DEEP LEARNING READING GROUP

DATA SCIENCE PRACTICE

# BEST/COMMON PRACTICES

# OUTLINE

- Preprocessing
- Weight Initialization
- Loss Functions
- Normalization
- Resources:
  - http://cs231n.github.io/neural-networks-2/
  - http://scikit-learn.org/stable/modules/preprocessing.html
  - http://ufldl.stanford.edu/tutorial/unsupervised/PCAWhitening/
  - http://ufldl.stanford.edu/wiki/index.php/Data_Preprocessing

# DATA NORMALIZATION

- Many methods work best after the data has been normalized and whitened ~ $N(0,1)$
- Computation/numeric stability.
- Guarantees that all dimensions (features) are being treated in a similar way
- Exact data-processing steps may vary from one data-set to another. It is always a good idea to inspect your data

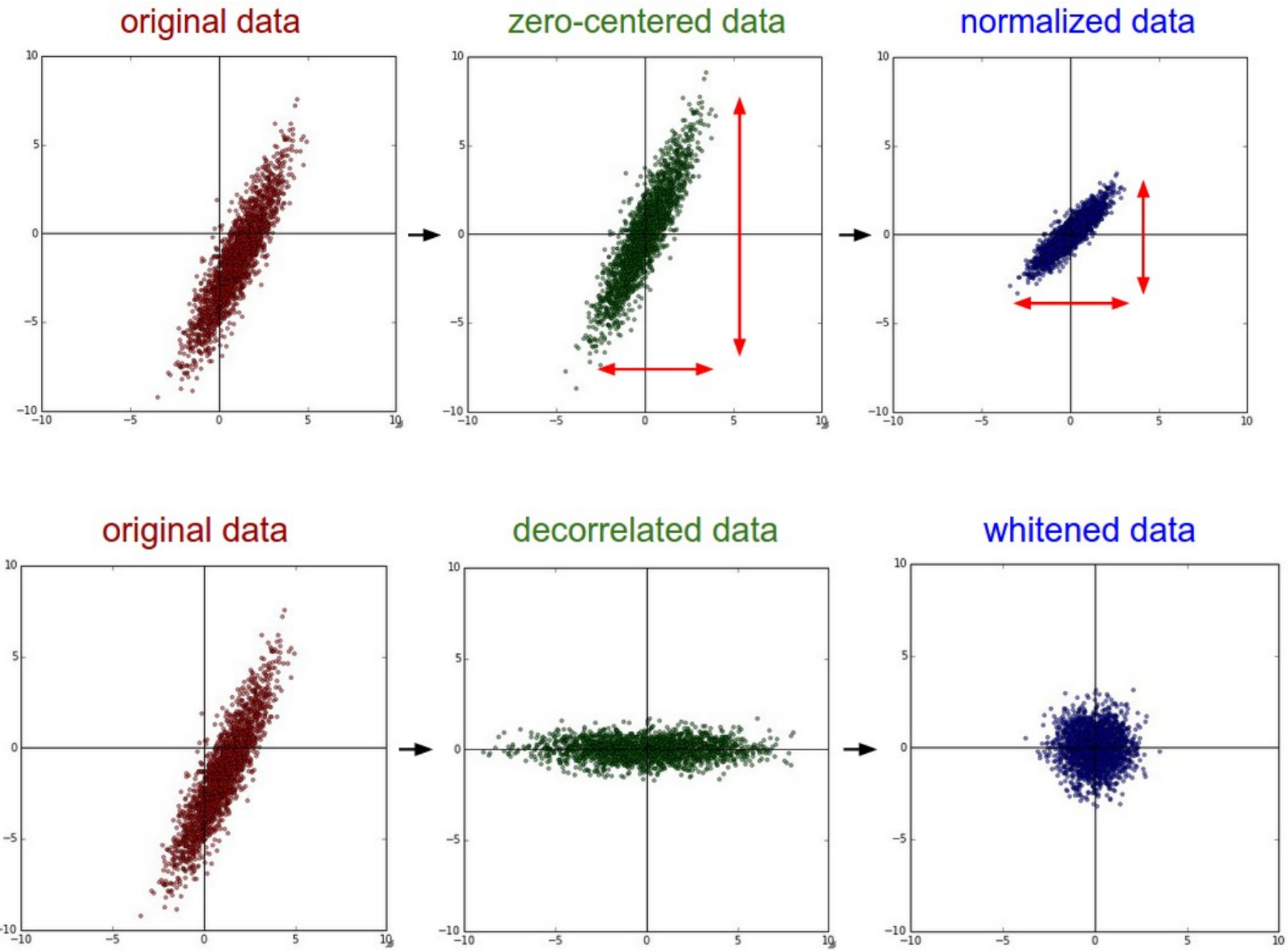
# COMMON STEPS

- **Rescaling:**
    - Rescale along each dimension (possibly independently) so that final vectors lie in the range [0,1] or [-1,1].
- **Per-example mean-subtraction, data-centering:**
    - Subtract the mean of the **training data** from each example
    - Particularly important for “stationary data” - (i.e., the statistics for each data dimension follow the same distribution)
    - Commonly done for grey-scale images, equivalent to subtracting “brightness”, but for instance this has not the same effect in color images.
    - Not sensible to do for sparse data as it destroys the sparseness in the data

## DATA PREPROCESSING

- **Feature Standardization:**
    - Set each dimension (independently) to have zero-mean and unit-variance.
    - Achieved by subtracting mean and dividing by standard deviation
    - Commonly done for audio data. Also recommended for SVM
- **Whitening:**
    - Many algorithms assume linear independence of the features
    - Use PCA to rotate the data such that the covariance matrix is transformed into the identity matrix
    - For PCA to work well
        - The features have approximately zero mean
        - The different features have similar variances to each other.
            - In images not need to scale as the scale is “global”
            - For “non-stationary” data,  rescale each feature independently

# DATA PREPROCESSING



# WEIGHT INITIALIZATION

- **~~Initialize all zero~~**
  - All neurons would have same output, same gradient and parameter updates.
  - Need “asymmetry” between neurons
- **Small random numbers:**
  - We want weights close to zero, but not too close so the gradients are not extremely small
  - In practice can use multivariate gaussian ~N(0,1) or uniform distribution
- **Calibrating the variances with 1/sqrt(n)**
  - Variance of output grows with the number of inputs, so need to scale
  - (**In practice**) For ReLU units use 2/sqrt(n) instead. [Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification]

# INITIALIZING THE BIASES

- Okay and common to use **zero**
- Some argue for small for ReLUs, but there is no consensus on whether improves or worsens performance

# BATCH NORMALIZATION - INCREASINGLY POPULAR

- Force activations throughout a network to take on a unit gaussian distribution at the beginning of the training.
- Done by normalizing each layer inputs
- Perform normalization for each training mini-batch
- Allows to use higher learning rates and care less about initialization
- Acts as a regularizer, sometimes removing the need for drop-out
- *"Applied to a state-of-the-art image classification model, Batch Normalization achieves the same accuracy with 14 times fewer training steps, and beats the original model by a significant margin. Using an ensemble of batch-normalized networks, we improve upon the best published result on ImageNet classification: reaching 4.9% top-5 validation error (and 4.8% test error), exceeding the accuracy of human raters."*
- *LINK TO PAPER*

# CROSS ENTROPY AND SOFTMAX

$$C = -\frac{1}{n}\sum_x \left(y \log(y) + (1-y)\log(1-y)\right)$$

$$a_j^L = \frac{e^{z_j^L}}{\sum_{k=1}^{K} e^{z_k^L}}$$