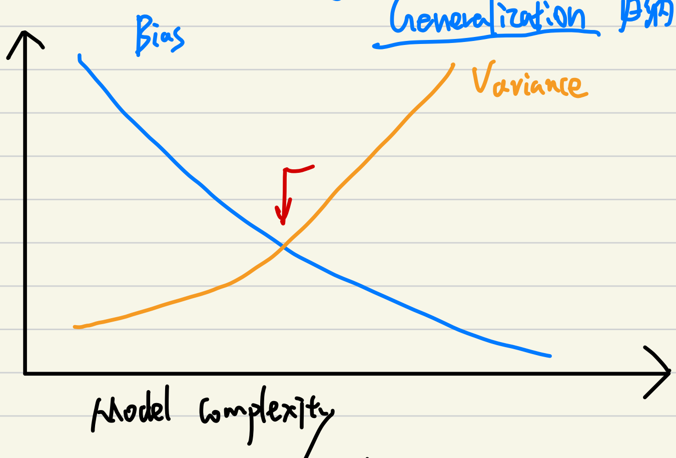
L11

* Recap : Neural Net Architecture
* Generalization in neural nets
* Intro to Unsupervised Learning
  - PCA
  - Autoencoders

Recap :

Neural Nets →
- Definition
- How to train
- Flavors of Neural Nets
- Generalization 归纳，概括

Does not suit to Neural Nets



Classic approach to control bias - variance —— Regularization

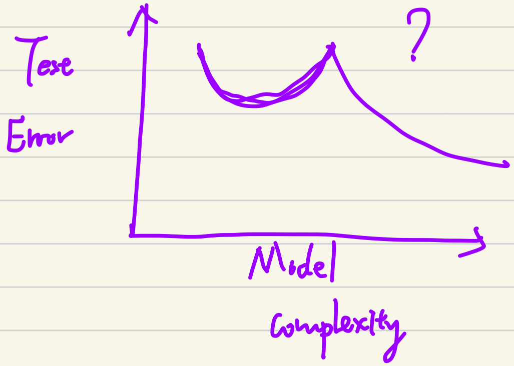
问题

Issue with neural nets : Dataset size Lots of parameters e.g. Object detection [ResNet - 1000]

~ 928 million (p) parameters

ImageNet ~ 14 million (n)

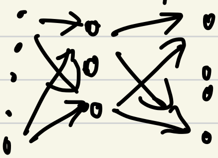
$n << p$



控制生成模型的手段：

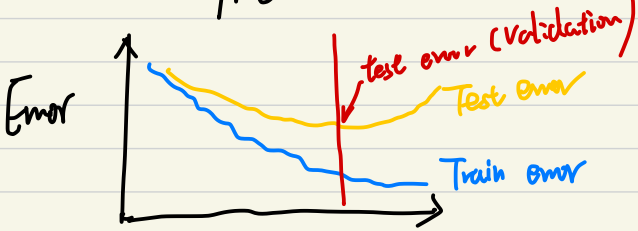
Controlling generalization

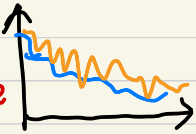
- Add regularization
- Add bottleneck layer



[Autoencoder / PCA] talk later

- Early stopping





— Stop training as soon as test error increase

— May not always work (fluctuation in test error)

- Weight decay (form of regularization)

- Dataset augmentation $\boxed{n \uparrow << p}$
  - apply transformation to input data samples eg. images shifting/rotation/flipping/cropping

- Drop out $\boxed{n << p\downarrow}$
  - zero out certain neurons
  - Define a binary random variables

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

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

- Transfer Learning
  - Fine Tune using an already existing networks

Unsupervised learning

Supervised learning

Data → label

$x \to \boxed{ML} \to y$

Unsupervised learning

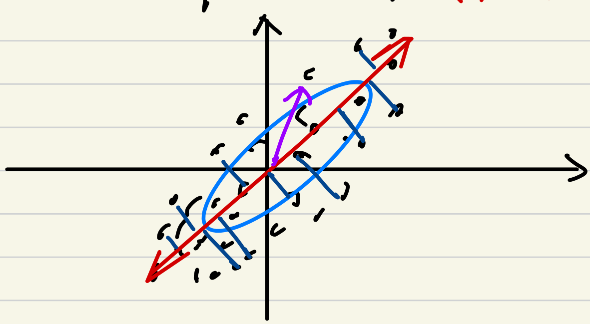
$x \to \boxed{ML} \to ?$

— visualization

— Dimensional reduction

— clustering

Priciple Components Analysis (PCA)



$$X: \quad n\times d = \begin{bmatrix} — x_1 — \\ — x_2 — \\ \vdots \\ — x_n — \end{bmatrix}$$

☆ Goal: compute direction vector w
such that w captures maximal variance

zero-mean

$$S(w) = \frac{1}{n}\sum_{i=1}^{n} \langle x_i, w \rangle^2 \quad \text{s.t.} \quad \|w\|_2 = 1$$

单位方向向量

投影分量

(normalized)

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

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

$$\max_{\|w\|_2=1} \; w^T \underbrace{\left[\frac{X^TX}{n}\right]}_{A} w$$

$Av = \lambda v$
v^TAv
$= v^T\lambda v$
$= \lambda$

→ achieved when w is the top eigenvector of A [= sample covariance matrix of your data]

$$A \to \begin{bmatrix} (\lambda_1, v_1), \ldots, & (\lambda_d, v_d) \\ (\lambda_2, v_2), \ldots & \end{bmatrix}$$

$v_1 \to$ "First principal component direction"

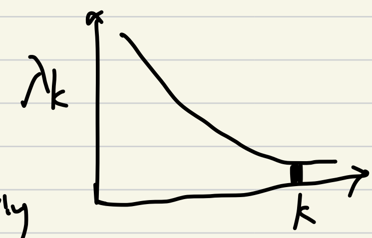
$$Xv_1 = [x_1^T v_1, \; x_2^T v_1, \; \ldots \; x_n^T v_1]$$

→ First principal component scores

Iteratively, repeat for v_2, v_3 ...

→ normalized, orthogonal

$$x_i = \sum_{j=1}^{d} \underbrace{\langle x_i, v_j \rangle}_{\text{scores}} \underbrace{v_j}_{\text{directions}}$$



☆ Points to use

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

PCA vs. neural networks

PCA : " linear auto encoder "

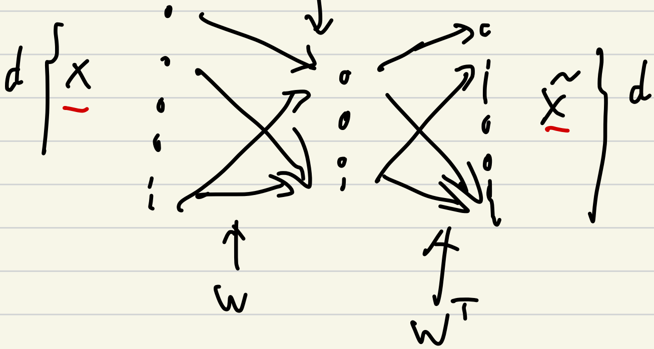
$[Xv_1, Xv_2, \cdots, Xv_k]_{n \times k}$

$\simeq XW$ dimension reduce $W = [v_1, v_2, \cdots v_k]$ k 维

$\tilde{X} \simeq XWW^T$ 还原

PCA 类比到 NN

k middle neurons with linear archiecture



"Auto encoders" = Encoder + Decoder

→ Denoising autoencoders

→ Sparse auto encoders