**Task:** build a network that can recognize handwritten numbers $c \in \{0, \ldots, 9\}$ from MNIST dataset.

- 60,000 training images, labelled $n = 1, \ldots, N$; 10,000 testing images.
- 28x28 pixels, labelled by $j = 1, \ldots, 784 = \mathbf{L}$
- Each pixel grayscale value $x_n^j \in (0, 1) \equiv I \subset \mathbb{R}$
- Image is represented by vector $\mathbf{x}_n^j = (x_n^1, \ldots, x_n^{784}) \in \mathbb{R}^L$
- True target label $\mathbf{y}_n = [y_n^0, y_n^1, \ldots, y_n^9]$
- $y_n^j = \{0, 1, \ldots, 10, 0\}$ = unit vector in $\mathbb{R}^{10}$, $y_n^j$ represents number $j \in \{0, \ldots, 9\}$

**Goal:** find "decision function" $f$ that maps image to predicted label:

$$f : \mathbb{R}^L \rightarrow \mathbb{R}^{10}, \quad \mathbf{x}_n \rightarrow \tilde{y}_n \text{ in predicted label.}$$

while minimizing cost function:

$$C = \sum_{n=1}^{N} | \tilde{y}_n - y_n |^2$$

---

**1. Neural network**

- **Output layer:** $f = (f^0, f^1, \ldots, f^9) \in \mathbb{R}^{10} \rightarrow f^k = \text{probability the image is number } k$
- **Hidden layer:** $\tilde{y} = (y^1, \ldots, y^a) \in \mathbb{R}^a \rightarrow$
- **Input layer:** $\mathbf{x} = (x^1, \ldots, x^L) \in \mathbb{R}^L \rightarrow$

**Non-linear transformation:**

$$y^k = \sigma \left( b^k + \sum_{j=1}^{L} \omega^k_j \cdot x_j \right)$$

$$f^k = \frac{e^{a^k + \sum_{j=1}^{a} \omega^k_j \cdot y^j}}{\sum_{k=0}^{9} e^{a^k + \sum_{j=1}^{a} \omega^k_j \cdot y^j}}$$

**Parameters:** $\omega = (b, a, \omega, \sigma)$ = network parameters, need to minimize.

(e.g. gradient descent) $\Rightarrow$ "train network" $\Rightarrow$ "supervised learning"
2. Supervised learning with tensor networks

[Novikov 2016]

Goal: construct f using tensor network (MPS)

Train network using optimization techniques for MPS.

Answer: \( f : I^l \rightarrow I^o \), \( x \rightarrow f(x) \)

\( f^\ell(x) = \langle W^l | \phi(x) \rangle \)

\( |\phi(x)\rangle \): "feature map": grayscale input \( x \) to product state MPS with \( l \) legs.

\( \langle W^l \rangle \): "weight vector": \( W^l \) is \( l \)-leg MPS, \( \langle W^l \rangle : \text{MPS} \rightarrow f^\ell(x) \)

\( f \): "decision function": predicted label \( \hat{y} \) that \( i \) for which \( f \cdot \hat{y} \) is maximal.
Encoding input data: learn vector ordering of pixels

\[ \text{\overline{E}(x)} = (\phi(x_1) \otimes (\phi(x_2)) \ldots \otimes (\phi(x_n)) \]

\[ (\phi(x)) = \begin{pmatrix} \cos \frac{\pi x}{L} \\ \sin \frac{\pi x}{L} \end{pmatrix} \in \begin{pmatrix} \phi_1(x) \\ \phi_2(x) \end{pmatrix} \]

\[ \langle \phi(x) | \phi(x') \rangle = \sum_s \phi_s(x) \phi_s(x') = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{if } x \neq x', x' = \text{ invalid} \end{cases} \]

\[ \langle \text{\overline{E}(x)} | \text{\overline{E}(x')} \rangle \]

- smooth function \( f \times \text{avg} x \)
- induce "distance metric" in "feature space"

\[ \text{\overline{E}(x)} = \begin{pmatrix} \begin{array}{c} \frac{\pi}{L} \\ \frac{\pi}{L} \end{array} \end{pmatrix} \]

Weight vector:

\[ \langle W_k \rangle = L \cdot \text{leg \_ \_ \_ KPS} \]

\[ = \sum_{\sigma} A^{\sigma_1} A^{\sigma_2} \ldots M^{\sigma_k} L^{\sigma_k} \]

position of leg can move:

\[ \begin{array}{c} \text{p} \\ \text{male} \end{array} \]

decision function:

\[ f(x) = \langle W_k | \Phi(x) \rangle \]

\[ \begin{array}{c} \text{global decision} \\ \begin{array}{c} \text{local decision} \end{array} \end{array} \]

location of hand tensor can be shifted (during sweeping)
Cost function:
\[ C = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=0}^{y_n} (f_k(x) - y_n)^2 \]

Minimizing:
gradient descent:
\[ \Delta \theta_k = -\frac{\partial C}{\partial \theta_k} = \sum_{n=1}^{N} (f_k(x) - y_n)^1 \Phi_k(x) \]

New update:
\[ \theta_k = \theta_k + \mu \Delta \theta_k \]

Adjoint training input to next set:
\[ A \tilde{y} = \tilde{y} \in \mathbb{R} \]

constraint:
\[ \langle w^c | w^c \rangle = 1 \]

Comments:

- Cost: \( O(d^3 D^3 NL) \)
  - \( d \): local dim. \((d=2)\)
  - \( N \): # images
  - \( L \): # pixels

Once network has been trained, predict:
\[ f^g = \langle w^c | \Phi(x) \rangle \]
\[ \bar{f} \cdot \tilde{e}_j = \text{maximized} \]

MNIST:
- \( 28 \times 28 \rightarrow 14 \times 14 \) (by scan features).
- At most 5 sweeps before training converges.
- "Good" dimension:
  - \( D = 10 \Rightarrow 5\% \) error rate
  - \( D = 20 \Rightarrow 2\% \)
  - \( D = 120 \Rightarrow 0.9\% \)
Implicit feature selection - where does learning happen?

\[ f^*(x) = \langle w^* | \Phi \rangle = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} \text{subsetual basis} \]

\[ \text{subsetual basis} = \begin{pmatrix} A & A & A & B & B \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix} = \langle \tilde{w}^* | \tilde{\Phi} \rangle \]

\( \tilde{\Phi} \) is projection of \( \Phi \) into state space spanned by subsetual basis.

\( \Phi \) has \( D^2 \) components.

- Training an RPS model reveals relatively small set of features
  - Simultaneously learn dictionary and select features
- “Feature selection” occurs when SVD is used.

Future prospects

- PERS, PMERA
- TRS, TNG

- by other sampling schemes
- ‘unsupervised’ learning with known weights