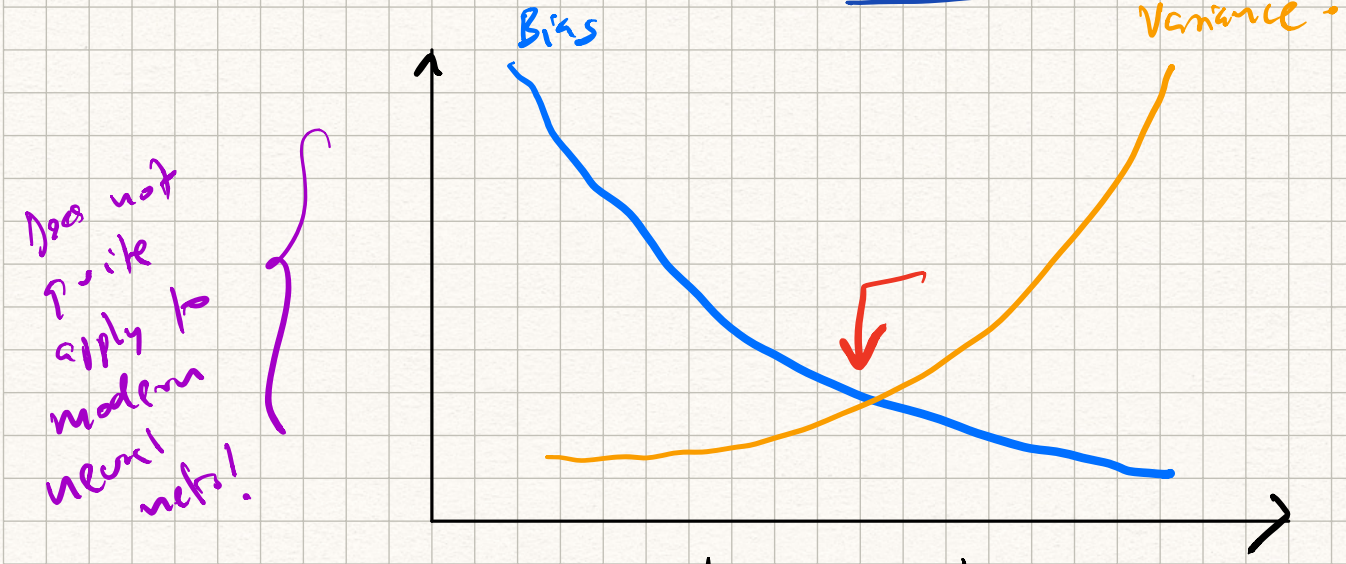
ECE-GY 6143
Intro to ML

- Recap: Neural net architectures
- Generalization in neural nets
- Intro to Unsupervised Learning
  - PCA
  - Autoencoders.

Recap:

Neural nets → Definition
→ How to train them
→ Flavors of neural nets
→ Generalization.



Model Complexity.

Classical approach to control bias-variance
— Regularization.

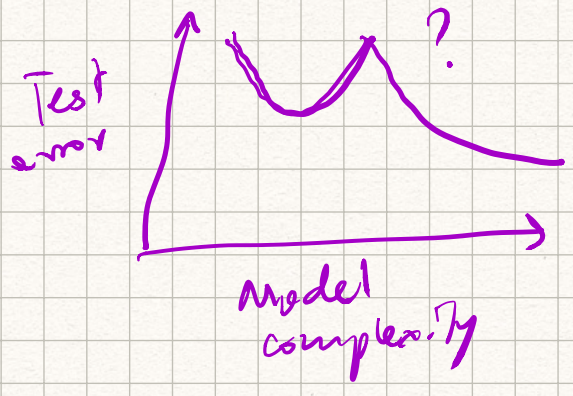
Issue with neural nets : Dataset size.

e.g. Object detection [ResNet-1000].

$\sim$ 928 million. (p)

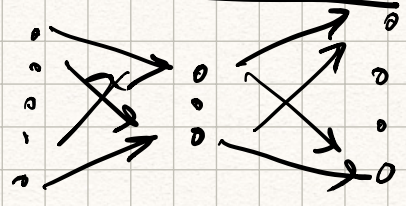
ImageNet $\sim$ 14 million (n)

$n \ll p$.



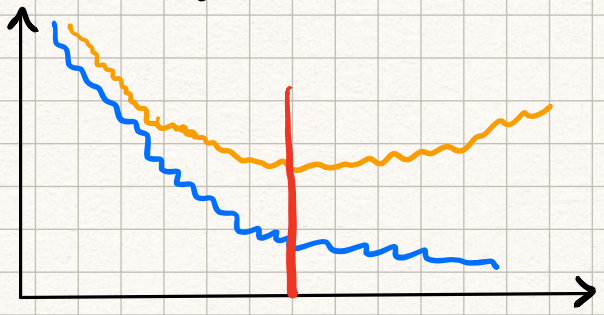
Controlling Generalization

- Add regularization
- Adding bottleneck layers.



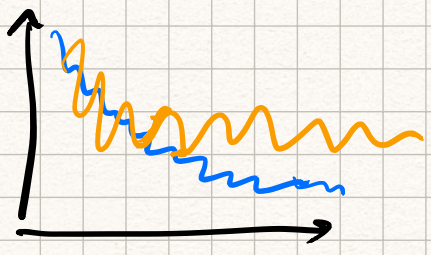
[Autoencoders/ PCA].

- Early stopping.



Iteration count

- Stop training as soon as test error increases.
- May not always work (fluctuations in test error).



- **Weight decay.** [form of regularization].
- **Dataset augmentation.** $n \uparrow << p$
  - Apply transformations to input data samples
  - e.g. Images: Shifting, Rotation, Cropping, Shearing, Flipping.
- **Dropout.** $n << p \downarrow$
  - Zero out certain neurons
  - Define a binary random variable

$$h_i = m_i \phi(z_i) \qquad m_i \sim \begin{cases} 0 & \text{with prob } p \\ 1 & \text{with } 1-p \end{cases}$$

$$\nabla_{z_i} L = \nabla_{h_i} L \cdot m_i \cdot \phi'(z_i)$$

- **Transfer learning.**

- Fine tune using an already existing network.

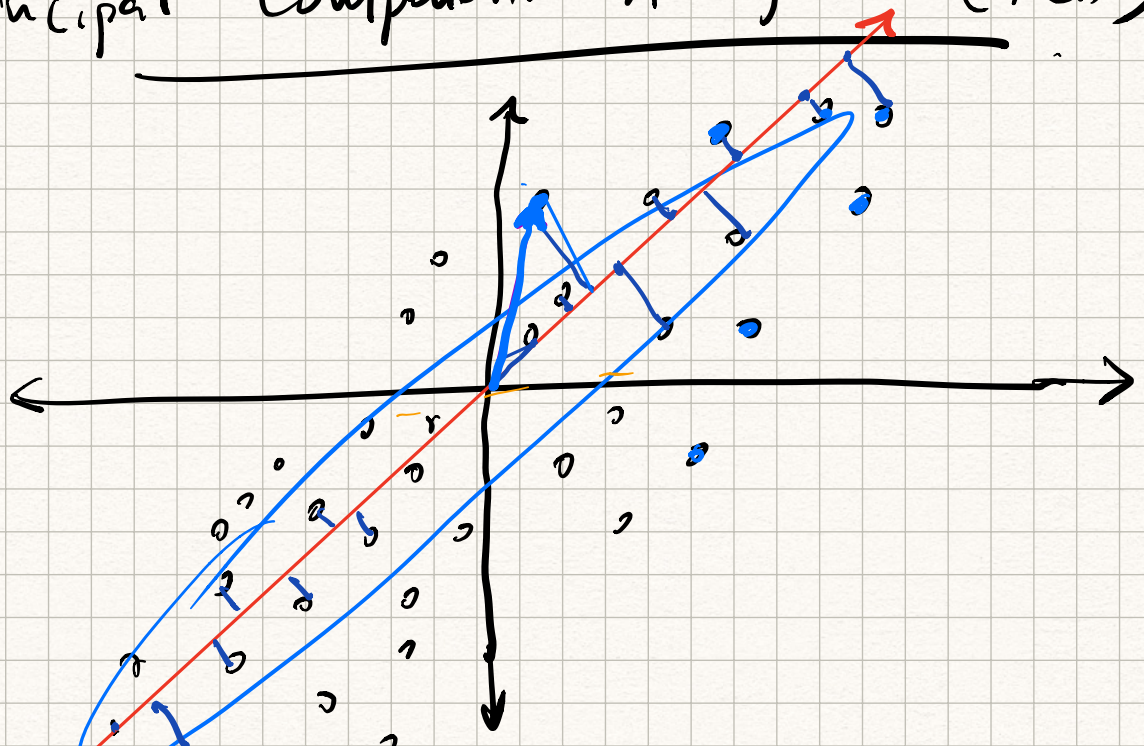
Unsupervised Learning.

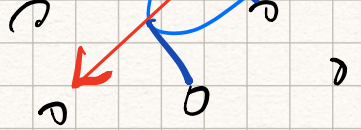
Supervised learning — Data → ML → label: $X \rightarrow \boxed{ML} \rightarrow y$

Unsupervised learning: $X \rightarrow \boxed{ML} \rightarrow ?$

- Visualization.
- Dimensionality reduction
- Clustering

Principal Component Analysis (PCA)





$$X = \begin{bmatrix} \text{---} & x_1 & \text{---} \\ \text{---} & x_2 & \text{---} \\ & \vdots & \\ \text{---} & x_n & \text{---} \end{bmatrix}_{n \times d}$$

Goal: Compute direction vector w such that w captures maximal variance.

$$\max S(w) = \frac{1}{n} \sum_{i=1}^{n} \langle x_i, w \rangle^2$$

s.t. $\|w\|_2 = 1$

$$= \frac{1}{n} \|Xw\|_2^2$$

$\|v\|_2^2 = v^T v$

$$= \frac{1}{n} (Xw)^T Xw$$

$$= \frac{1}{n} w^T X^T X w$$

$$\max_{\|w\|_2 = 1} w^T \left[\frac{X^T X}{n} \right] w$$

$$A_{d \times d}$$

$Av = \lambda v$

$v^T A v = v^T \lambda v = \lambda$

↳ Achieved when w is the top eigenvector of A [= sample covariance matrix of your data].

$$A \Rightarrow \left[(\lambda_1, v_1), (\lambda_2, v_2), \ldots, (\lambda_d, v_d) \right].$$

$v_1 \rightarrow$ "First principal component direction".

$$Xv_1 = \left(x_1^T v_1 ; x_2^T v_1 ; \ldots \; x_n^T v_1 \right)$$

$\rightarrow$ "First principal component scores"

Iteratively repeat for $v_2, v_3, \ldots$

$\{ v_1, v_2, v_3, v_k, \ldots, v_d \}$ } Basis for the data space.

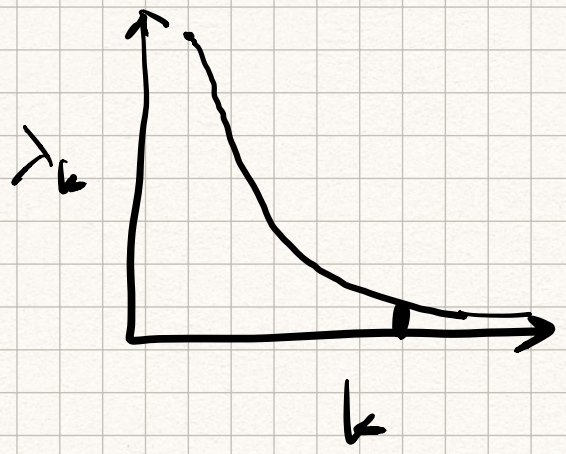
$\rightarrow$ Normalized, orthogonal.

$$x_i = \sum_{j=1}^{d} \langle x_i, v_j \rangle v_j$$

$\langle x_i, v_j \rangle$: Scores. v_j: directions.

Points to note

- Always center data.
- Make sure there are no outliers.
- Choose # principal component scores (k) wisely.



PCA vs. neural networks

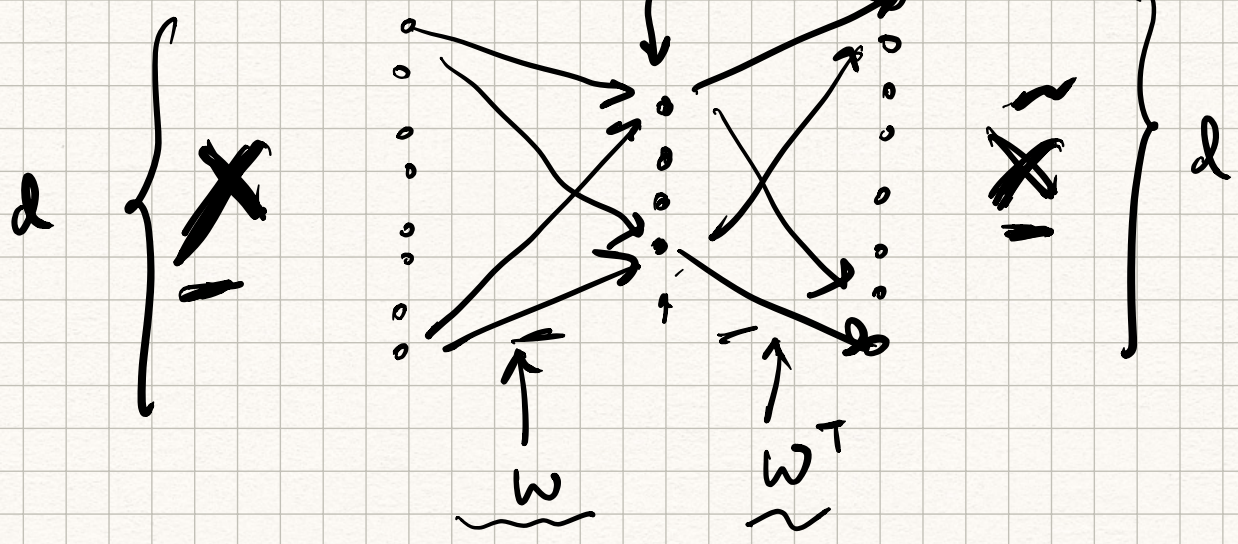
PCA : "linear autoencoder".

$$\left[X\underline{v}_1, X\underline{v}_2, \ldots, X\underline{v}_k \right]_{n \times k}$$

$$:= XW \qquad W = \left[\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_k \right]$$

$$\tilde{X} \approx X\underline{W}\underline{W}^T$$

k hidden neurons with linear activations



"Autoencoders" ≠ Encoder + Decoder

→ Denoising autoencoders

→ Sparse autoencoders

→ Stacked autoencoders