Introduction to Machine Learning

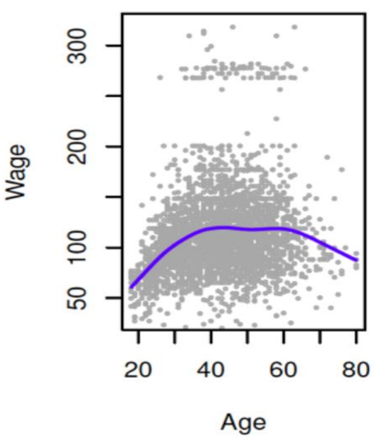
Department of Mechanical Engineering
Massachusetts Institute of Technology

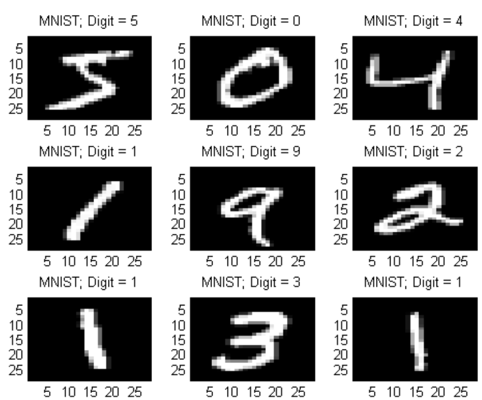
Outline

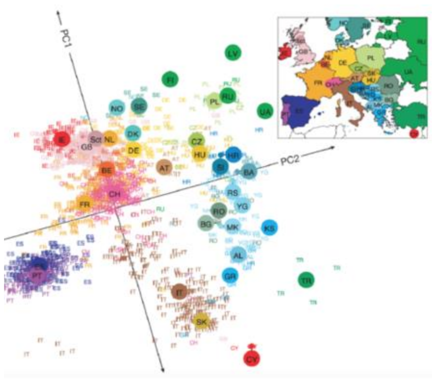
- Machine Learning Overview
  - Tasks of machine learning
  - Linear regression (regression)
  - Logistic regression (classification)
  - Support vector machine and kernels (classification)
  - Perceptron and neural network (classification or regression)
  - Decision tree (classification)
  - Unsupervised learning

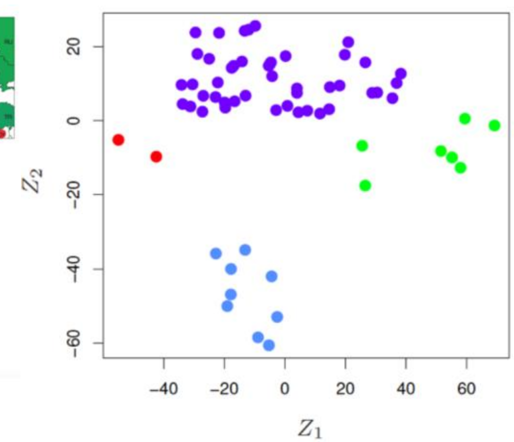
Tasks of machine learning

- Parameterized regression
- Classification
- Dimensionality reduction
- Exploit data structure (unsupervised clustering)
- Many other methods: RNN, reinforcement learning and etc.









Machine learning key ideas

- Maximum likelihood $\mathbb{P}(\text{data}|\text{parameters})$
- Maximum posterior $\mathbb{P}(\text{parameters}|\text{data})$
- Bayesian rule

$$\underbrace{\mathbb{P}(\text{parameters}|\text{data})}_{\text{posterior}} \propto \underbrace{\mathbb{P}(\text{data}|\text{parameters})}_{\text{likelihood}} \times \underbrace{\mathbb{P}(\text{parameters})}_{\text{prior}}$$

- Problem: Variance bias tradeoff (prevent over fitting)
  - Training data (find model)
  - Validation data (identify model)
  - Test data (make prediction and report result)

$$\underbrace{\mathbb{E}[(Y-\hat{Y}(X))^2]}_{\text{loss}} = \underbrace{\mathbb{E}[(Y-\mathbb{E}[Y|X])^2]}_{\text{inherent loss}} + \underbrace{\mathbb{E}[(\mathbb{E}[Y|X]-\mathbb{E}[\hat{Y}(X)])^2]}_{(\text{bias})^2} + \underbrace{\mathbb{E}[(\hat{Y}(X)-\mathbb{E}[\hat{Y}(X)])^2]}_{\text{variance}}$$

- Gradient descent $\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha_t \nabla g(\mathbf{w}^t)$

$$\alpha_t \geq 0, \quad \lim_{t\to\infty} \alpha_t = 0, \quad \sum_t \alpha_t = \infty$$

Linear Regression

- Approximate $f(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$ as

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \text{constant}$$
$$= w_{\text{TV}}x^{\text{TV}} + w_{\text{Radio}}x^{\text{Radio}} + w_{\text{NewsPaper}}x^{\text{NewsPaper}} + w_0$$

- Or, more generally

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \sum_{i=0}^{p} w_i x_i, \ \text{with } x_0 = 1$$

- Regression: find $\mathbf{w}$ that minimizes

$$\sum_{n=1}^{N} (y_n - \mathbf{w}^T\mathbf{x}_n)^2 \boxed{+ \lambda\,\mathbf{w}^T\mathbf{w}}$$

regularization

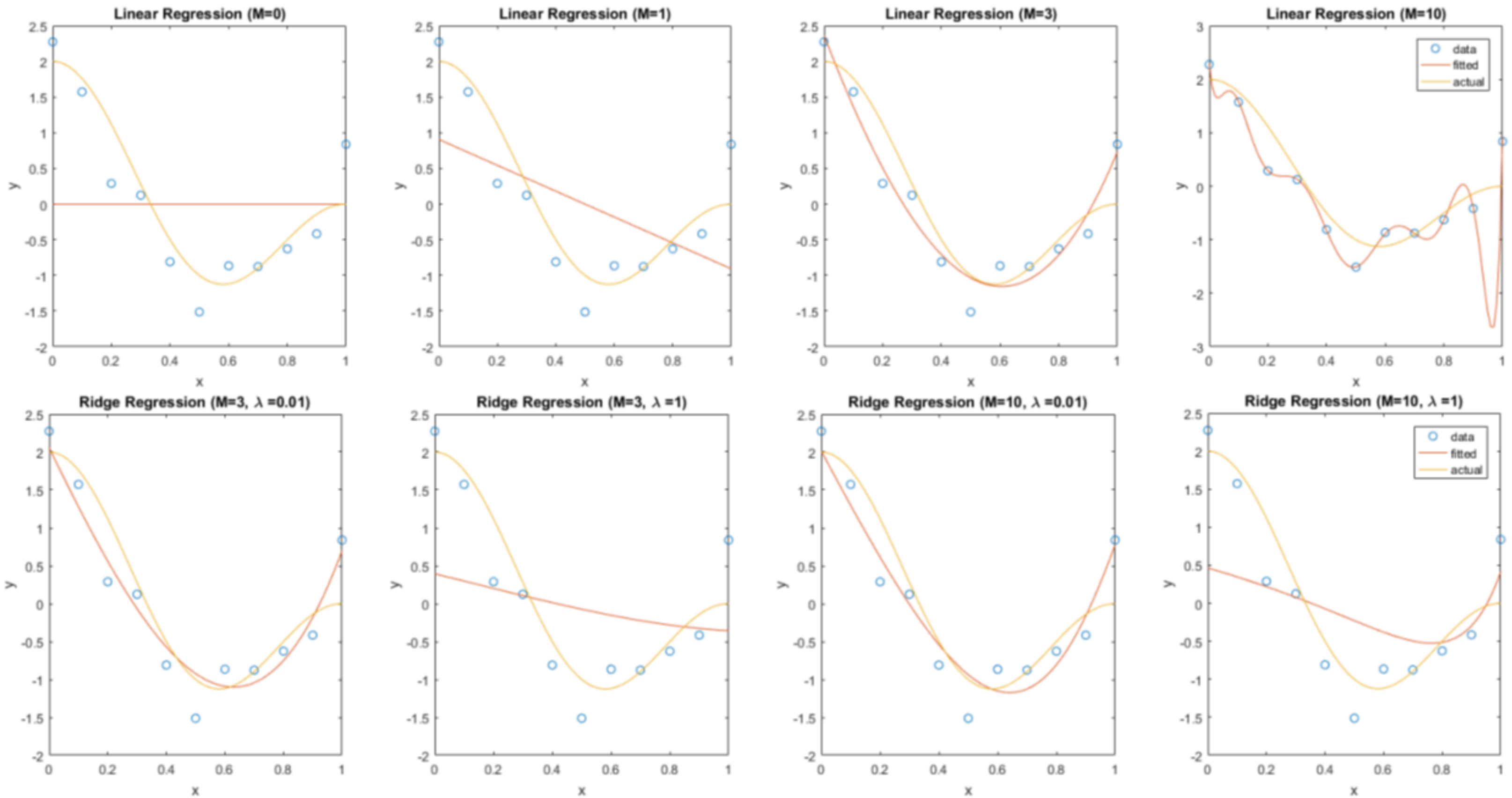
Let $Y^N = [y_n]^T$ and $X^N = [\mathbf{x}_n]^T$

Results

$$\nabla L(\mathbf{w}) = -2X^TY + 2(X^TX)\mathbf{w}$$
$$\mathbf{w}^\star = (X^TX)^{-1}X^TY$$

$$\nabla g(\mathbf{w}) = -2X^T\mathbf{Y} + 2X^TX\mathbf{w} + 2\lambda\,\mathbf{w}$$
$$\mathbf{w} = (X^TX + \lambda\mathbf{I})^{-1}X^T\mathbf{Y}$$

Effect of Complex Model and Regularization



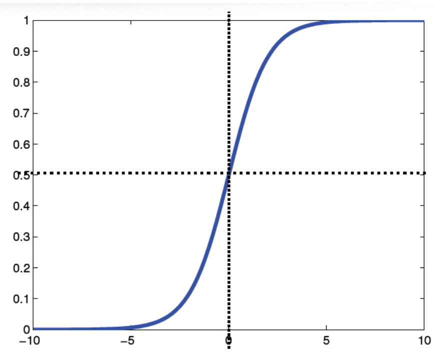
Logistic Regression

- Model probability with linear combinations

$$\log \frac{p(x)}{1-p(x)} = \beta_0 + x \cdot \beta$$

- Obtain sigmoid function for prediction

$$\sigma(z) := \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}$$



Sigmoid / logistic function

- Define cross entropy loss function

$$-\sum_{i=1}^{N}\left[y_i \log \sigma(w^T x_i) + (1-y_i)\log\left(1-\sigma(w^T x_i)\right)\right]$$

- Transform to using label 1 and -1 to get easier loss function

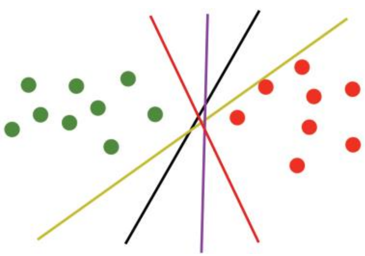
$$p(y=1|x) = \frac{1}{1-e^{-w^T x}}, \quad p(y=-1|x) = \frac{1}{1+e^{+w^T x}} \qquad \mathcal{L}(w) = \sum_{i=1}^{N} \log\left(1+\exp(-y_i w^T x_i)\right)$$

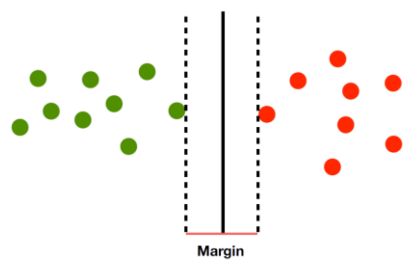
- Apply gradient descent and note that

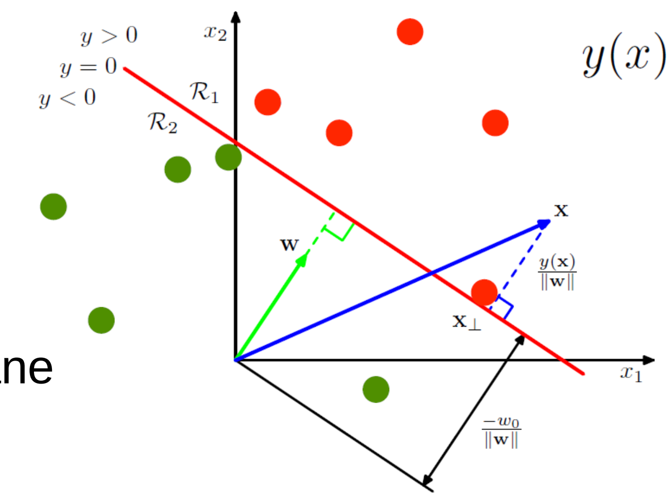
$$\sigma'(z) = \sigma(z)(1-\sigma(z))$$

Support Vector Machine

- Which line should we choose for separation?
- Idea: maximize the margin





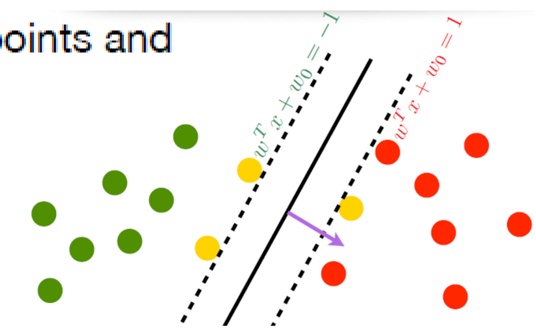


- Signed distance of a point to a hyperplane

$$\gamma = \frac{w^T x + w_0}{||w||}$$

There exists (w, w_0) such that $w^T x + w_0 > 0$ for positive points and $w^T x + w_0 < 0$ for negative points *(assume data are 'bounded')*

$$\min_{1 \leq i \leq N} \quad |w^T x_i + w_0| = 1$$



Support Vector Machine

- How to maximize the margin?

Hard problem

$$\max_{w,w_0} \frac{1}{\|w\|}$$

$$\min_{1 \le i \le N} y_i(w^T x_i + w_0) = 1$$

Easier formulation

$$\min_{w,w_0} \frac{1}{2}\|w\|^2$$

$$y_i(w^T x_i + w_0) \geqslant 1, \quad 1 \leqslant i \leqslant N$$

- Constrained optimization problem with Lagrange multiplier

$$L(w, w_0, \alpha) := \tfrac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i [y_i(w^T x_i + w_0) - 1].$$

(stationarity) $$\frac{\partial L}{\partial w} = 0, \ \frac{\partial L}{\partial w_0} = 0$$

(complementarity) $$\alpha_i [y_i(w^T x_i + w_0) - 1] = 0, \ \forall i.$$

(primal feasibility) $$y_i(w^T x_i + w_0) \ge 1, \ \forall i$$

(dual feasibility) $$\alpha_i \ge 0, \ \forall i$$

- Alternative dual form SVM

SVM dual

$$\max_{\alpha \ge 0} \left[g(\alpha) := \min_{w,w_0} L(w, w_0, \alpha) \right]$$

$$= -\tfrac{1}{2} \left\| \sum_i \alpha_i y_i x_i \right\|^2 + \sum_i \alpha_i$$

$$\sum_i y_i \alpha_i = 0.$$

SVM Inseparable Case

- Include slack variable for inseparable case

Soft margin SVM

$$\min_{w,w_0,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

$$y_i(w^T x_i + w_0) \geqslant 1 - \xi_i, \quad 1 \leqslant i \leqslant N$$

$$\xi_i \geq 0, \quad 1 \leqslant i \leqslant N$$

Dual form

$$\max_{\alpha} \quad -\frac{1}{2}\left\|\sum_i \alpha_i y_i x_i\right\|^2 + \sum_i \alpha_i$$

$$\sum_i y_i \alpha_i = 0$$

$$0 \leq \alpha \leq C.$$

- How to identify support vectors

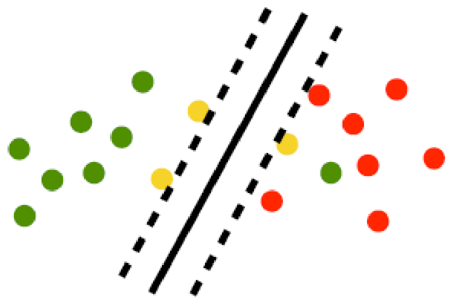
$$\alpha_i = 0 \implies y_i(w^T x_i + w_0) \geqslant 1 \qquad \text{(correctly classified)}$$

$$\alpha_i = C \implies y_i(w^T x_i + w_0) \leqslant 1 \qquad \text{(margin violation)}$$

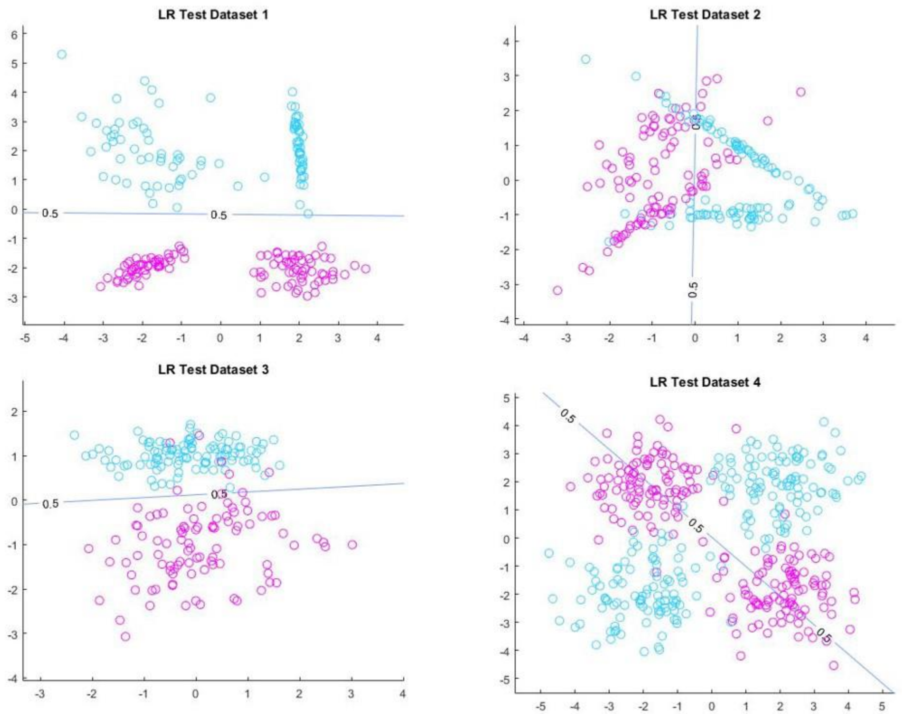
$$0 < \alpha_i < C \implies y_i(w^T x_i + w_0) = 1 \qquad \text{(support vector)}$$

- Solution takes the form

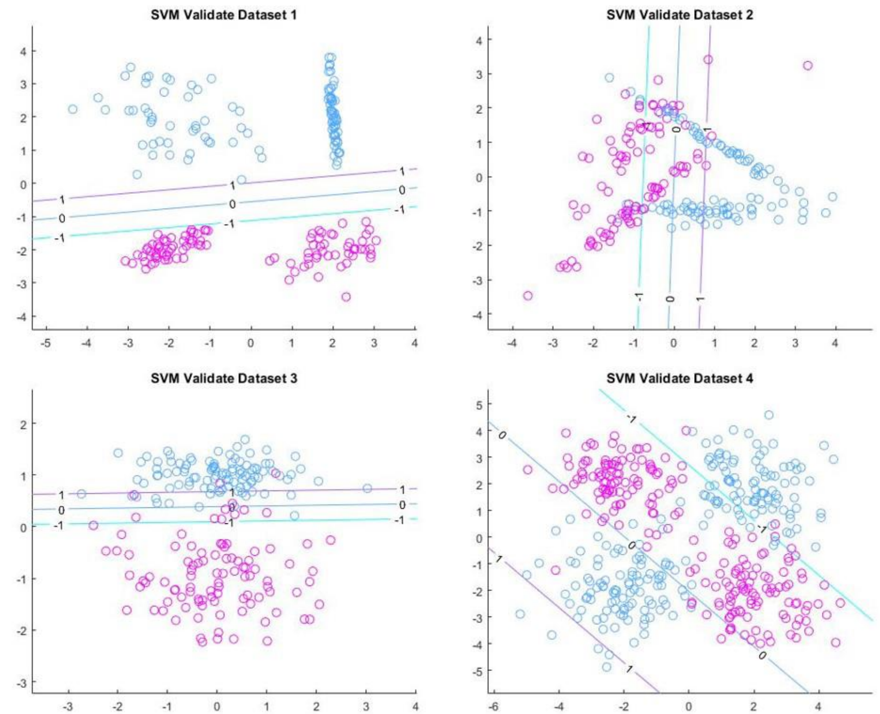
$$w = \sum_i \alpha_i y_i \phi(x_i)$$



Classification of LR and SVM



Logistic regression classification



Linear SVM classification

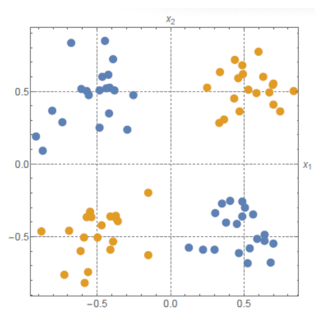
Kernel Trick

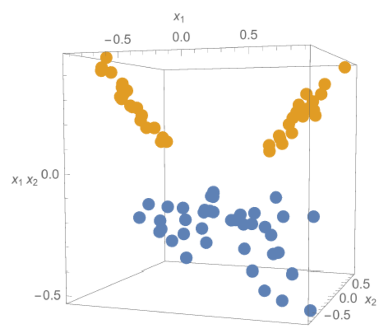
- Intuition: project data to high dimension and include non-linearity
- Key idea: Decision depends on just inner-products

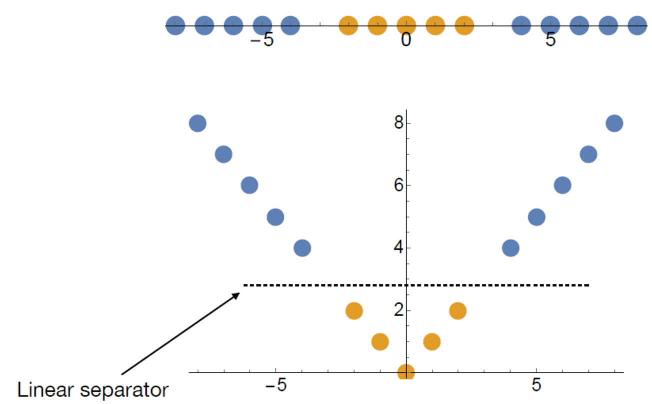
Decision function

$$h(x) = \operatorname{sgn}\left(\langle w, \phi(x)\rangle + w_0\right) \qquad \langle w, \phi(x)\rangle = \sum_i \alpha_i y_i \langle \phi(x_i), \phi(x)\rangle$$

$$w = \sum_i \alpha_i y_i \phi(x_i) \qquad k(x, x') = \langle \phi(x), \phi(x')\rangle$$



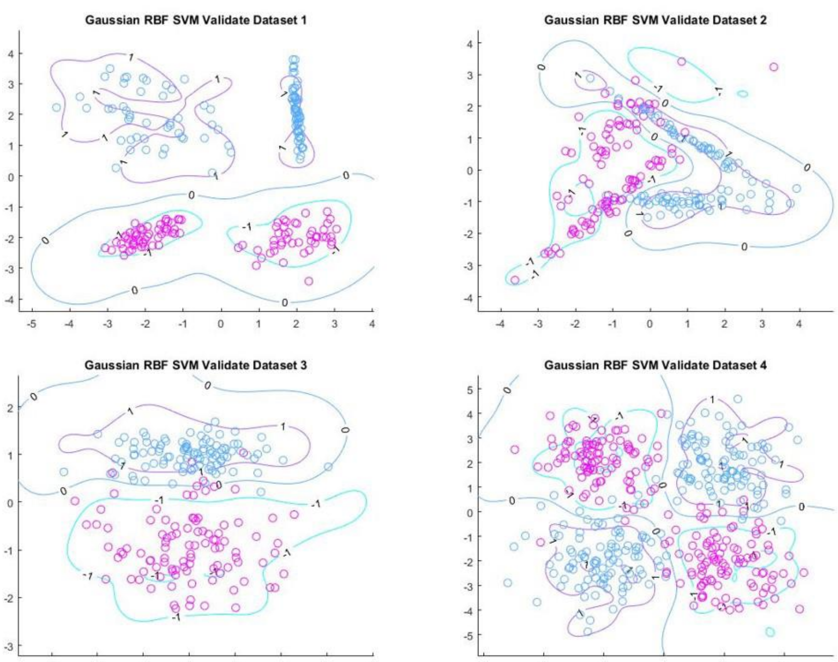




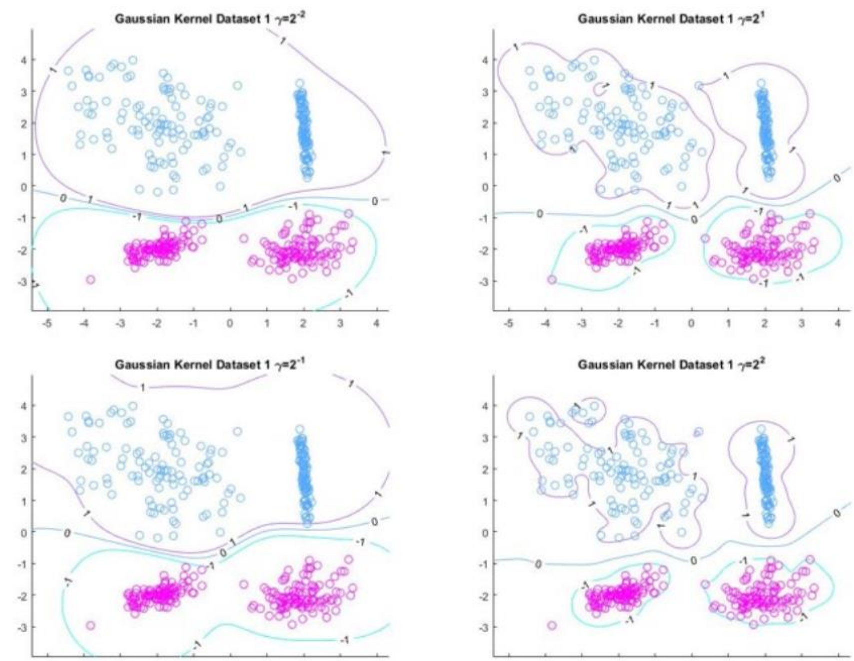
$\phi(x) : (x_1, x_2) \mapsto (x_1, x_1 x_2, x_2)$ $\qquad$ $\phi(x) : x \mapsto (x, |x|)$

Gaussian RBF Kernel

- Kernel formulation: $\exp\left(-\lambda\|x - x'\|^2\right)$



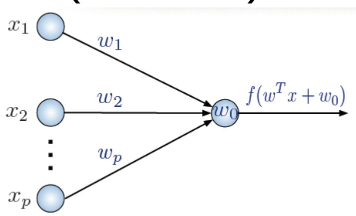
Performance on 4 datasets



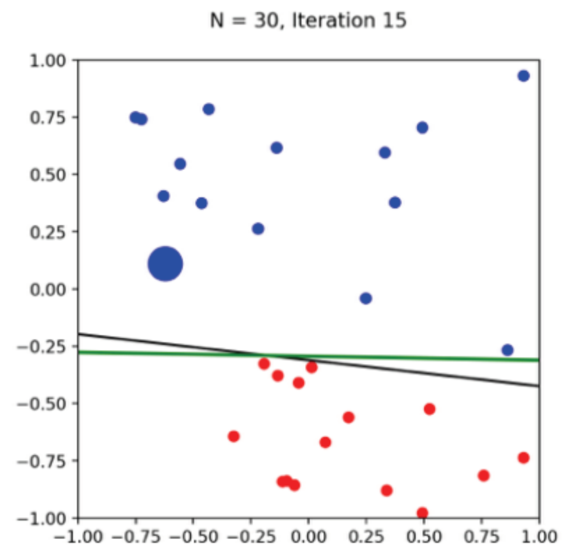
Effect of different λ

Perceptron Algorithm

- Idea: keep adjusting when miss classified
- 1 neuron neural (network)



- Converges when data is linearly separable



Algorithm:

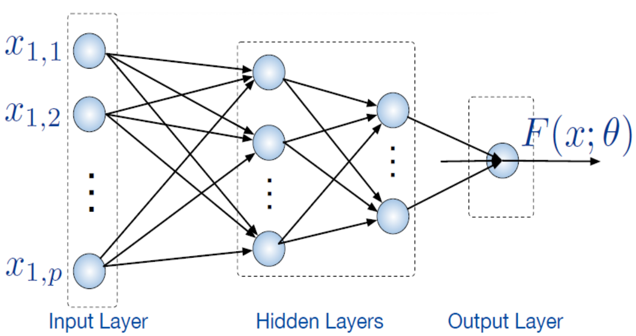
1. Initialize parameters; set iteration counter t = 1.
2. Cycle through training data $(x_1, y_1), \ldots, (x_N, y_N)$ and update

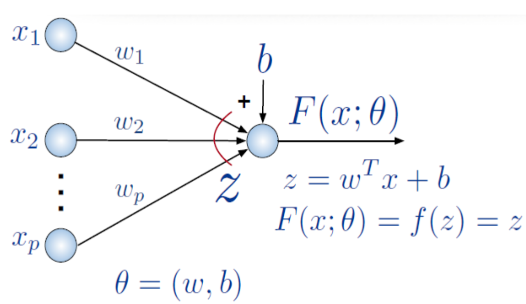
$$\left.\begin{aligned} &\text{if } y_i \neq h(x_i; w^t, w_0^t), \text{ then} \\ &w^{t+1} = w^t + y_i x_i \end{aligned}\right\} \quad y_i(x_i^T w^t + w_0^t) \leq 0$$

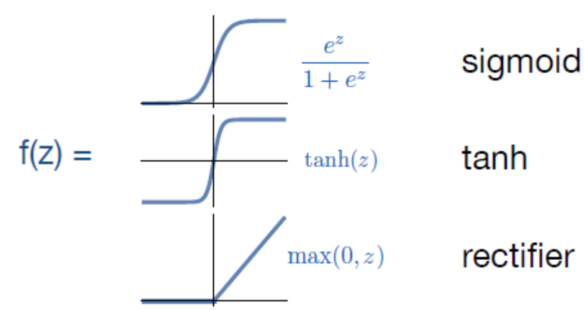
$$w_0^{t+1} = w_0^t + y_i$$

Neural Network

- Architecture: input layer, hidden layers, output layer, loss function
- Individual neuron: weighed sum, bias, activation function (non-linear)
- Very powerful tool that can accomplish complex tasks
- Can do both classification and regression
- Multiple variations available: Convolution NN, Recurrent NN, etc.







Back Propagation

- Back propagation to compute gradient with chain rule
- Can be hard with many weights to compute
- Back propagation saves derivative results in the middle
- We trade memory storage for speed

$$\frac{\partial \ell}{\partial W^l} = \frac{\partial z^l}{\partial W^l}\left[\frac{\partial \ell}{\partial z^l}\right] \qquad \delta^l = \frac{\partial \ell}{\partial z^l} = \frac{\partial z^{l+1}}{\partial z^l} \cdot \frac{\partial \ell}{\partial z^{l+1}} = \frac{\partial z^{l+1}}{\partial z^l} \cdot \delta^{l+1} \qquad \frac{\partial z^{l+1}}{\partial z^l} = \mathrm{Diag}[f'(z^l)]W^{l+1}$$

$$\delta^l = \mathrm{Diag}[f'(z^l)]W^{l+1}\mathrm{Diag}[f'(z^{l+1})]W^{l+2}\cdots W^L\delta^L$$

1. Input: (x, y) set activations for a^1 input layer
2. Feedforward:

   for each layer $l = 2, \ldots, L$ **do**

   $$z^l = (W^l)^T a^{l-1} + b^l$$

   $$a^l = f(z^l)$$
3. Output layer: $\delta^L = \mathcal{D}[f'(z^L)]\nabla_a \mathrm{loss}$
4. Backpropagation:

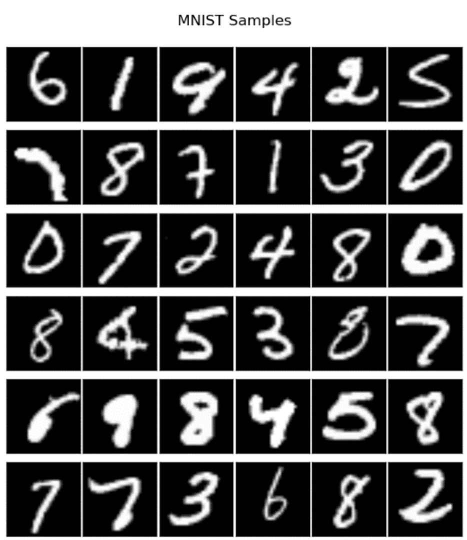
   for each $l = L-1, \ldots, 2$ **do**

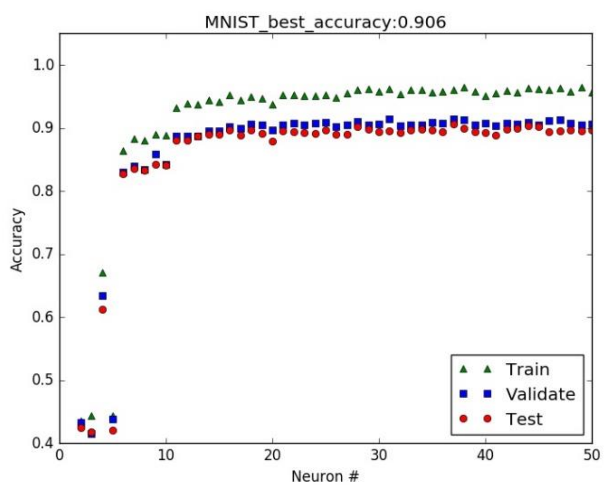
   $$\delta^l = \mathcal{D}[f'(z^l)]W^{l+1}\delta^{l+1}$$
5. Final gradients:

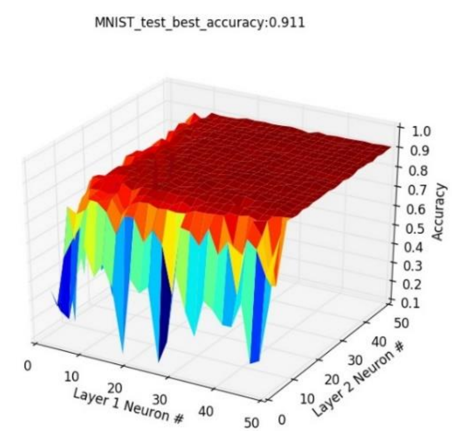
   $$\frac{\partial \mathrm{loss}}{\partial W^l} = a^{l-1}\delta^l, \quad \frac{\partial \mathrm{loss}}{\partial b^l} = \delta^l$$

NN Performance on MNIST

- Classification for MNIST at 90.6% accuracy with 1 layer
- 2 layer accuracy: 91.1%
- Exercise: use python NN implementation for MNIST dataset

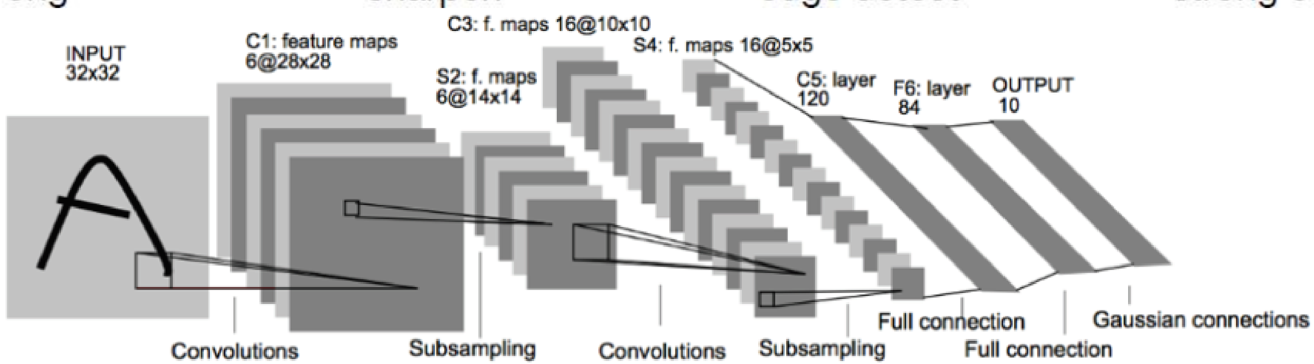






MITMECHE

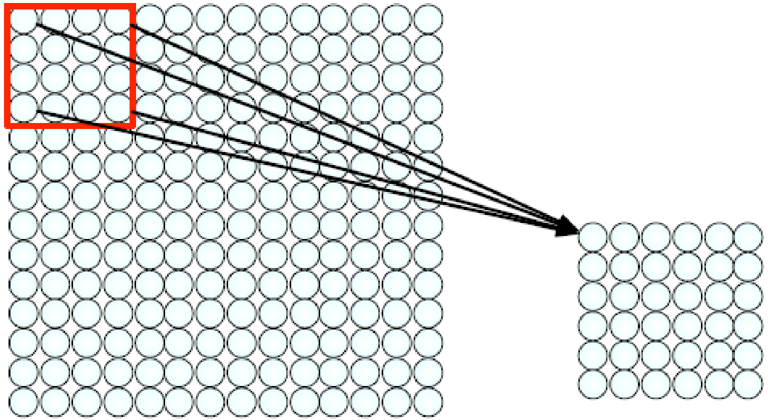
Convolutional Neural Network



CNN called LeNet by Yann LeCun (1998)

Convolutional Neural Network

- Majorly used in image classification applications
- Utilize special information from image to reduce parameters
- Convolution: apply filter moving through each pixel
- Filter can be constructed manually to achieve various effects
- Learn filter parameters using CNN instead of manually constructing

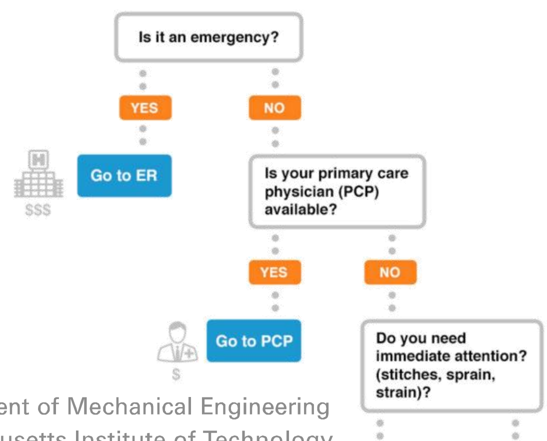


Decision Tree

- Simple heuristic in real life for making decisions
- Need formalization on how we split data (greedy algorithm)
- Define (conditional) entropy and maximize information gain

$$H(Y \mid X) = -\sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j)$$

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i) \qquad IG(X) = H(Y) - H(Y \mid X)$$



Example:

$P(X_1=t) = 4/6$

$P(X_1=f) = 2/6$

X_1: t → Y=t : 4, Y=f : 0; f → Y=t : 1, Y=f : 1

$H(Y|X_1) = -\ 4/6\ (1 \log_2 1 + 0 \log_2 0)$

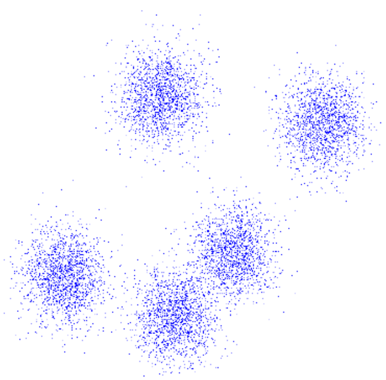
$\qquad - 2/6\ (1/2 \log_2 1/2 + 1/2 \log_2 1/2)$

$\qquad = 2/6$

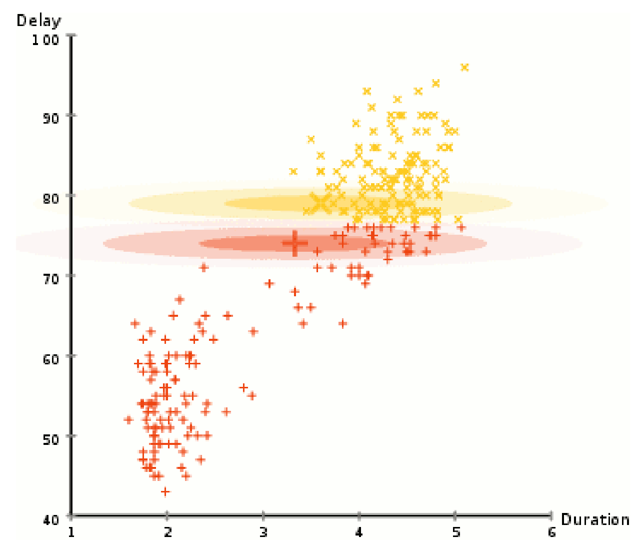
X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Unsupervised learning

- When no data labels are available
- K-means clustering
- EM Algorithm: compute expectation based on parameter to assign labels, maximize probability by adjusting parameter values



- Initialize: Pick *K* random points as cluster centers
- Alternate:
  1. Assign data points to closest cluster center
  2. Change the cluster center to the average of its assigned points
- Stop when no points' assignments change

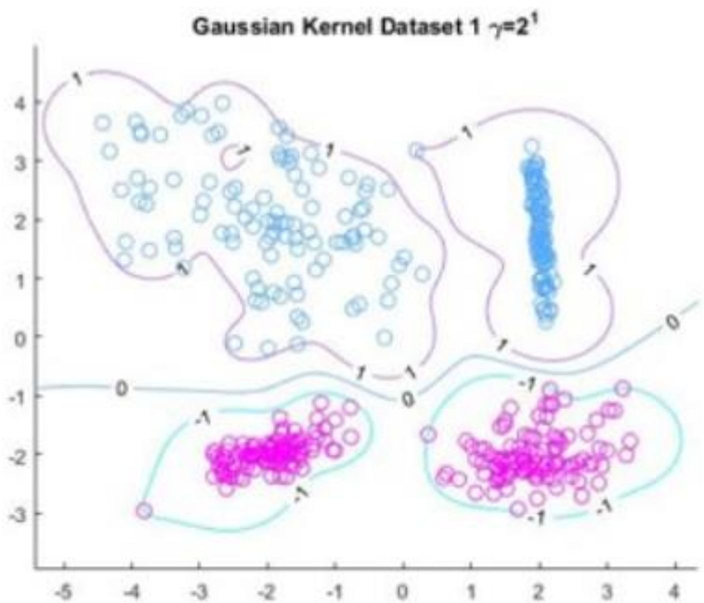


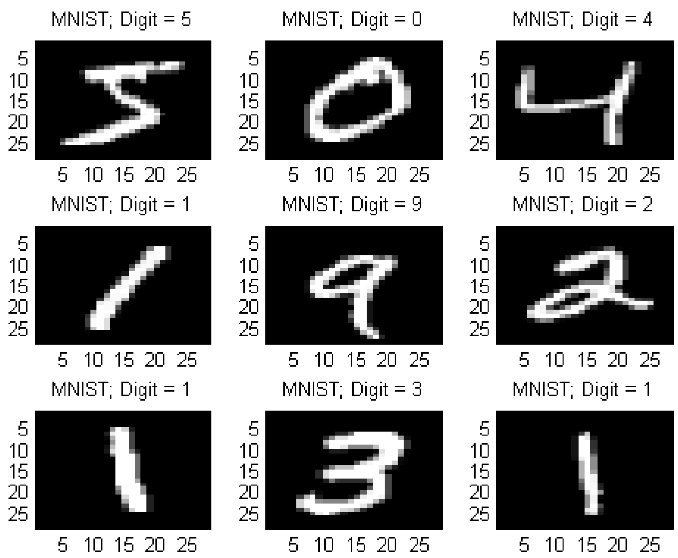
Machine Learning Summary

- Very active research field for both algorithm and hardware
- Many other techniques exist
  - Recursive neural network for language processing
  - Latent Dirichlet allocation for topic group modeling
  - Reinforcement learning for game play
  - Matrix data principle component analysis for dimensionality reduction
- Many good packages: Scikit-learn, TensorFlow, Pytorch, etc.
- Many dataset available online (e.g. UC Ivrine Repository)
- Fun to plan around and apply to various tasks
- Many resources online to learn in more details
  - Udacity, Coursera, Lynda, Packt Publishing
  - MIT OCW, Stanford,

Demo

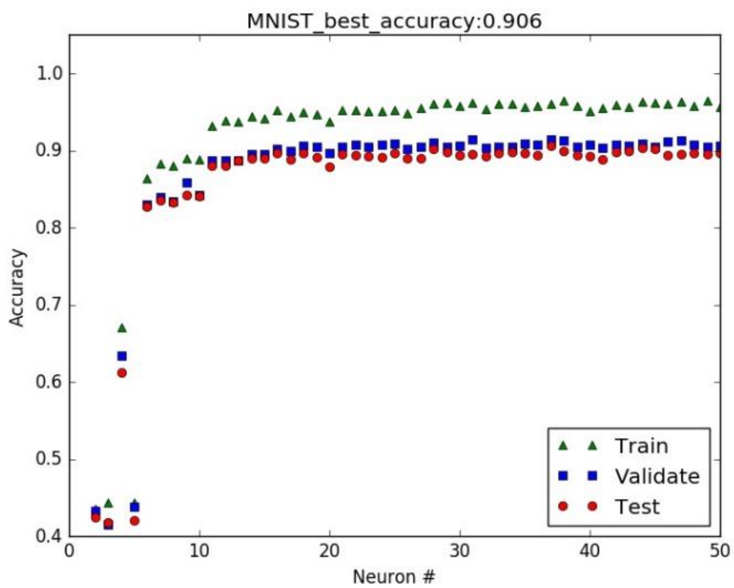
- Matlab SVM for 2 Dataset and MNIST classification
- Python Neural Network for MNIST classification

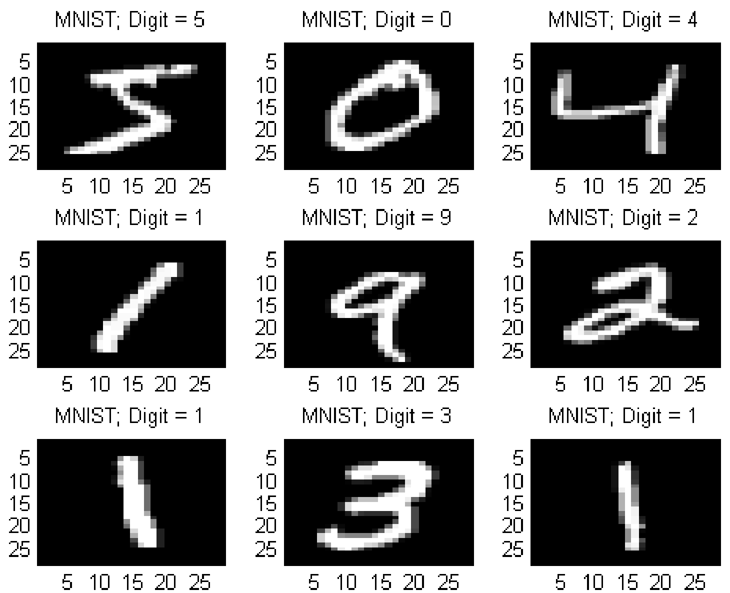




Exercise

- Implement 1 layer neural network for MNIST multiclass classification
- Report your result for 1 hidden layer NN with 0 to 50 neurons





MITMECHE

Thank You!