0.1 Learning representations of data

0.1.1 AI/MI/ML/DL

Definition:

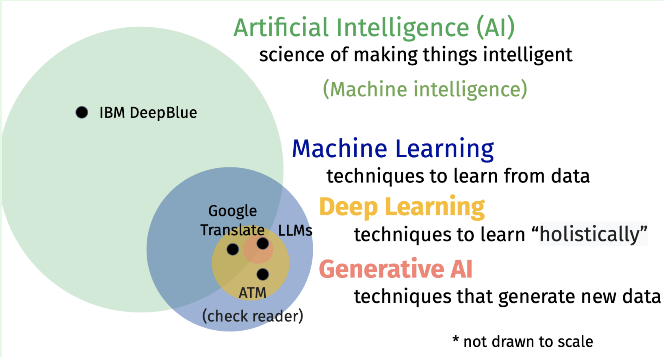

Figure 1:

0.1.2 Learning algorithms

Definition: "A **computer program M** is said to **learn** from **experience E** with respect to some class of **tasks T** and **performance measure P**, if its performance at **tasks in T**, as measured by **P**, improves with **experience E**"

- **experience E** ∼ Data
- **performance measure P** ∼ Loss function, evaluation metric
- **tasks T** ∼ "Prediction problem"
- **computer program M** ∼ Model
- **learn** ∼ Optimize

0.1.3 Linear models

Definition: Linear Regression

$$W \cdot x = y \tag{1}$$

- **E?** (x and y)
- **P?** mean squared error
- **T?** Predict y from x
- **M?** Linear model (W)
- **learn?** Analytical solution or gradient descent

Definition: Generalized Linear models in equations

$$\text{Link}(W \cdot x) = y \tag{2}$$

- x: Input features
- W: Linear transformation
- y: Output / target
- $\text{Link}(x)$: Warping function

Example:

1. If x has dim 50 and W projects to dimension 100, what is the shape of W?
   - W is a 100×50 matrix

2. If W is learnable, how many parameters does W have?
   - $100 \times 50 = 5000$ parameters

Notes: **How does a generalized linear model make a prediction?** By either mapping to a line or separating data by a line (hyperplane)

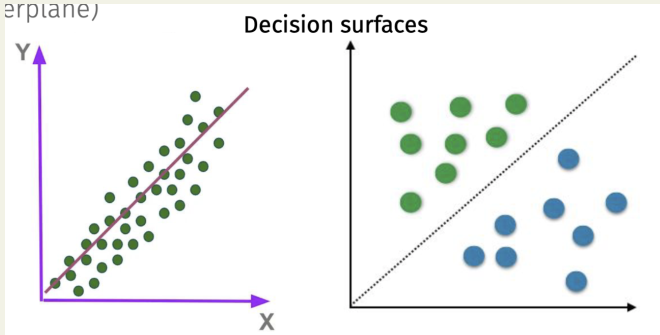

Figure 2:

Notes: **What can we do when the data cannot be separated by a line?** Resort to different decision surfaces.

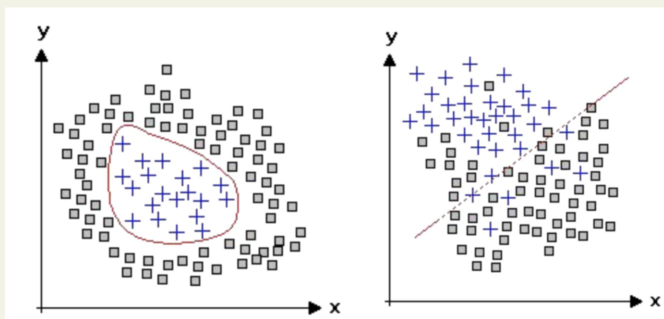

Figure 3:

0.1.4 Representations

Definition: **Representation** is a way of encoding data.

$$x \overset{\text{Representation}}{\mapsto} z \tag{3}$$

- z: Feature vectors, embeddings, latent codes, intermediate activations, etc.

Notes:

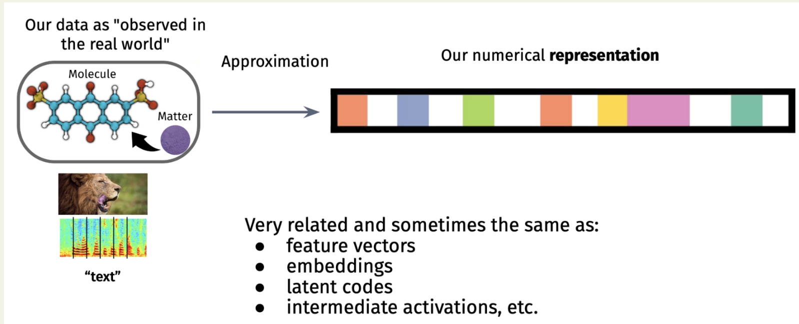

Figure 4:

0.2 Neural networks

Definition: Learnable (optimizable) transformations of data.

$$x \overset{\text{Model}}{\mapsto} y \tag{4}$$

0.2.1 2-layer MLP

Definition: By stacking linear transforms with activation functions.

$$\text{Link}(W_2 \cdot \text{relu}(W_1 \cdot x)) = y \tag{5}$$

- x: Input features.
- W_1, W_2: Linear transformations or Weight Matrices.
- $\text{relu}(x) = \max(0, x)$: Non-linear activation function, s.t. $f'(x) = 1$ if $x > 0$ and 0 otherwise.
- y: Output / target.

- **E?** (x and y)
- **P?** mean squared error
- **T?** Predict y from x
- **M?** Neural net (W1, W2)
- **learn?** gradient descent

Example:

1. What purpose does relu serve?
   - Introduces non-linearity into the model, allowing it to learn more complex functions.
2. If x has dim 50 and y dim 10, we have layer size of 50, how many parameters do we have?
   - W_1 is a 50×50 matrix, so it has $50 \times 50 = 2500$ parameters.
   - W_2 is a 10×50 matrix, so it has $10 \times 50 = 500$ parameters.
   - Total parameters: $2500 + 500 = 3000$ parameters. IS THIS CORRECT?

0.2.2 Geometric intuition

Notes: **Decision surfaces** Different ways of cutting up space to make predictions.

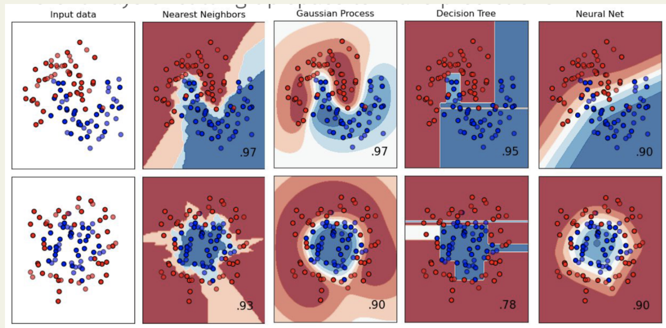

Figure 5:

Notes: **Linear Transformation** Transform data from one vector space to another

$$W \cdot x \tag{6}$$

Notes: **SVD of Linear Transformation** Factorizing matrices into geometrical transformations.

$$W = U\Sigma V^T \tag{7}$$

- U, V: Rotation
- Σ: Scaling

Notes: **Affine Transformation**

$$W \cdot x + b \tag{8}$$

- b: Bias vector

- Translate (b)
- Rotate (W-SVD)
- Reflect (W-SVD)
- Scale (W-SVD)
- Project up or down (dimensionality of $W\mathbf{x}$)

Notes: **ReLU** Rectified linear unit, which has a geometric effect of "gating", some info passes, some doesn't.

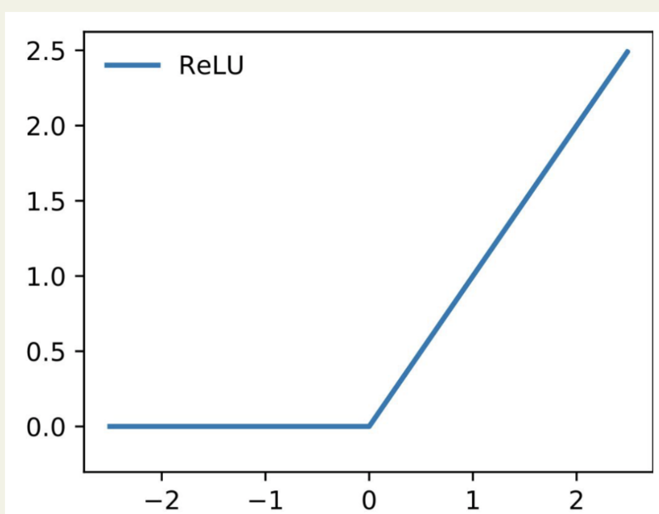

Figure 6:

Notes: Neural nets Learn to warp space to make better predictions.

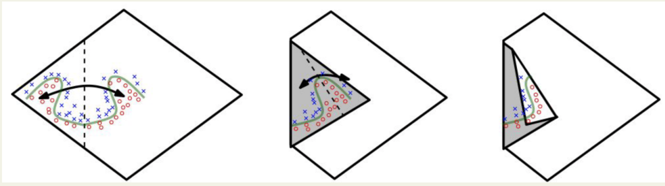

Figure 7:

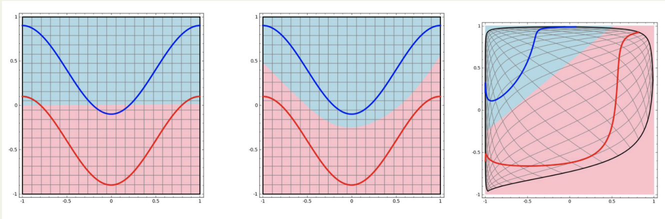

Figure 8:

0.2.3 Encoder-Decoder view

Definition:

$$x \overset{\text{Encoder}}{\mapsto} z \overset{\text{Decoder}}{\mapsto} y \tag{9}$$

- z: Embeddings, latent vectors, learned representations.

Example: Supervised Learning

$$x \overset{\text{Model}}{\mapsto} y \tag{10}$$

- $\text{Model}(x) = \text{Decoder}\big(\text{Encoder}(x)\big)$
- $\text{Decoder}(z) = \text{Pred}(z)$

Example: PCA

$$x \overset{\text{Encoder}}{\mapsto} z \overset{\text{Decoder}}{\mapsto} y \tag{11}$$

- $\text{Encoder}(x) = W \cdot x$
- $\text{Decoder}(z) = W^{-1} \cdot z$

- **E?** x
- **P?** Reconstruction loss
- **T?** Reduce dimension
- **M?** W
- **learn?** Eigendecompositions

Example: Neural Networks

$$x \overset{\text{Encoder}}{\mapsto} z \overset{\text{Decoder}}{\mapsto} y \tag{12}$$

- $\text{Encoder}(x) = \text{Neural Network}$
- $\text{Decoder}(z) = \text{Neural Network}$

0.2.4 Typical ML Pipeline

Notes:
- Setup data (x, y)
- Define a model: $y = f(x, \theta)$
- Training algorithm to find θ
- Evaluate the model.