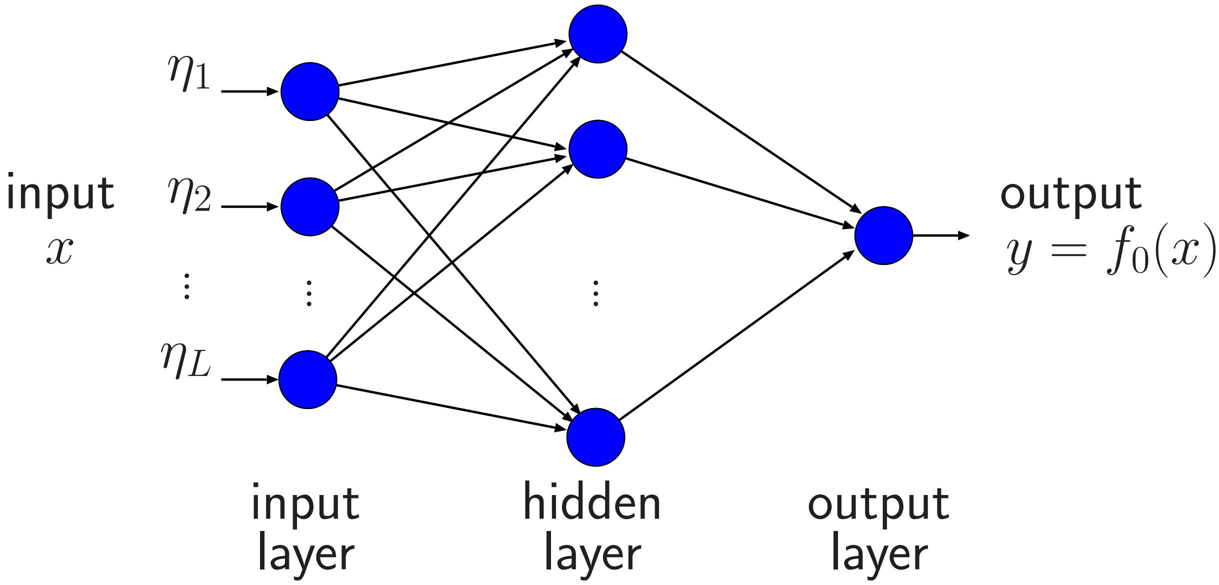
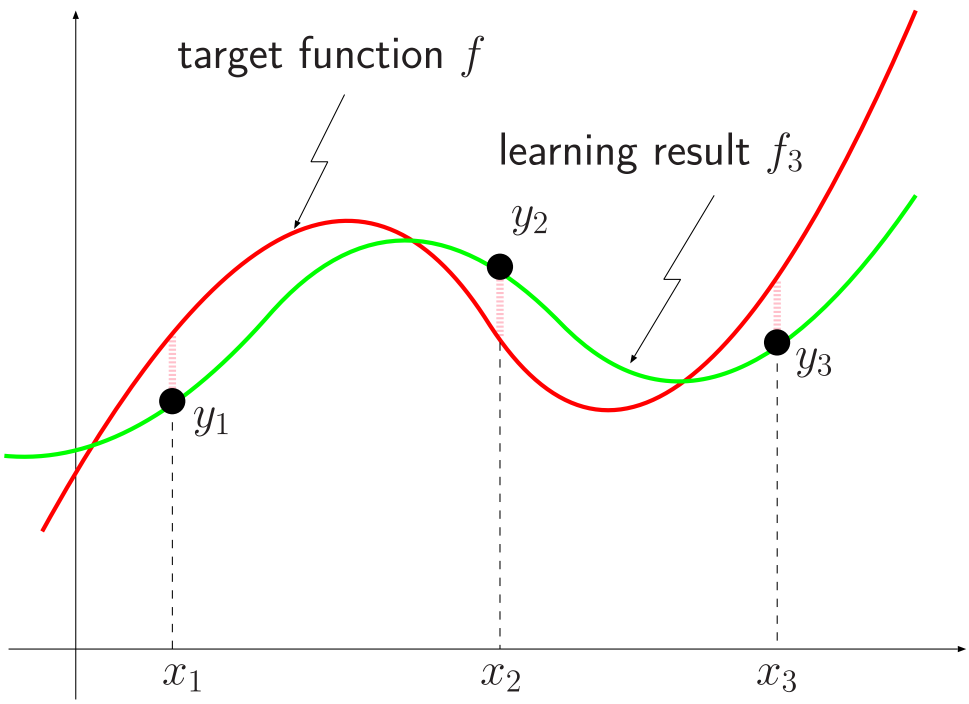
Learning in Neural Networks



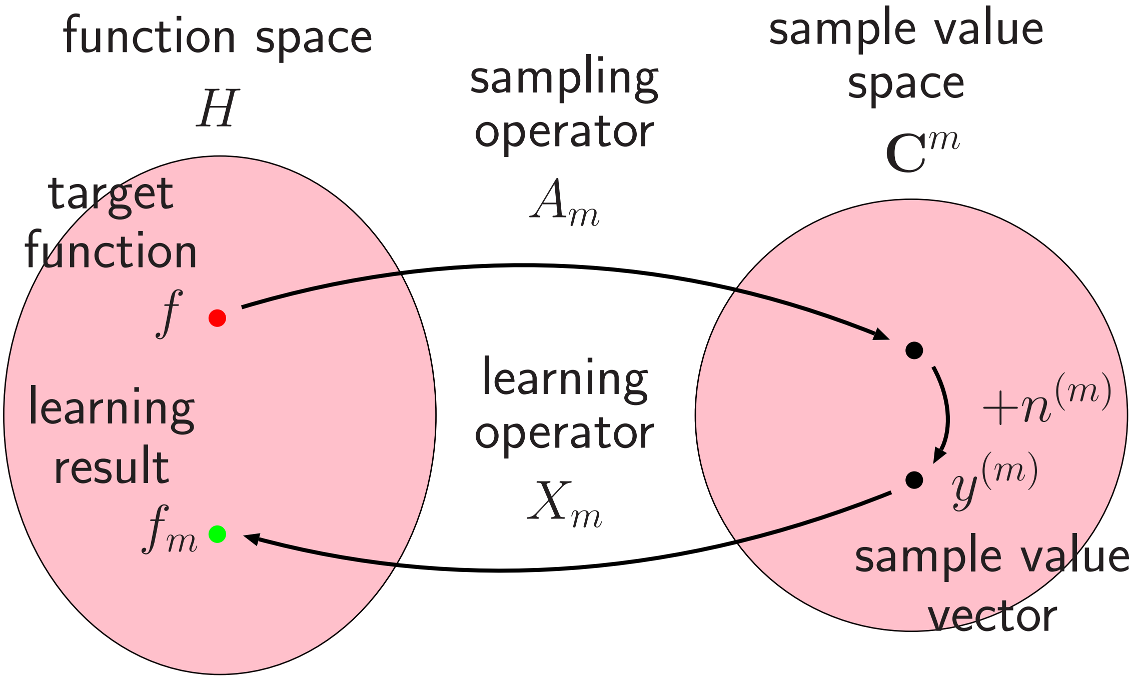


training examples : $\{(x_i,\ y_i) \mid y_i = f(x_i) + n_i\}_{i=1}^{m}$

x_i : sample points

y_i : sample values

NN learning as an inverse problem



$$\text{sampling} \;:\; y^{(m)} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = A_m f + n^{(m)}$$

$$\text{learning} \;:\; f_m = X_m y^{(m)}$$

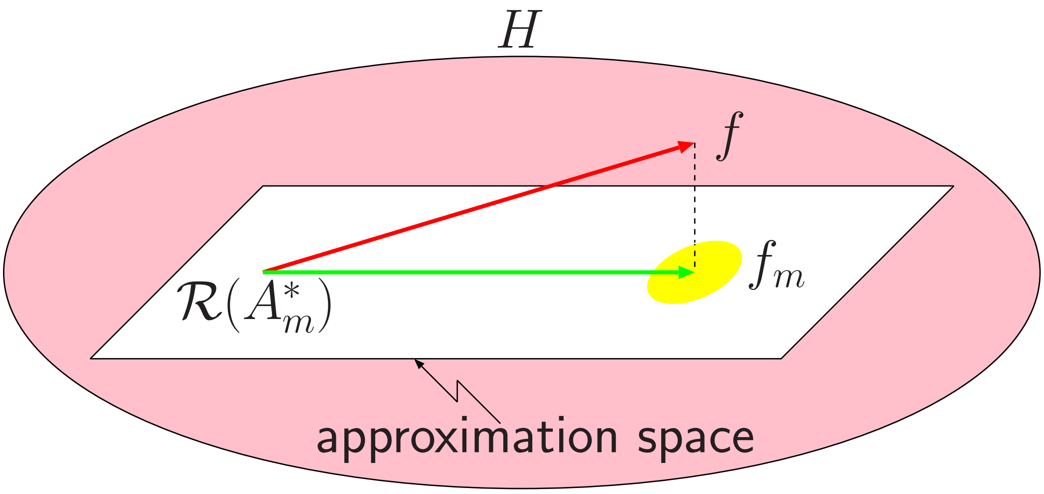
representation of sampling operator A_m

$$\begin{aligned} A_m &= \sum_{i=1}^{m} (e_i^{(m)} \otimes \overline{\psi_i}) \\ \psi_i(x) &= K(x, x_i) \\ K(x, x') &: \text{reproducing kernel} \end{aligned}$$

Projection learning

$$f_m = \underbrace{X_m A_m f}_{\substack{\text{signal}\\ \text{component}}} + \underbrace{X_m n^{(m)}}_{\substack{\text{noise}\\ \text{component}}}$$

minimize	$E_n \lVert X_m n^{(m)} \rVert^2$
under the constraint of	$X_m A_m f = P_{\mathcal{R}(A_m^*)} f$



projection learning operator

$$\underline{X_m = V_m^\dagger A_m^* U_m^\dagger + Y_m(I_m - U_m U_m^\dagger)}$$

Q_m : noise correlation matrix

$U_m = A_m A_m^* + Q_m$

$V_m = A_m^* U_m^\dagger A_m$

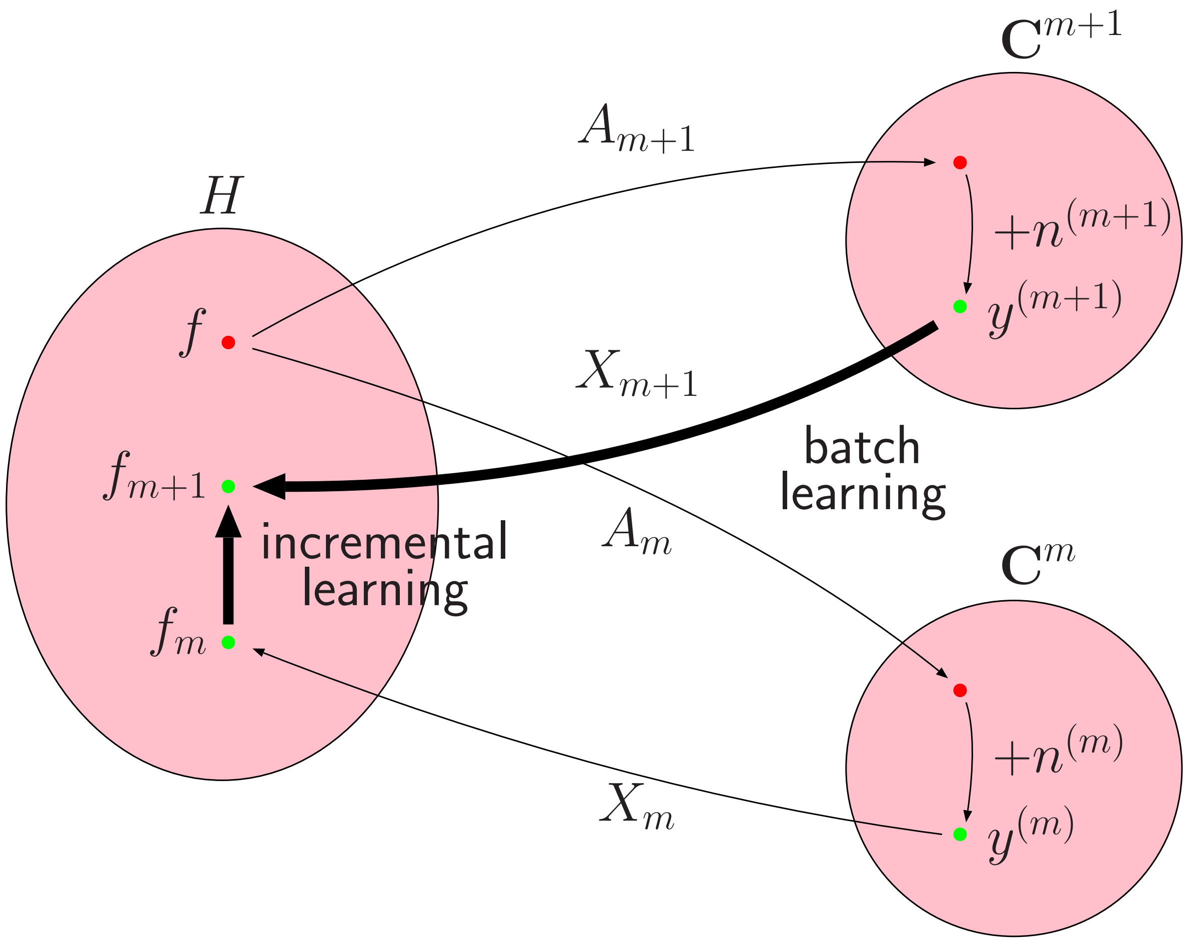
Y_m : arbirary operator

A_m^* : adjoint operator of A_m

$U_m^\dagger$: Moore-Penrose generalized inverse of U_m

Incremental learning

In practical situations, training example (x_{m+1}, y_{m+1}) is often added to further improve the generalization capability after f_m has been obtained.



Former research

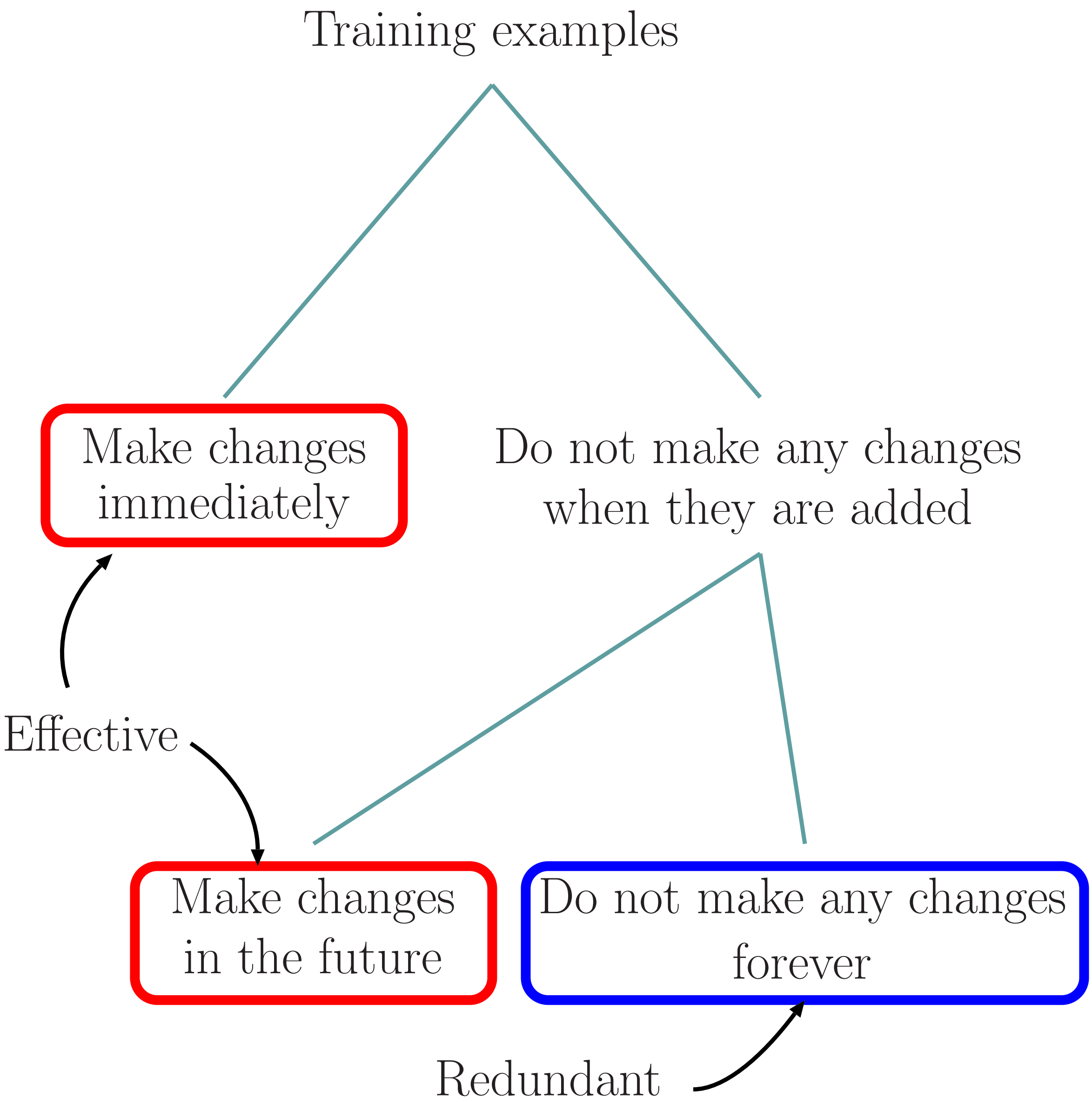
- Novel hidden units are added as new training examples are provided.
  → Too many hidden units!
- Resource Allocating Network (RAN). (Platt, 1991)
  Novelty criteria are introduced to control the increase of hidden units.
  → Old training examples tend to be forgotten!
- Some of the old training examples are stored in buffer. (Yoneda *et al.* 1992; Yamakawa *et al.* 1993)
  Artificial examples are created. (Yamauchi *et al.* 1997)
  → Gereralization capability is not guaranteed!
- Natural gradient method. (Amari 1998)
  Same result as batch learning can be asymptotically obtained.
  → # of training examples is finite in practice!

Objectives

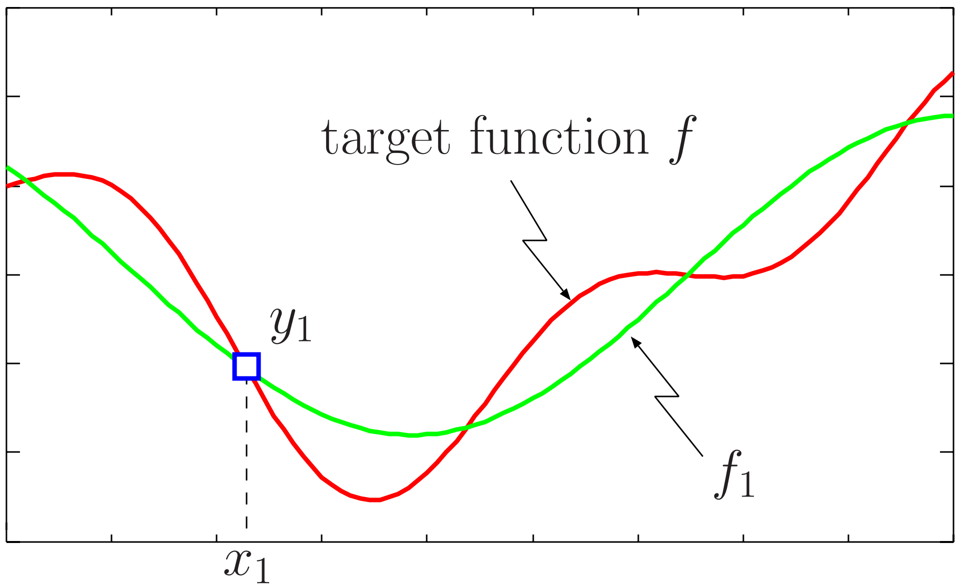
Development of an incremental learning method which gives the same result as batch learning when the number of training examples is finite.

Potentially effective training examples

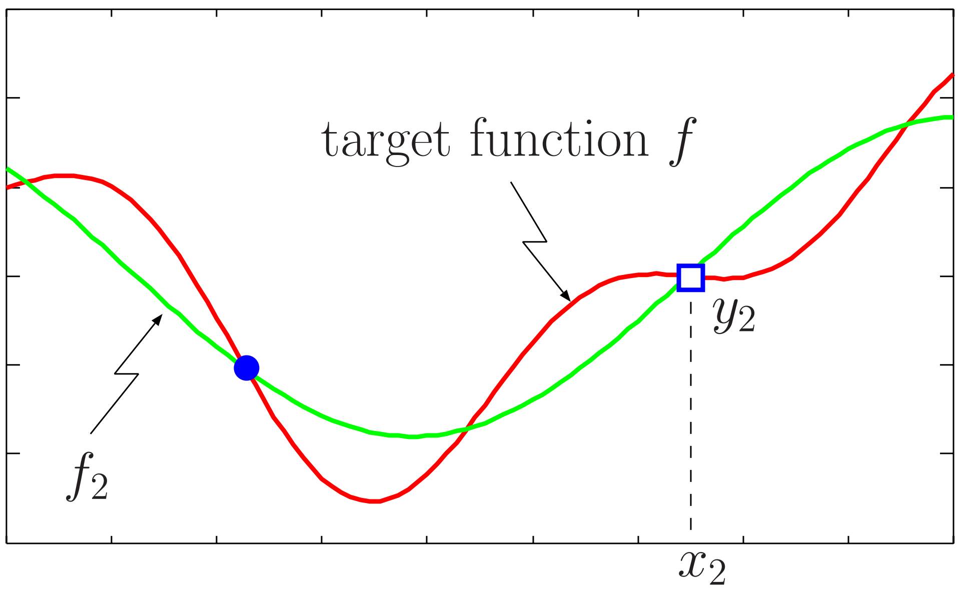
In traditional incremental learning methods,
the additional training example is rejected if $f_{m+1} = f_m$.
However, there exist some training examples
which are effective in the future.



Potentially effective training examples



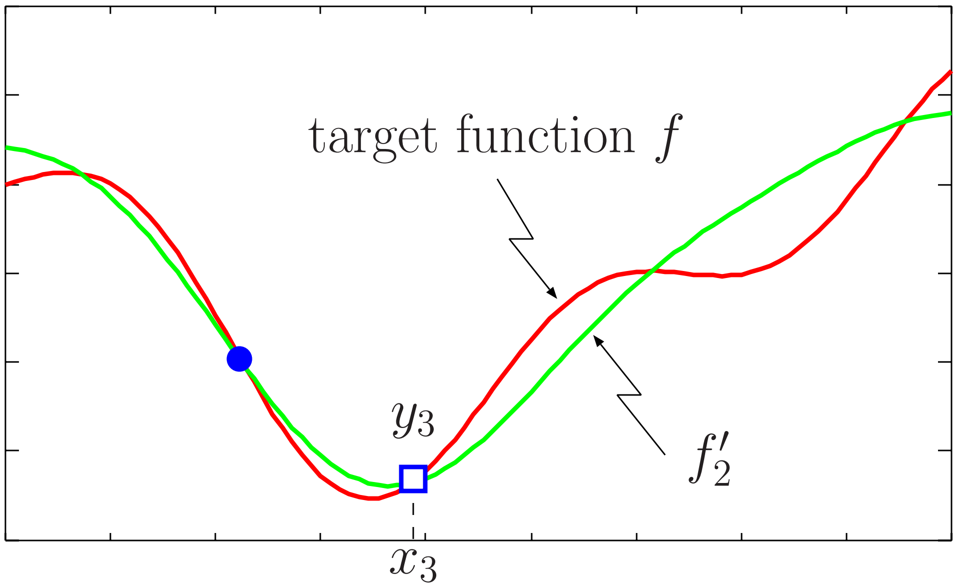
Learning result from (x_1, y_1).



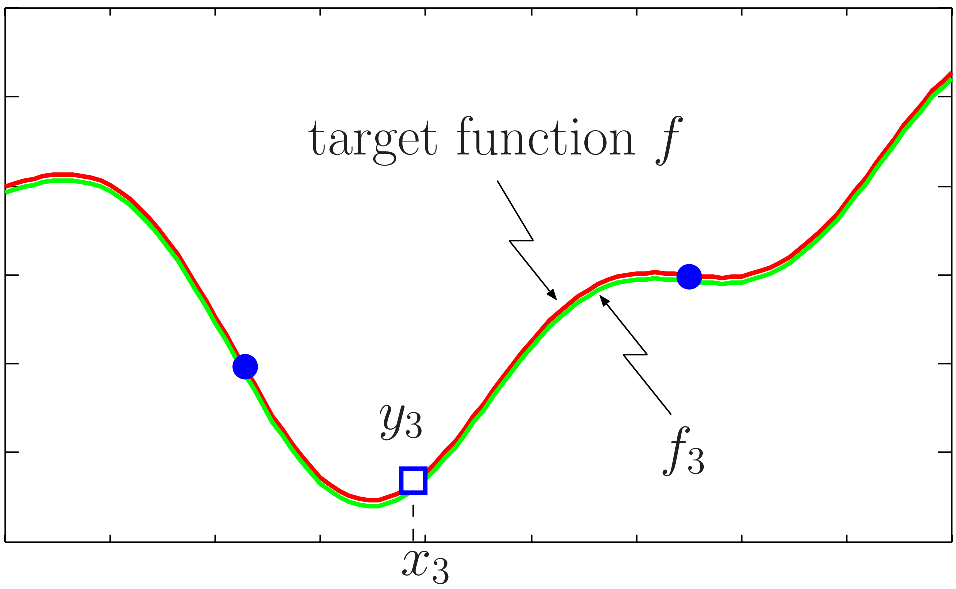
(x_2, y_2) is added to f_1.

(x_2, y_2) does not make any changes.

Potentially effective training examples



(x_2, y_2) is rejected and (x_3, y_3) is added to f_1.



(x_2, y_2) is used and (x_3, y_3) are added to f_2.

Incremental projection learning

(i) When $\xi_{m+1} = 0$,

(x_{m+1}, y_{m+1}) is rejected since it is redundant.

(ii) When $\xi_{m+1} \neq 0$,

$$\underline{f_{m+1} = f_m + \beta_{m+1}\zeta_{m+1}}$$

β_{m+1} : scalar determined from (x_{m+1}, y_{m+1})

ζ_{m+1} : function calculated from x_{m+1}.

(a) When $\psi_{m+1} \in \mathcal{R}(A_m^*)$,

$$\zeta_{m+1} = \frac{\tilde{\xi}_{m+1}}{\alpha_{m+1} + \langle \tilde{\xi}_{m+1}, \xi_{m+1} \rangle}$$

(b) When $\psi_{m+1} \notin \mathcal{R}(A_m^*)$,

$$\zeta_{m+1} = \frac{\tilde{\psi}_{m+1}}{\tilde{\psi}_{m+1}(x_{m+1})}$$

Other variables

$\Gamma_{m+1} = \sum_{i=1}^{m}(e_i^{(m+1)} \otimes \overline{e_i^{(m)}})$	$\tilde{\psi}_{m+1} = P_{\mathcal{N}(A_m)}\psi_{m+1}$
$\psi_{m+1} = K(x, x_{m+1})$	$\xi_{m+1} = \psi_{m+1} - A_m^* t_{m+1}$
$\sigma_{m+1} = E_n(n_{m+1}^2)$	$\tilde{\xi}_{m+1} = V_m^\dagger \xi_{m+1}$
$q_{m+1} = E_n(n_{m+1} n^{(m)})$	$h_{m+1} = e_{m+1}^{(m+1)} - \Gamma_{m+1}(t_{m+1} + X_m^* \xi_{m+1})$
$s_{m+1} = A_m \psi_{m+1} + q_{m+1}$	$\alpha_{m+1} = \psi_{m+1}(x_{m+1}) + \sigma_{m+1} - \langle t_{m+1}, s_{m+1} \rangle$
$t_{m+1} = U_m^\dagger s_{m+1}$	$\beta_{m+1} = y_{m+1} - f_m(x_{m+1}) - \langle y^{(m)} - A_m f_m, t_{m+1} \rangle$

Bias and variance

Let E_n be the ensemble average over noise.
Let us measure the generalization error of f_m by

$$\begin{aligned} J_G &= E_n\|f_m - f\|^2 \\ &= \underbrace{\|E_n f_m - f\|^2}_{\text{Bias}} + \underbrace{E_n\|f_m - E_n f_m\|^2}_{\text{Variance}}. \end{aligned}$$

Let J_b and J_v be changes in bias and variance:

$$\begin{aligned} J_b &= \|E_n f_{m+1} - f\|^2 - \|E_n f_m - f\|^2, \\ J_v &= E_n\|f_{m+1} - E_n f_{m+1}\|^2 - E_n\|f_m - E_n f_m\|^2. \end{aligned}$$

Assume the mean of noise is zero.
For any additional training example (x_{m+1}, y_{m+1}) satisfying $\xi_{m+1} \neq 0$,

(a) when $\psi_{m+1} \notin \mathcal{R}(A_m^*)$,

$$J_b = 0 \quad \text{and} \quad J_v < 0.$$

(b) when $\psi_{m+1} \in \mathcal{R}(A_m^*)$,

$$J_b \leq 0 \quad \text{and} \quad J_v \geq 0.$$

Computer simulation 1

Mackey-Glass delay-difference equation

$$g(t+1) = \begin{cases} (1-b)g(t) + \frac{a\, g(t-\tau)}{1+g(t-\tau)^{10}} & : t \geq \tau + 1, \\ 0.3 & : 1 \leq t \leq \tau, \end{cases}$$

where $a = 0.2$, $b = 0.1$, $\tau = 17$.

Let us perform the interploation of $f(t)$:

$$f(-1 + \frac{t}{100}) = g(t + \tau + 1) \quad \text{for } 1 \leq t \leq 199$$

(a) IPL : $H = \mathcal{L}(\{x^n\}_{n=0}^{19})$. The inner product in H is

$$\langle f, g \rangle = \int_{-1}^{1} f(x)\overline{g(x)}dx$$

(b) M-RAN : (Yingwei *et al.* 1997, 1998)

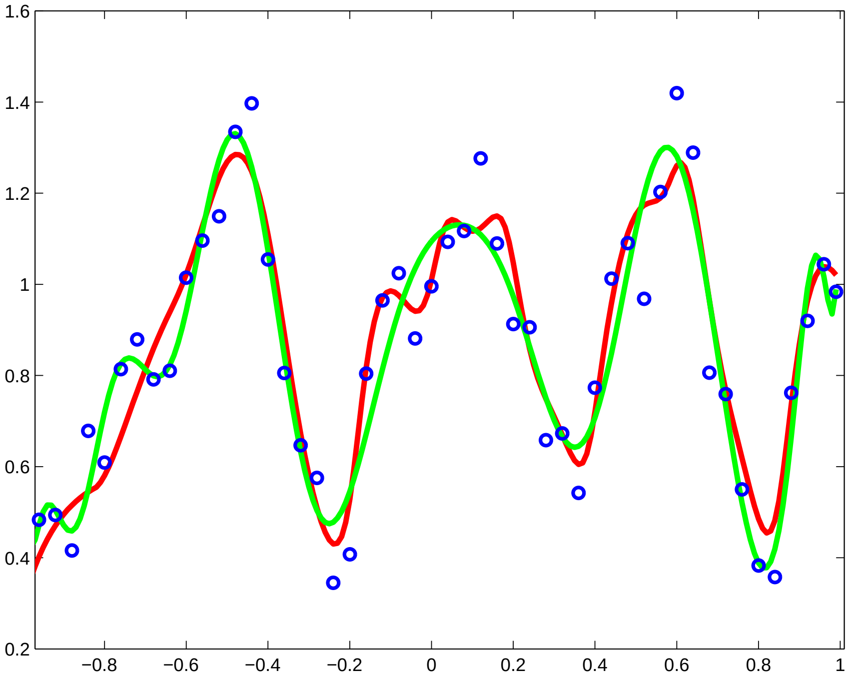
Parameters are assigned as

$\epsilon_{max} = 0.18$, $\epsilon_{min} = 0.04$, $\gamma = 0.96$, $e_{min} = 0.004$, $e'_{min} = 0.006$, $\kappa = 0.8$, P_0=1, $P_n = 0.19$, $M = 30$, $\delta = 0.000001$

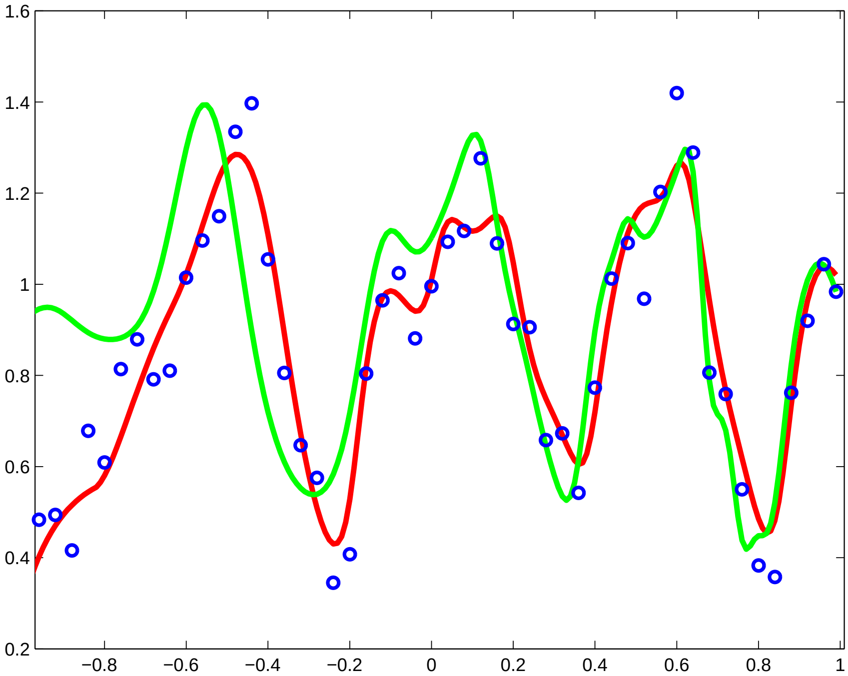
(c) BP (Batch learning) : # of hidden units is 20.

$$\text{Error} = \sum_{k=1}^{199} \left[f(-1 + \frac{k}{100}) - f_0(-1 + \frac{k}{100}) \right]^2$$

Computer simulation 1

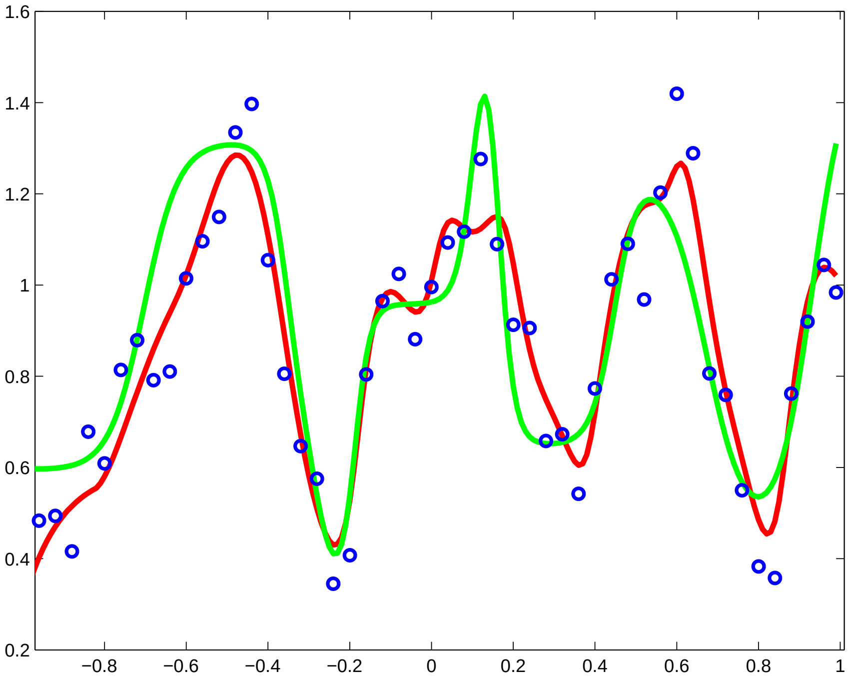


(a) IPL : Error=0.87



(b) M-RAN : Error=8.26

Computer simulation 1

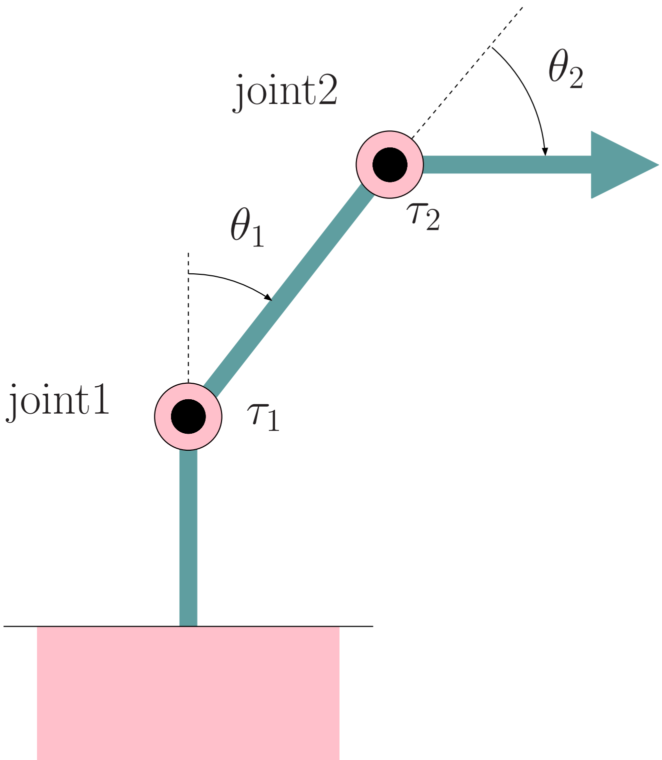


(c) BP : Error=2.97

- In M-RAN, old training examples are forgotten.
- The learning result of IPL does not depend on the order of training examples.
- IPL gives better generalization capability than M-RAN and BP.
- Tuning of parameters in IPL is easier than M-RAN.

Computer simulation 2

Learning of sensorimotor maps of 2-joint robot arm



The tip of robot arm moves in a plane.

Sensorimotor map

Mapping from posture $\theta_1, \dot{\theta}_1, \ddot{\theta}_1, \theta_2, \dot{\theta}_2, \ddot{\theta}_2$
to torque τ_1, τ_2 which should be applied to joints:

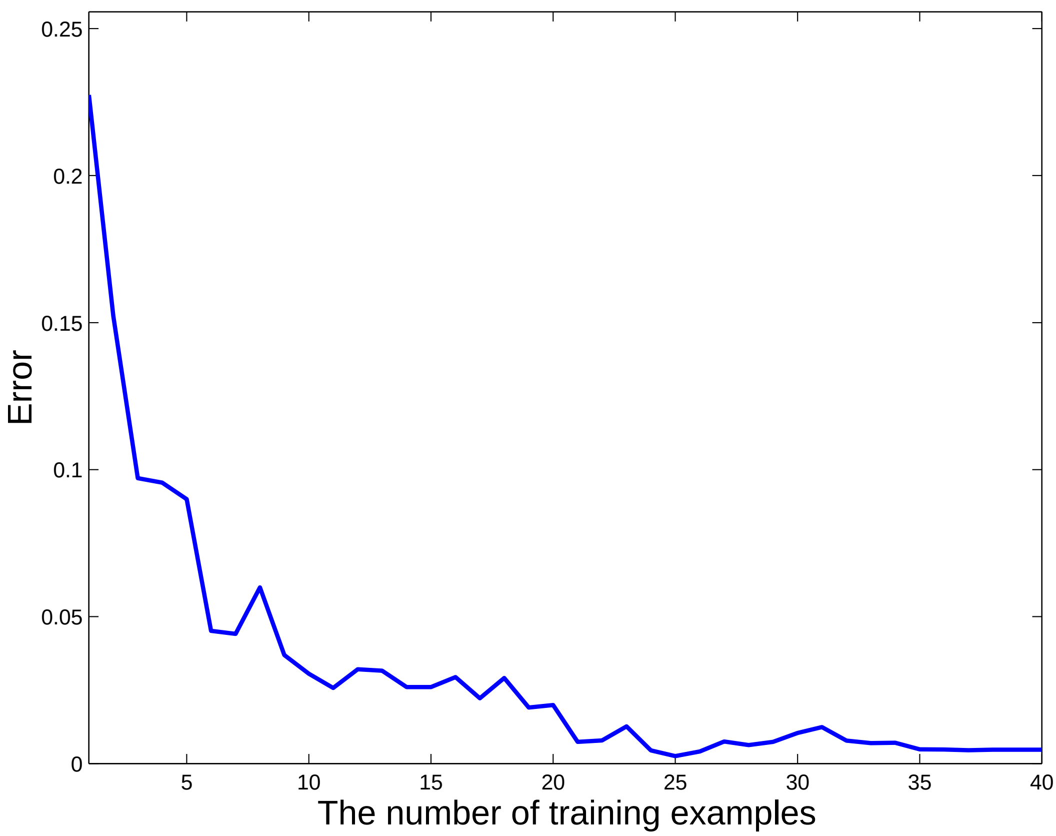
$$\text{joint 1: } \tau_1 = f_1(\theta_1, \dot{\theta}_1, \ddot{\theta}_1, \theta_2, \dot{\theta}_2, \ddot{\theta}_2)$$
$$\text{joint 2: } \tau_2 = f_2(\theta_1, \dot{\theta}_1, \ddot{\theta}_1, \theta_2, \dot{\theta}_2, \ddot{\theta}_2)$$

Computer simulation 2

Function spaces which include sensorimotor maps

$$\begin{aligned} H_1 &= \{\ddot{\theta}_1,\ \ddot{\theta}_1 \cos\theta_2,\ \ddot{\theta}_2,\ \ddot{\theta}_2 \cos\theta_2,\ \dot{\theta}_2^2 \sin\theta_2, \\ &\quad \dot{\theta}_1\dot{\theta}_2 \sin\theta_2,\ \sin\theta_1,\ \sin\theta_1 \cos\theta_2,\ \sin\theta_2 \cos\theta_1\} \\ H_2 &= \{\ddot{\theta}_1,\ \ddot{\theta}_1 \cos\theta_2,\ \ddot{\theta}_2, \\ &\quad \dot{\theta}_1^2 \sin\theta_2,\ \sin\theta_1 \cos\theta_2,\ \sin\theta_2 \cos\theta_1\} \end{aligned}$$

$$\text{Error} = \frac{1}{c} \int (f(\theta) - f_0(\theta))^2 \, d\theta$$



Conclusion

1. We proposed a new incremental learning method which provides exactly the same generalization capability as that obtained by batch learning.

2. Tune of parameters is easier than former methods.

3. Due to the reproducing kernel, the proposed method is applicable to the problem where input dimension is very large.