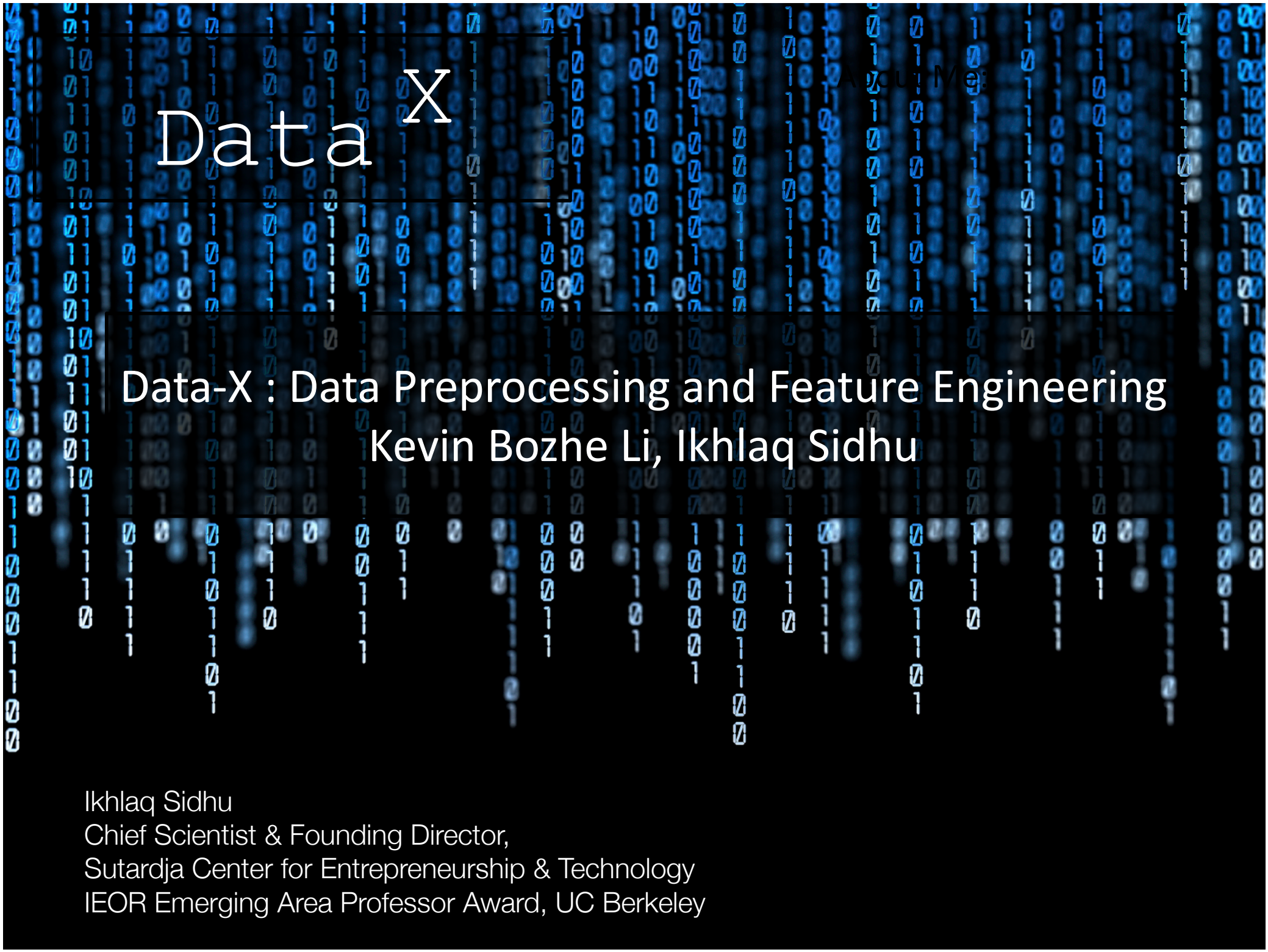


Data-X : Data Preprocessing and Feature Engineering

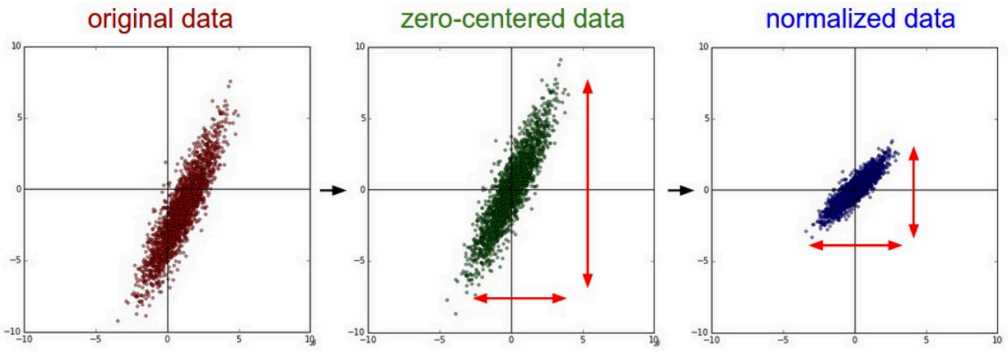
Kevin Bozhe Li, Ikhlaq Sidhu

Ikhlaq Sidhu
Chief Scientist & Founding Director,
Sutardja Center for Entrepreneurship & Technology
IEOR Emerging Area Professor Award, UC Berkeley

Data Preprocessing

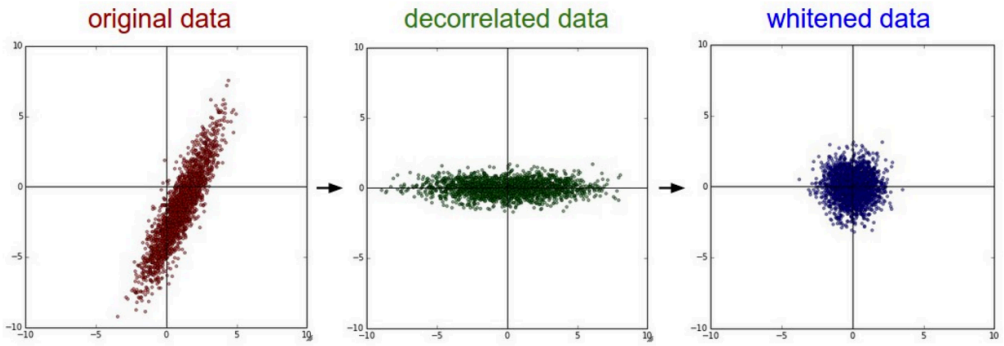
- Assume we are given a data matrix X of size [N x D], where N is the number of observations and D is the dimensionality of each observation
- We will consider the following 3 types of data preprocessing techniques:
  1. Mean Subtraction
  2. Normalization
  3. PCA and Whitening
- Without Data Pre-Processing, our results would simply be:
  - Garbage in, Garbage out

Data Preprocessing II



- Mean Subtraction
  - Subtract the mean across every feature in the data
  - Geometrically, we want to center the cloud of data around the origin along every dimension
  - In Python, the following function suffices:
  - X -= np.mean(X, axis = 0)
- Normalization
  - Geometrically, we want the features to be approximately on the same scale
  - 2 common ways
    - Divide the features by its standard deviation after mean subtraction
    - Normalizing

Data Preprocessing III

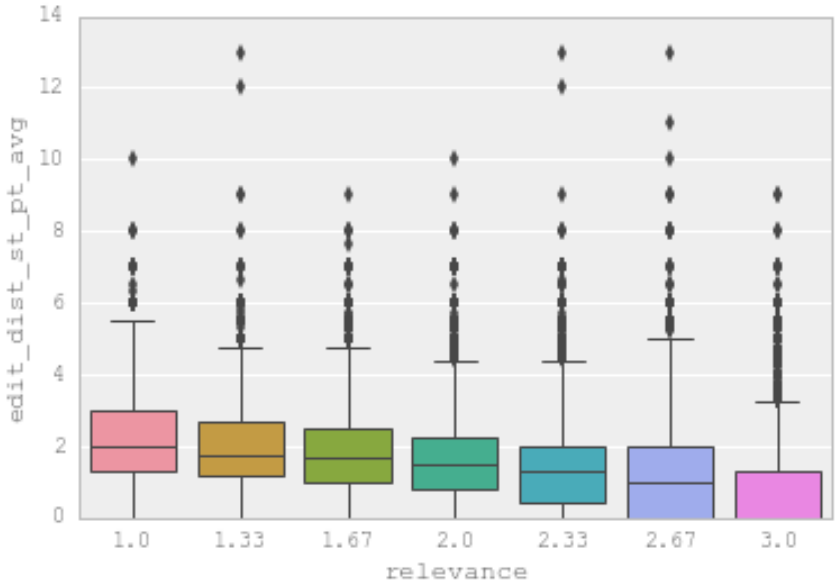


- PCA and Whitening:
  - After mean subtraction, we can perform principal component analysis which allows us to compute the covariance matrix across the feature space
  - Allows for efficient dimensionality reduction and we may select a subset of the transformed features (principal components) for training
  - The whitening transformation, on the other hand, takes the data in the eigenbasis and divide every dimension by the eigenvalue (from the PCA step)

- Final Recommendation:
  The recommended preprocessing is to simply center the data to have mean of zero and normalize its scale to [-1, 1] along each feature

Feature Engineering

- Many tasks in machine learning are highly dependent on feature engineering supplemented by model tuning and ensemble learning
- Feature engineering gets you very far
  - Use domain knowledge when you can
- When you feel that a variable is intuitively useful for the task, include it and try it as a feature:



Always check and see if it works!

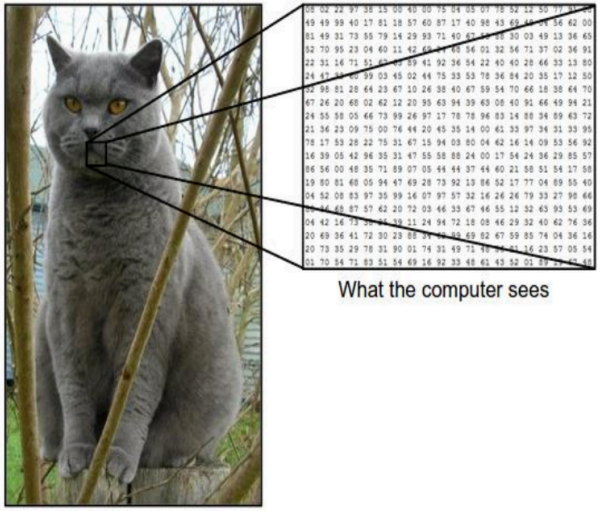
Image Classification: Feature Engineering

- Assume that we are given a set of discrete labels:
  - Cats, Dogs, Elephants, Ships
  - Question: What should we use as predictors?

Cats

What does a computer “see”?

- Images are arrays of numbers



1. Images are represented as 3D arrays of numbers, with integers between [0,255]
   i.e. 32 x 32 x 3 (a stack of 3 2-D arrays representing the 3 color channels RGB)

- Features we can use:
  - raw pixels
  - Features created from the raw images

Training on Raw Pixel

$$f(x, W) = Wx$$

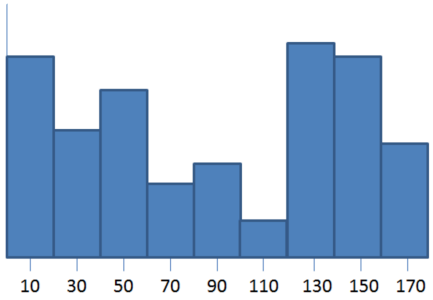
[32 x 32 x 3]
array of numbers

10 numbers
indicating class
scores

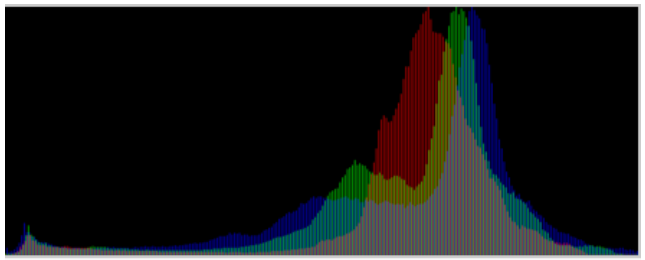
X: is a 32 x 32 x 3 image
(flattened into a vector)!!!

- *W: Parameters (Weights)*
- *x: Input*
- *f(x, W):* Score

Feature Engineering: HOG, Color Histogram



- HOG (Histograms of Oriented Gradients)
  1. A type of feature descriptor
  2. Goal: generalize the object in such a way that the same image viewed under different conditions would create the same feature descriptor
     a. Within each cell, we compute the **gradient vector** at each pixel
     b. We take a the gradient vectors of a predefined batch size and bin them in a histogram (Quantization)
- Color Histogram
  – Representation of the distribution of colors in an image



Training on derived features

$$f(x, W) = Wx$$

10 numbers indicating class scores

[32 x 32 x 3]
array of numbers

X: HOG, Color Histogram, etc (flattened into a vector)!!!

- *W: Parameters (Weights)*
- *x: Input*
- *f(x, W):* Score

References

- The material presented in this lecture references lecture material draws on the materials the following courses:
  - UC Berkeley – CS 294-129 (Designing, Visualizing, and Understanding Deep Neural Networks): https://bcourses.berkeley.edu/courses/1453965/pages/cs294-129-designing-visualizing-and-understanding-deep-neural-networks
  - Stanford – CS231n (Convolutional Neural Networks for Visual Recognition): http://cs231n.stanford.edu/

End of Section