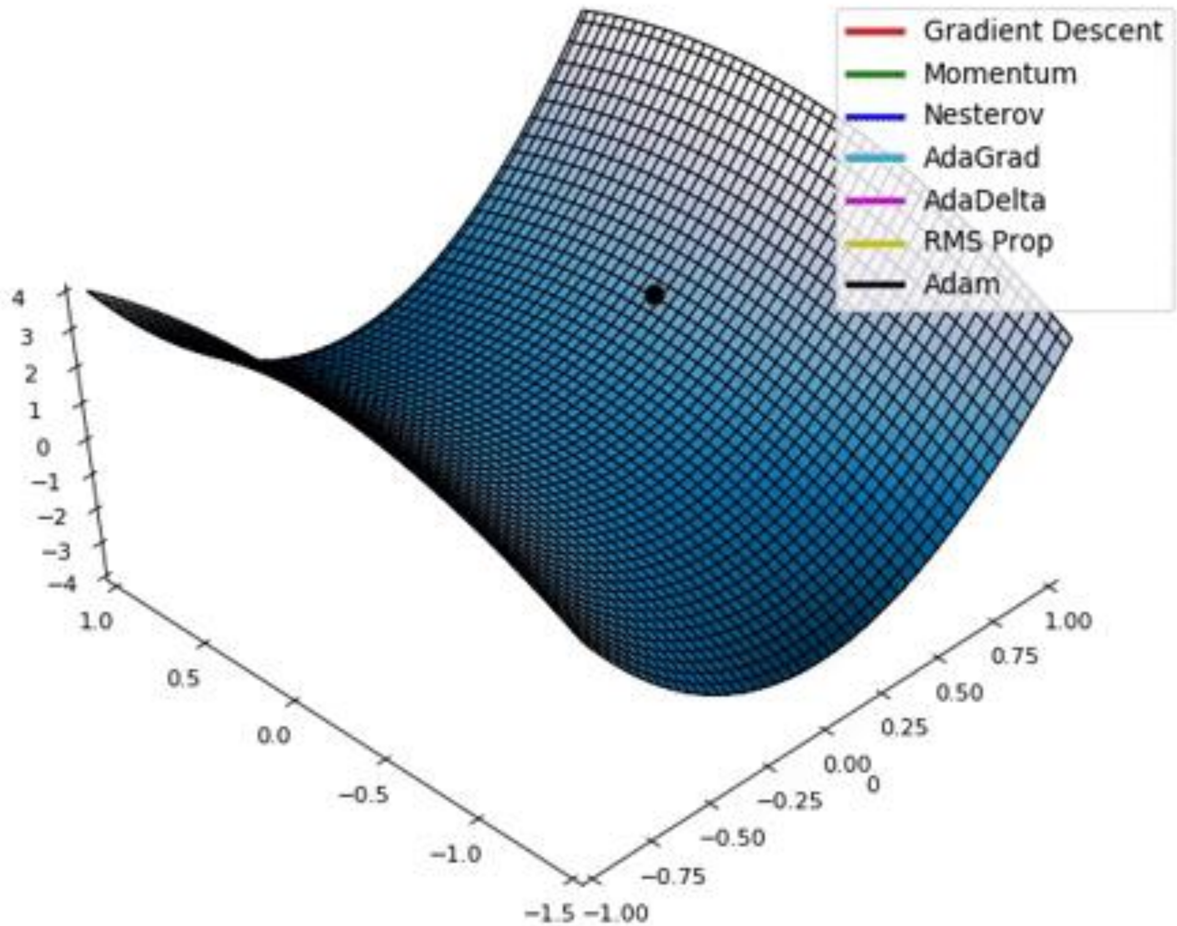
RMSProp

- Use the idea of Momentum
- Also tries to solve Adagrad's issue
- Similar idea to AdaDelta:
  - Normalize learning rate by the magnitudes of recent gradient of a weight.
  - But with different formulations.

Adaptive Moment Estimation (Adam)

- Similar idea to Adadelta and RMSprop
- Keep track of exponentially decaying average of past gradients
- Also keeps an exponentially decaying average of past gradients, similar to momentum

Adaptive Moment Estimation (Adam)



https://rnrahman.com/blog/visualising-stochastic-optimisers/

Parametric models

- Models that are parameterized **by a fixed size vector/matrix**. (Formally, it assumes a finite set of parameters independent of the dataset)

$$P(x|\theta, D) = P(x|\theta)$$

- Model structure (parameters) is **pre-determined**.
  - Linear regression, MLP, Convolutional NN etc.
  - Linear SVM

- Minimize the loss function by adjusting the parameters.

Non-Parametric models

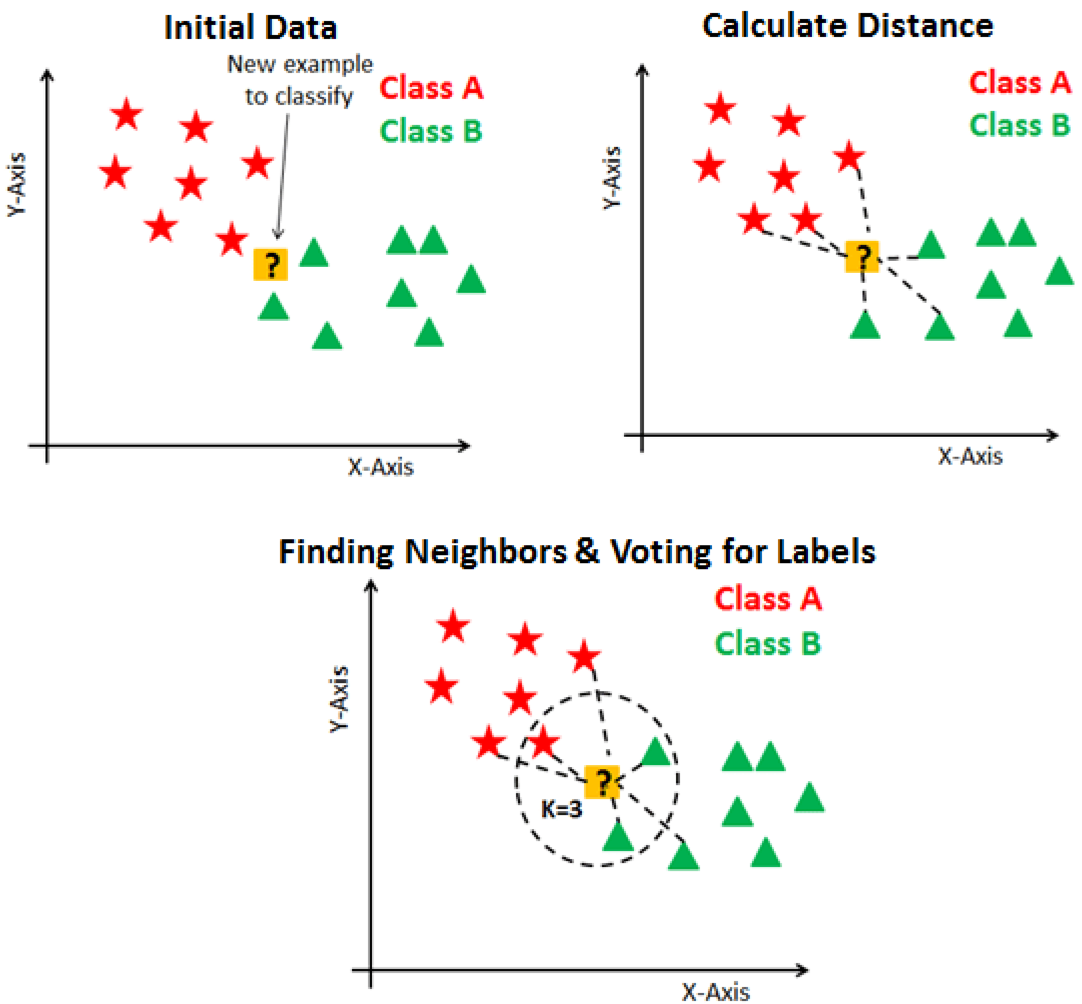
- **NO** parameters at all
  - May have hyperparameters
  - Instance-based learning
- **Or,** No such a prior that dictates the parameterization of the model
  - Still there are parameters
  - Number/Structure of the parameters are flexible
  - Depends on the data
  - Kernel SVM (kernel matrix)
  - Topic Modeling (Part II)

Instance-based Learning

- **Non-parametric**
  - Instance-based learning

- STORE all the training sample
- When a query comes in, predict/classify the query based on the aggregation of its nearest neighbors

K-Nearest Neighbor (KNN)



https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

Instance-based Learning

- **Non-parametric**
  - Instance-based learning

- STORE all the training sample
- When a query comes in, predict/classify the query based on the aggregation of its nearest neighbors

- Nearest => which measurement of distance?
- Neighbor**s** => how many?
- Aggregation => how?
- Tie => how to deal with?

Distance Measure

- Minkowski distance:

$$X = (x_1, x_2, \ldots, x_n) \text{ and } Y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$$

$$D(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

- p=1: manhattan distance (l1)
- p=2: euclidean distance (l2)

https://en.wikipedia.org/wiki/Minkowski_distance

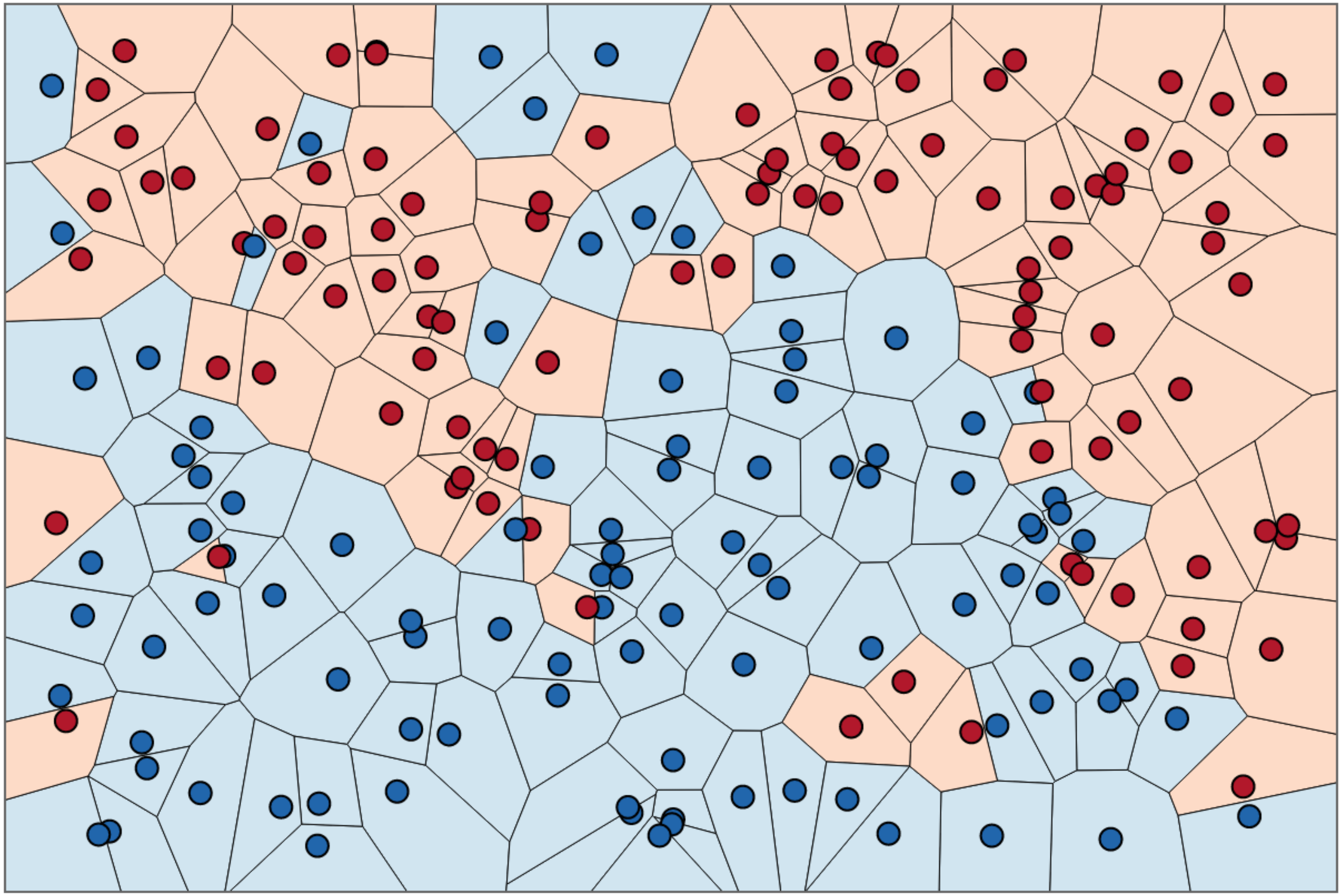
Distance Measure

- Chebyshev
- Cosine
- Jaccard
- Hamming
- …

- A lot!
- Which to pick?
  - Domain (aka application) specific
  - Data Type specific
    - Nominal? Numeric? …

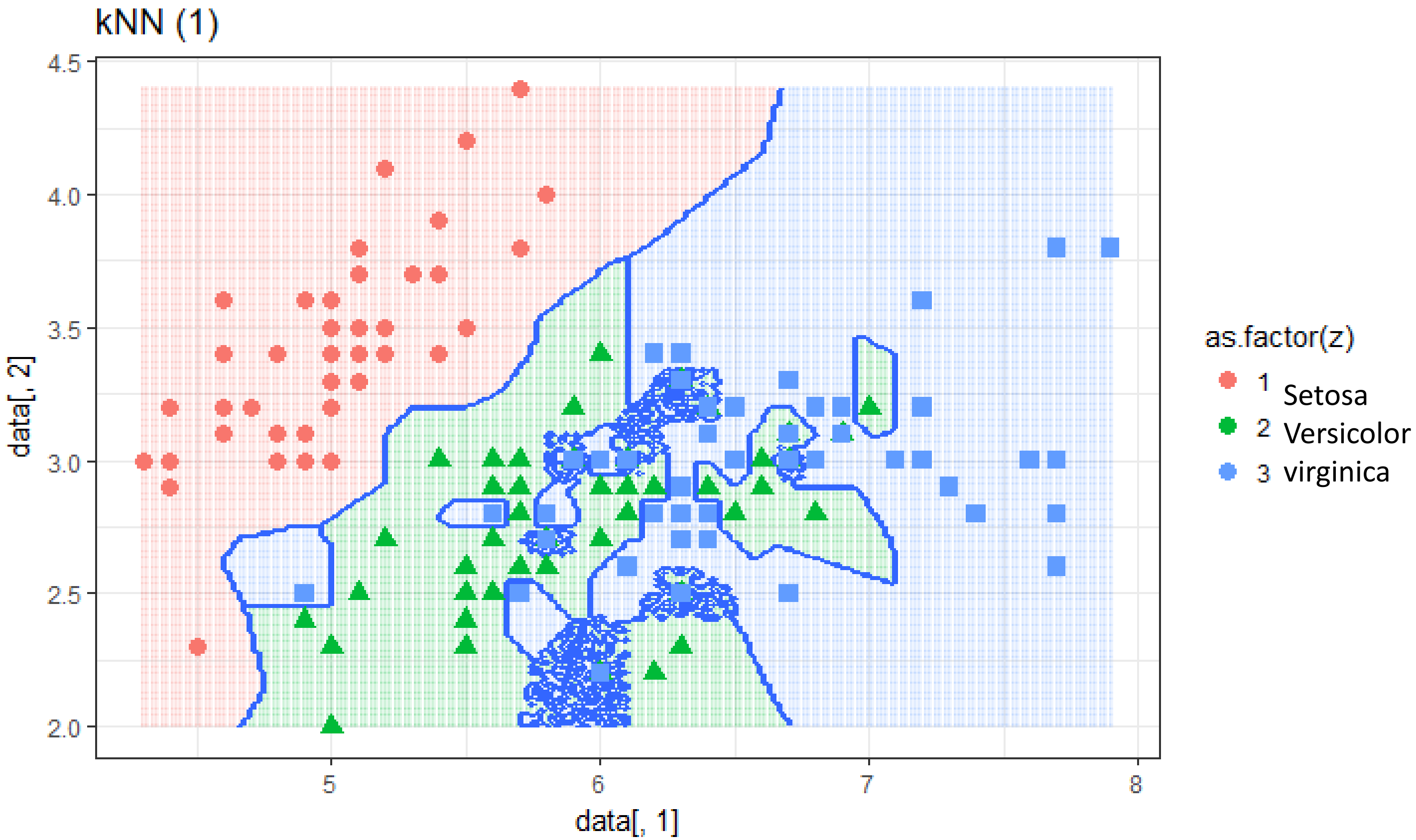
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html

Voronoi Cell Visualization 1-NN



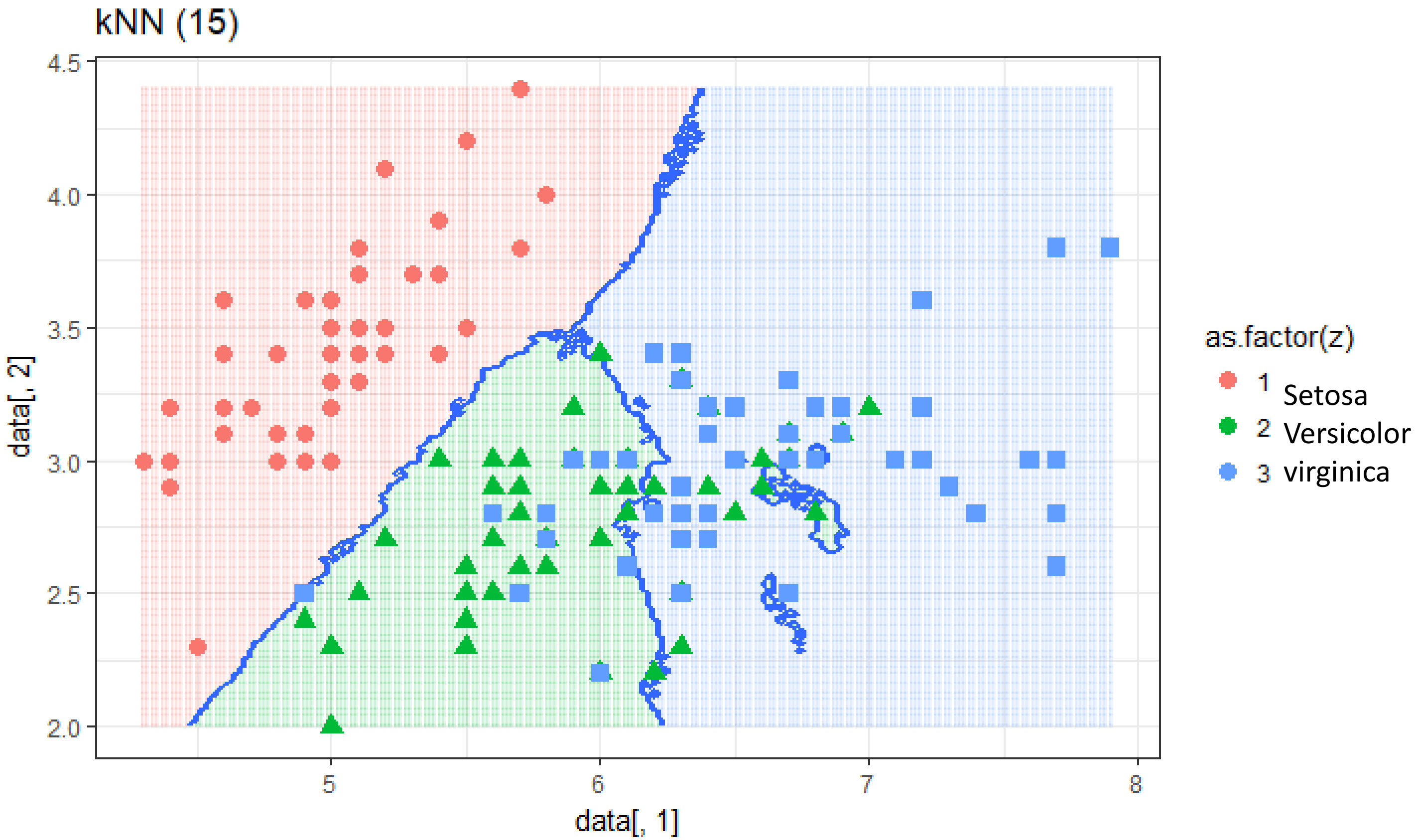
https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

KNN - 1



kNN (1)
4.5
4.0
3.5
3.0
2.5
2.0
data[, 2]
5
6
7
8
data[, 1]
as.factor(z)
1 Setosa
2 Versicolor
3 virginica

KNN - 15



kNN (15)
4.5
4.0
3.5
3.0
2.5
2.0
data[, 2]
5
6
7
8
data[, 1]
as.factor(z)
1
Setosa
2
Versicolor
3
virginica

K=???

- Low K (e.g. k =1)

- High K (e.g. k =15)

K=???

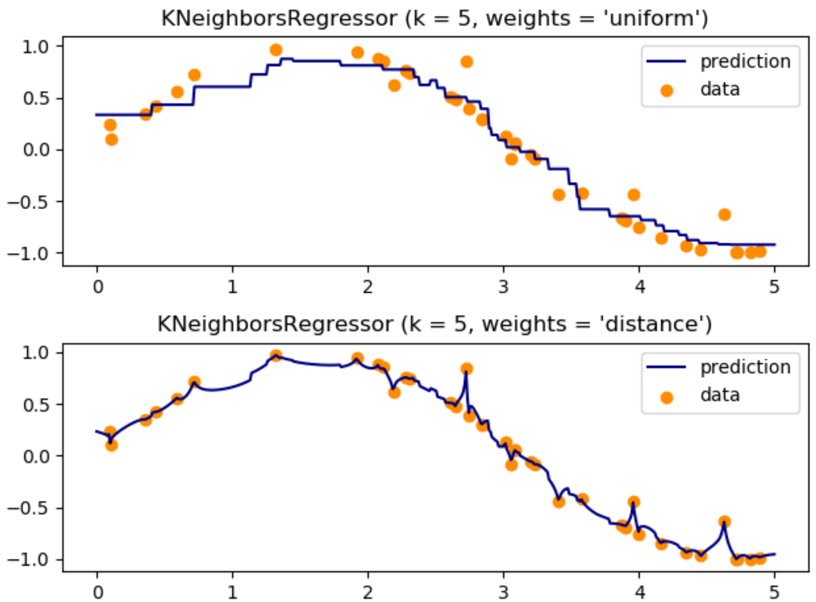
- Low K (e.g. k =1)
  - Low bias, high variance

- High K (e.g. k =15)
  - High bias, high variance

- But what is considered low/high?
  - Data -> density? Boundary?

Aggregation

- Classification – Votes
  - Tie – Reduce K until no tie is found
  - Scikit-Learn: whoever happens to come first in the original order of the dataset...
- Prediction/Regression
  - Average
  - Aka Local Interpolation



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

Aggregation

- Weights (like kernel)

$$w_{q,x} = w(d(q,y))$$

  - "uniform" => equally importance for voting/average
  - "distance" => weighted by distance
    - Smoothing function
  - "custom"

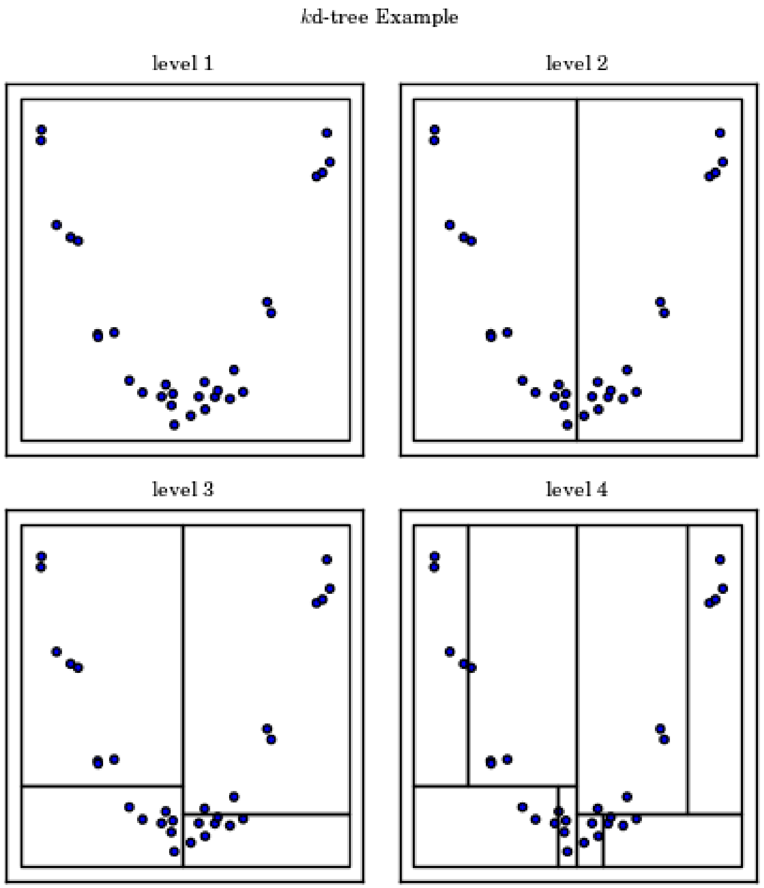
Finding nearest neighbor? (average complexity)

- Given a data set of N points/samples/records..
  - Brute-force
    - Time O(N)
    - Space O(0)

https://www.astroml.org/book_figures_1ed/chapter2/fig_kdtree_example.html

Finding nearest neighbor? (average complexity)

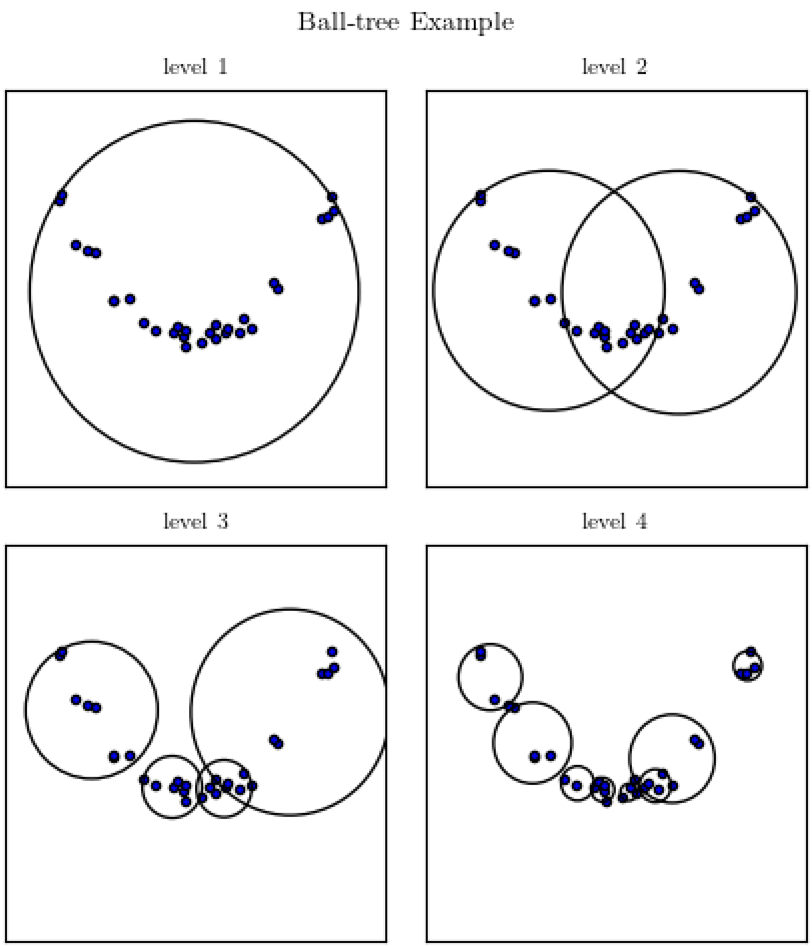
- Given a data set of N points/samples/records..
  - Brute-force
    - Time O(N)
    - Space O(0)
  - KD-tree
    - Time O(log(N))
    - Space O(N)



https://www.astroml.org/book_figures_1ed/chapter2/fig_kdtree_example.html

Finding nearest neighbor? (average complexity)

- Given a data set of N points/samples/records..
  - Brute-force
    - Search Time O(N)
    - Space O(0)
  - KD-tree
    - Search Time O(log(N))
    - Space O(N)
  - Ball-tree
    - Search Time O(log(N))
    - Space O(N)
  - AUTO
    - Determine based on data

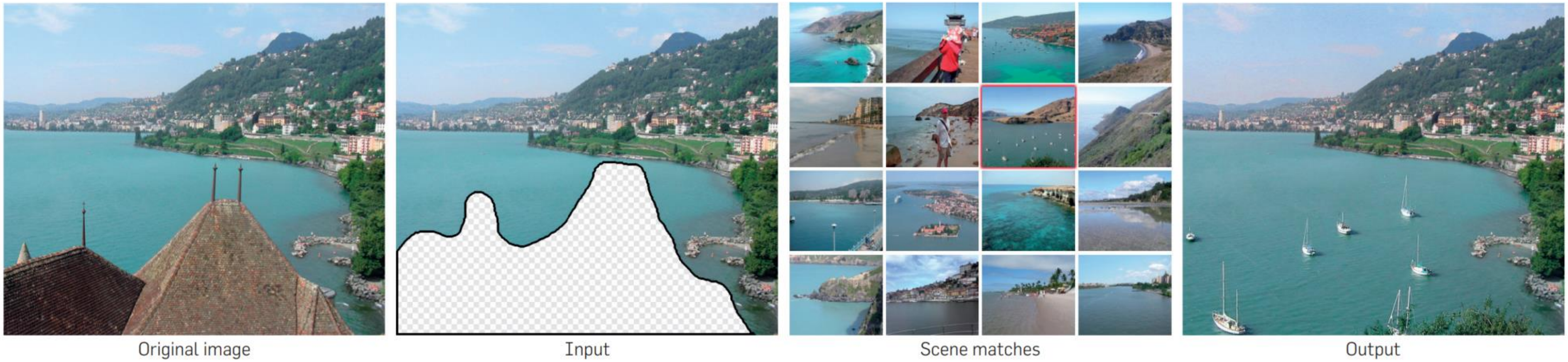


https://www.astroml.org/book_figures/chapter2/fig_balltree_example.html

Application

Scene completion

Figure 1: Given an input image with a missing region, we use matching scenes from a large collection of photographs to complete the image.



http://graphics.cs.cmu.edu/projects/scene-completion/scene_comp_cacm.pdf

Why & Why not?

- Pros:
  - Interpretability – explainable prediction
  - Fast training – with the trade-off of storage

- Corns:
  - Man this could be very slow

Lazy vs. Eager Learning

- Lazy vs. eager learning
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify

- Lazy: less time in training but more time in predicting

Bayesian Classification

$$P(B|A) = \frac{P(A|B) \times P(B)}{P(A)}$$

Given a person **X** is (age <=30, Income = medium, Student = yes Credit_rating = Fair)

Predicts **X** belongs to class C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_i|X)$ for all classes.

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

P(buy = yes | age <= 30 $\wedge$ medium $\wedge$ student $\wedge$ fair)

P(buy = no | age <= 30 $\wedge$ medium $\wedge$ student $\wedge$ fair)

age	income	student	credit_rating	buy
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayesian Classifier
An Example

age	income	student	credit_rating	_comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643

P(buys_computer = "no") = 5/14= 0.357

Compute $P(X|C_i)$ for each class

P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222

P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6

P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444

P(income = "medium" | buys_computer = "no") = 2/5 = 0.4

P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667

P(student = "yes" | buys_computer = "no") = 1/5 = 0.2

P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667

P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

X = (age <= 30 , income = medium, student = yes, credit_rating = fair)

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

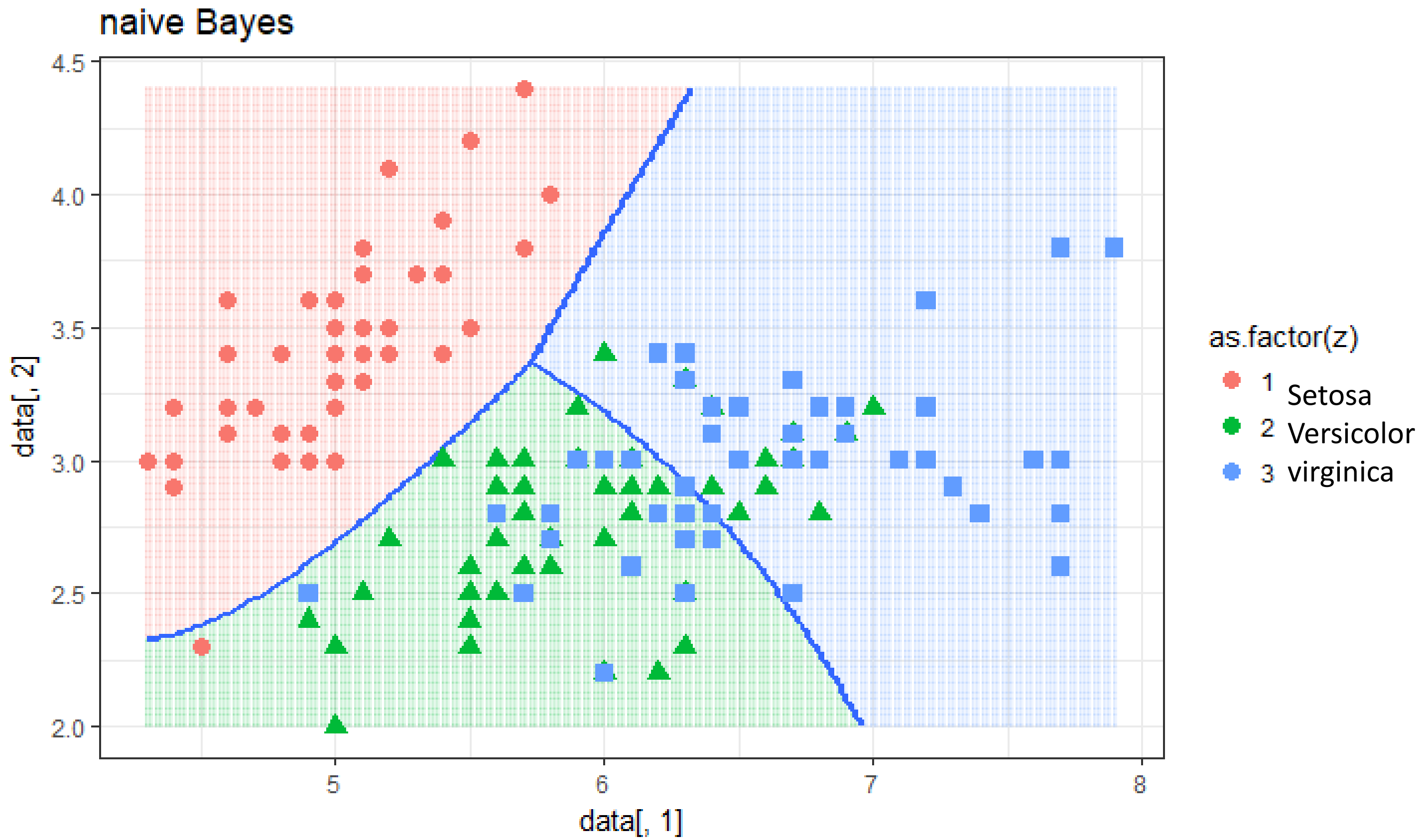
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") x P(buys_computer = "yes") = 0.028

P(X|buys_computer = "no") x P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

Naïve Bayesian – Decision Boundary



Naïve Bayesian Classifier: Comments

- Advantages
  - Easy to implement
  - Reasonably good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.
    - Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks