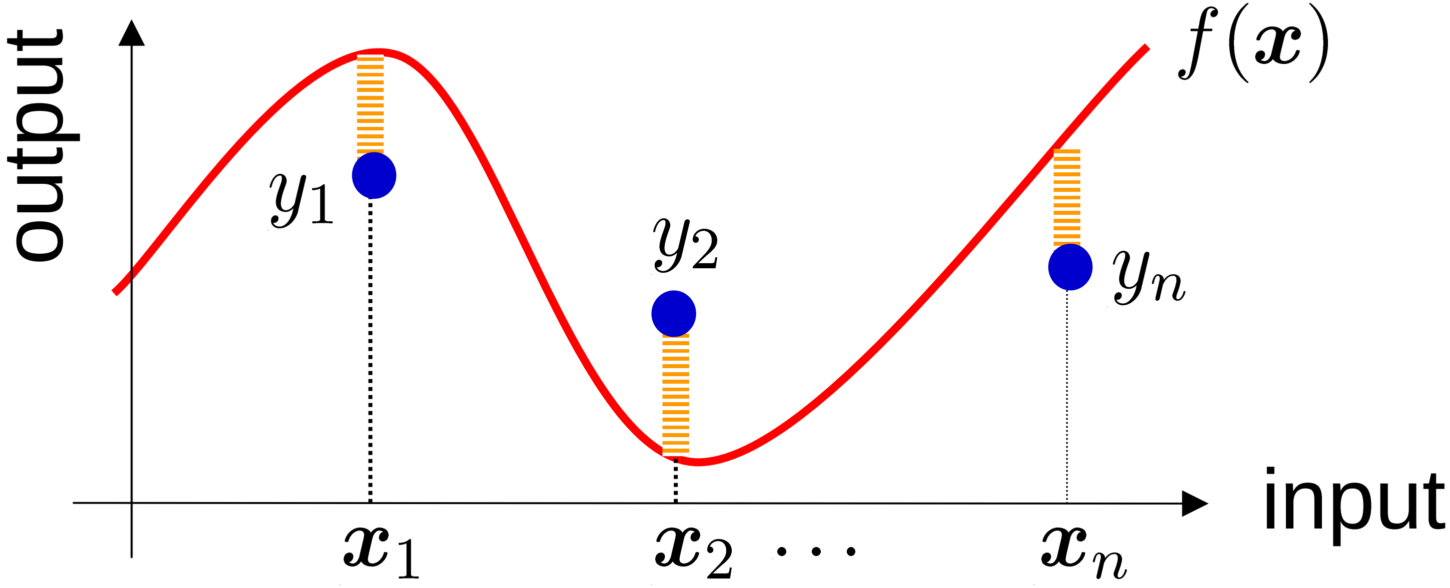
Active Learning for Regression: Algorithms and Applications

Masashi Sugiyama
Tokyo Institute of Technology
sugi@cs.titech.ac.jp
http://sugiyama-www.cs.titech.ac.jp/~sugi/

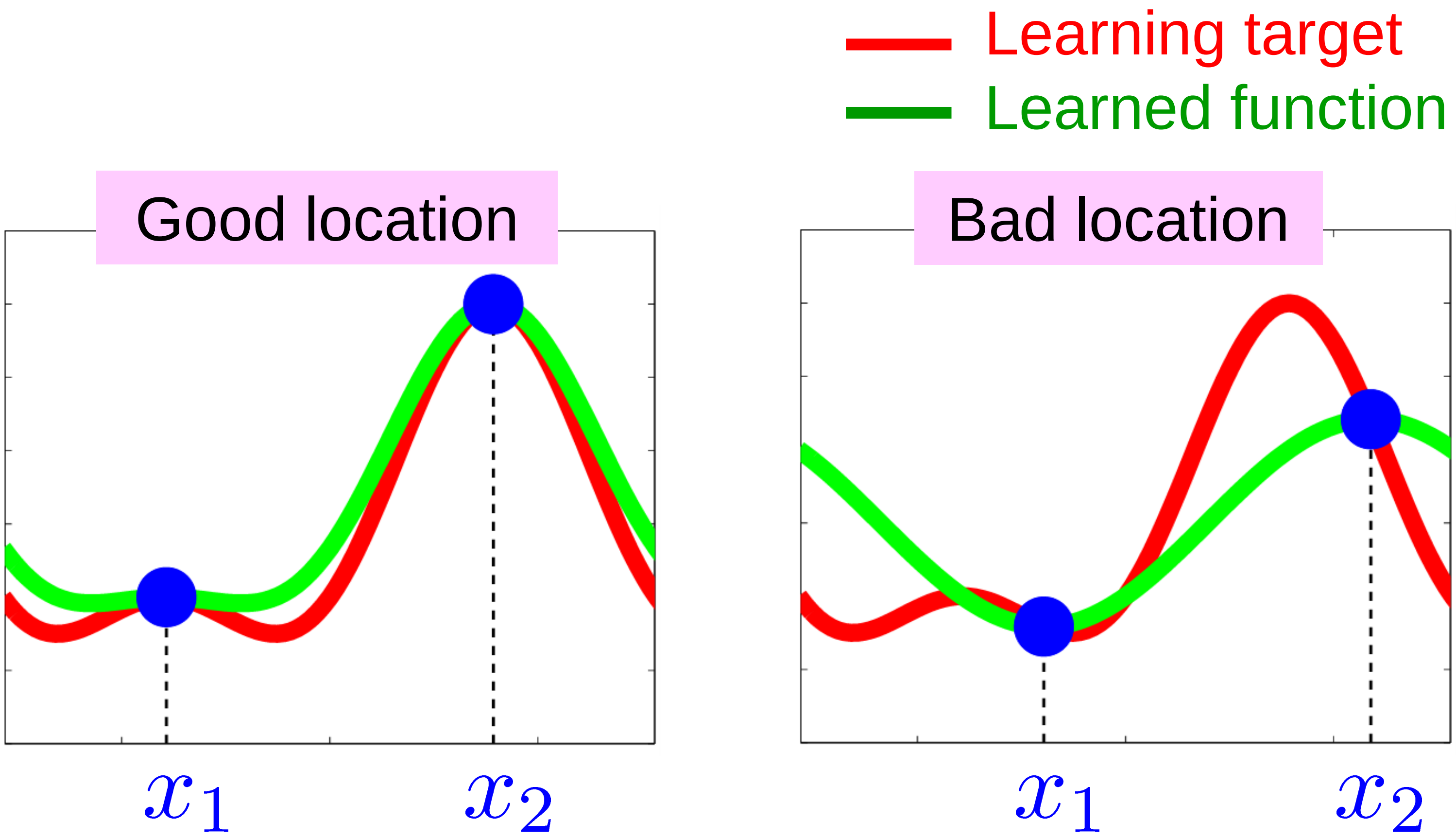
Supervised Learning

- Learn a target function $f(\boldsymbol{x})$ from input-output samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.
- This allows us to predict outputs of unseen inputs: "generalization"



Active Learning (AL)

- Choice of input location affects the generalization performance.
- Goal: choose the best input location!



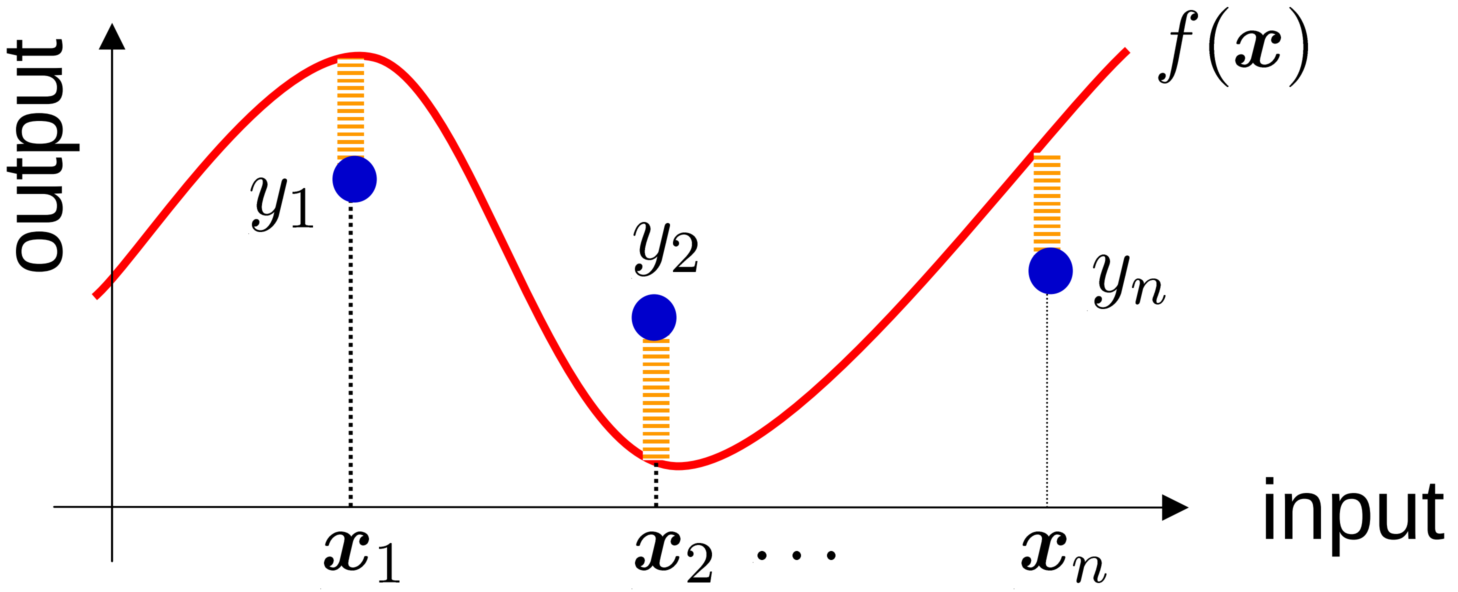
Motivation of AL

- AL is effective when sampling cost is high.
- Ex.) Predicting the length of a patient's life
  - Input x : features of patients
  - Output y : the length of life
  - In order to observe the outputs, the patients need to be nursed for years
- It is highly valuable to optimize the choice of input locations!

Organization of My Talk

1. Formulation.
2. AL for correctly specified models.
3. AL for misspecified models.
4. Choosing inputs from unlabeled samples.
5. AL with model selection.

Problem Formulation



- Training samples: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$
  - Input: $\boldsymbol{x}_i \overset{i.i.d.}{\sim} p_{train}(\boldsymbol{x})$
  - Output: $y_i = f(\boldsymbol{x}_i) + \varepsilon_i$
  - Noise: $\varepsilon_i \overset{i.i.d.}{\sim}$ mean 0, unknown variance σ^2

Problem Formulation

- Use a linear model for learning:

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi_i(\boldsymbol{x})$$

α_i : parameter
$\varphi_i(\boldsymbol{x})$: basis function

- Generalization error:

$$G = \int \left(\widehat{f}(\boldsymbol{x}) - f(\boldsymbol{x}) \right)^2 p_{test}(\boldsymbol{x}) d\boldsymbol{x}$$

  - $p_{test}(\boldsymbol{x})$:Test input density (assumed known)

- Goal of AL: Choose $p_{train}(\boldsymbol{x})$ so that the generalization error is minimized.

$$\min_{p_{train}} G$$

Difficulty of AL

$$\min_{p_{train}} G$$

$$G = \int \left(\widehat{f}(\boldsymbol{x}) - f(\boldsymbol{x}) \right)^2 p_{test}(\boldsymbol{x}) d\boldsymbol{x}$$

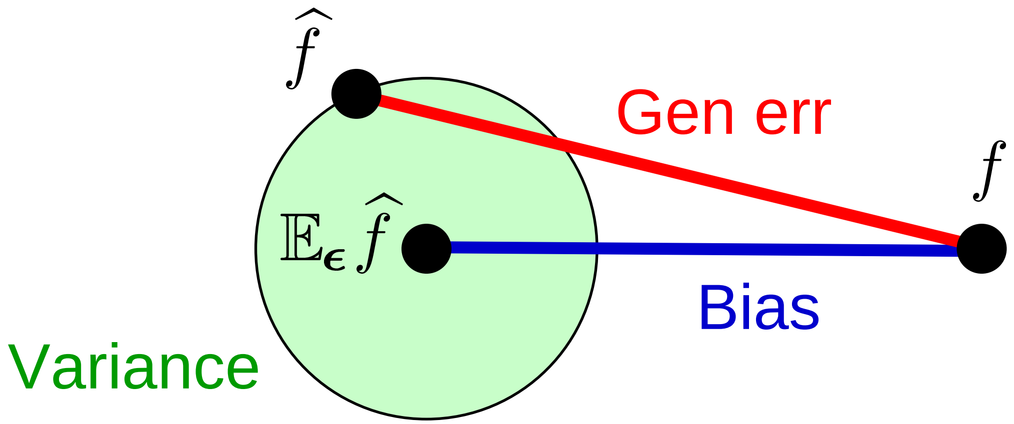
- Gen err is unknown.
- In AL, gen error needs to be estimated before observing output samples $\{y_i\}_{i=1}^{n}$.
- Thus standard gen err estimators such as cross-validation or Akaike's information criterion cannot be used in AL.

Bias-Variance Decomposition

$$\mathbb{E}_{\boldsymbol{\epsilon}} G = B + V$$

$\mathbb{E}_{\boldsymbol{\epsilon}}$:Expectation over noise

- Gen err: $G = \int \left(\widehat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2 p_{test}(\boldsymbol{x}) d\boldsymbol{x}$
- Bias: $B = \int \left(\mathbb{E}_{\boldsymbol{\epsilon}} \widehat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2 p_{test}(\boldsymbol{x}) d\boldsymbol{x}$
- Variance: $V = \mathbb{E}_{\boldsymbol{\epsilon}} \int \left(\mathbb{E}_{\boldsymbol{\epsilon}} \widehat{f}(\boldsymbol{x}) - \widehat{f}(\boldsymbol{x})\right)^2 p_{test}(\boldsymbol{x}) d\boldsymbol{x}$



Bias and Variance

- Bias: depends on the unknown target function $f(\boldsymbol{x})$, so it is not possible to estimate it before observing output samples $\{y_i\}_{i=1}^n$.

$$B = \int \left(\mathbb{E}_{\boldsymbol{\epsilon}} \widehat{f}(\boldsymbol{x}) - f(\boldsymbol{x}) \right)^2 p_{test}(\boldsymbol{x}) d\boldsymbol{x}$$

- Variance: for linear estimator $\widehat{\boldsymbol{\alpha}} = \boldsymbol{L}\boldsymbol{y}$,

$$V = \mathbb{E}_{\boldsymbol{\epsilon}} \int \left(\mathbb{E}_{\boldsymbol{\epsilon}} \widehat{f}(\boldsymbol{x}) - \widehat{f}(\boldsymbol{x}) \right)^2 p_{test}(\boldsymbol{x}) d\boldsymbol{x}$$

$$= \sigma^2 \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top) \propto \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)$$

$$\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^\top \qquad \boldsymbol{U}_{i,j} = \int \varphi_i(\boldsymbol{x}) \varphi_j(\boldsymbol{x}) p_{test}(\boldsymbol{x}) d\boldsymbol{x}$$

Basic Strategy for AL

- For an unbiased linear estimator, we have

$$\mathbb{E}_{\boldsymbol{\epsilon}} G = B + V \propto \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)$$

- Thus, gen error can be minimized before observing output samples $\{y_i\}_{i=1}^n$!

$$\operatorname*{argmin}_{p_{train}} \mathbb{E}_{\boldsymbol{\epsilon}} G = \operatorname*{argmin}_{p_{train}} \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)$$

Organization of My Talk

1. Formulation.
2. AL for correctly specified models.
3. AL for misspecified models.
4. Choosing inputs from unlabeled samples.
5. AL with model selection.

Correctly Specified Models

- Assume that the target function is included in the model:

$$^{\exists}\boldsymbol{\alpha}^*,\ \widehat{f}(\boldsymbol{x};\boldsymbol{\alpha}^*) = f(\boldsymbol{x})$$

- Learn the parameters by ordinary least-squares (OLS):

$$\min_{\boldsymbol{\alpha}}\left[\sum_{i=1}^{n}\left(\hat{f}(\boldsymbol{x}_i) - y_i\right)^2\right] \qquad \hat{f}(\boldsymbol{x}) = \sum_{i=1}^{b}\alpha_i\varphi_i(\boldsymbol{x})$$

Properties of LS

- OLS estimator is linear:

$$\widehat{\boldsymbol{\alpha}} = \boldsymbol{L}\boldsymbol{y}$$

$$\boldsymbol{L} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top$$

$$\boldsymbol{X}_{i,j} = \varphi_j(\boldsymbol{x}_i)$$

$$\boldsymbol{y} = (y_1, \ldots, y_n)^\top$$

⇒ Variance is $V = \sigma^2 \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top) \propto \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)$

- OLS estimator is unbiased:

$$\mathbb{E}_{\boldsymbol{\epsilon}}\widehat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^*$$

⇒ Bias is $B = 0$

AL for Correctly Specified Models

- When OLS is used,

$$\mathbb{E}_{\boldsymbol{\epsilon}} G = \underbrace{B}_{=0} + \underbrace{V}_{\propto \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)}$$

- Thus

$$\underset{p_{train}}{\mathrm{argmin}}\, \mathbb{E}_{\boldsymbol{\epsilon}} G = \underset{p_{train}}{\mathrm{argmin}}\, \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)$$

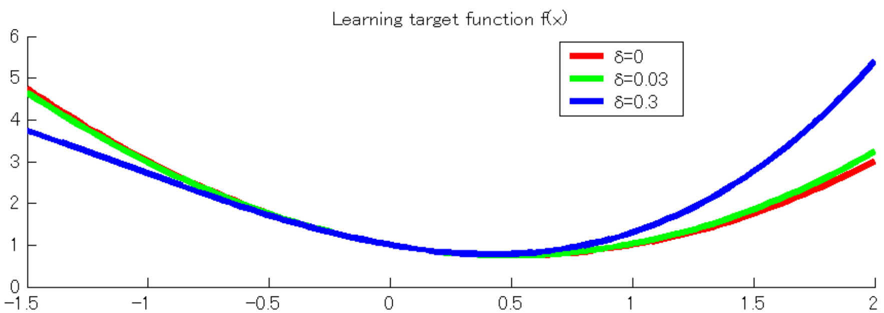
Fedorov, *Theory of Optimal Experiments*, Academic Press, 1972.

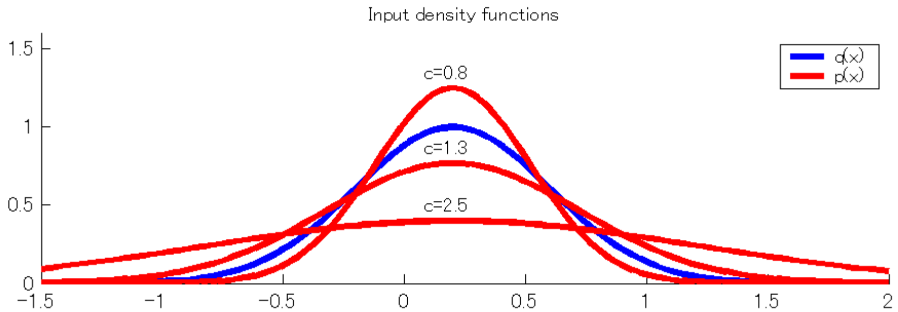
Illustrative Examples

$\delta = 0, 0.03, 0.3$

- Learning target: $f(x) = 1 - x + x^2 + \delta x^3$
- Model: $\hat{f}(x) = \alpha_1 + \alpha_2 x + \alpha_3 x^2$
- Test input density: $\mathcal{N}(0.2, (0.4)^2)$
- Training input density: $\mathcal{N}(0.2, (0.4c)^2)$

$c = 0.8, 0.9, 1.0, \ldots, 2.5$





Obtained Generalization Error

Mean±Std (1000 trials)

	$\delta = 0$	$\delta = 0.03$	$\delta = 0.3$
OLS-AL	1.45±1.82	2.56±2.24	113±63.7
Passive	3.10±2.61	3.13±2.61	5.75±3.09

- When model is correctly specified, OLS-AL works well.
- Even when model is slightly misspecified, the performance degrades significantly.
- When model is highly misspecified, the performance is very poor.

OLS-based AL: Summary

$$\min_{p_{train}} \operatorname{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)$$

$$\{\boldsymbol{x}_i\}_{i=1}^n \overset{i.i.d.}{\sim} p_{train}(\boldsymbol{x})$$

$$\boldsymbol{U}_{i,j} = \int \varphi_i(\boldsymbol{x})\varphi_j(\boldsymbol{x})p_{test}(\boldsymbol{x})d\boldsymbol{x}$$

$$\boldsymbol{X}_{i,j} = \varphi_j(\boldsymbol{x}_i)$$

- Pros:
  - Gen err estimation is exact.
  - Easy to implement.
- Cons:
  - Correctly specified models are not available in practice.
  - Performance degradation for model misspecification is significant.

Organization of My Talk

1. Formulation.
2. AL for correctly specified models.
3. AL for misspecified models.
4. Choosing inputs from unlabeled samples.
5. AL with model selection.

Misspecified Models

- Consider general cases where the target function is not included in the model:

$$^{\forall}\boldsymbol{\alpha},\ \widehat{f}(\boldsymbol{x};\boldsymbol{\alpha}) \neq f(\boldsymbol{x})$$

- However, if the model is completely misspecified, learning itself is meaningless (need model selection, discussed later)
- Here we assume that the model is approximately correct.

Orthogonal Decomposition

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) + r(\boldsymbol{x})$$

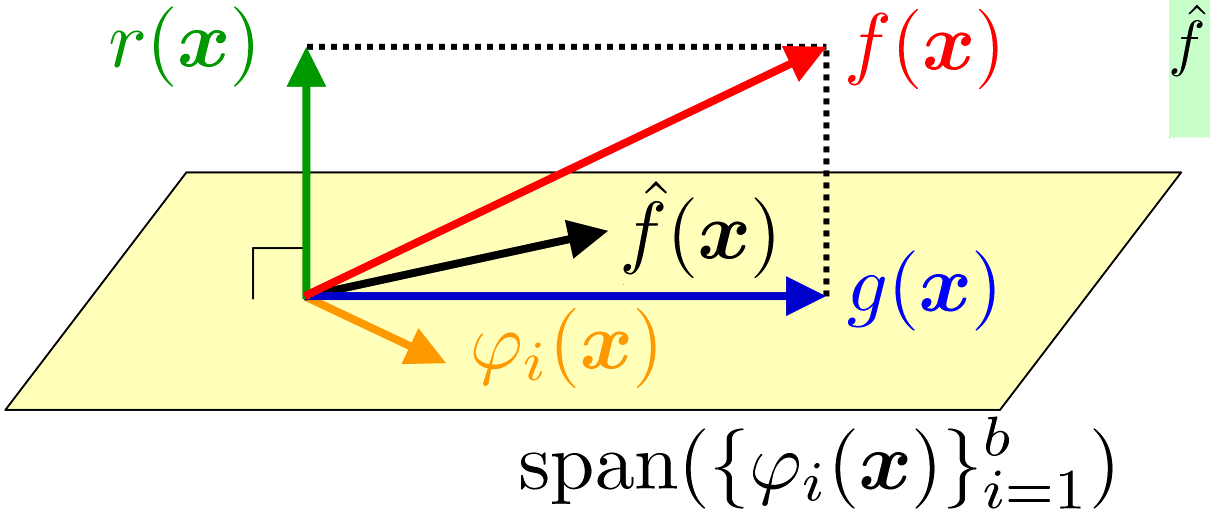
$$g(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i^* \varphi_i(\boldsymbol{x})$$

$$\int \varphi_i(\boldsymbol{x}) r(\boldsymbol{x}) p_{test}(\boldsymbol{x}) d\boldsymbol{x} = 0$$

($\varphi_i(\boldsymbol{x})$ and $r(\boldsymbol{x})$ are orthogonal)

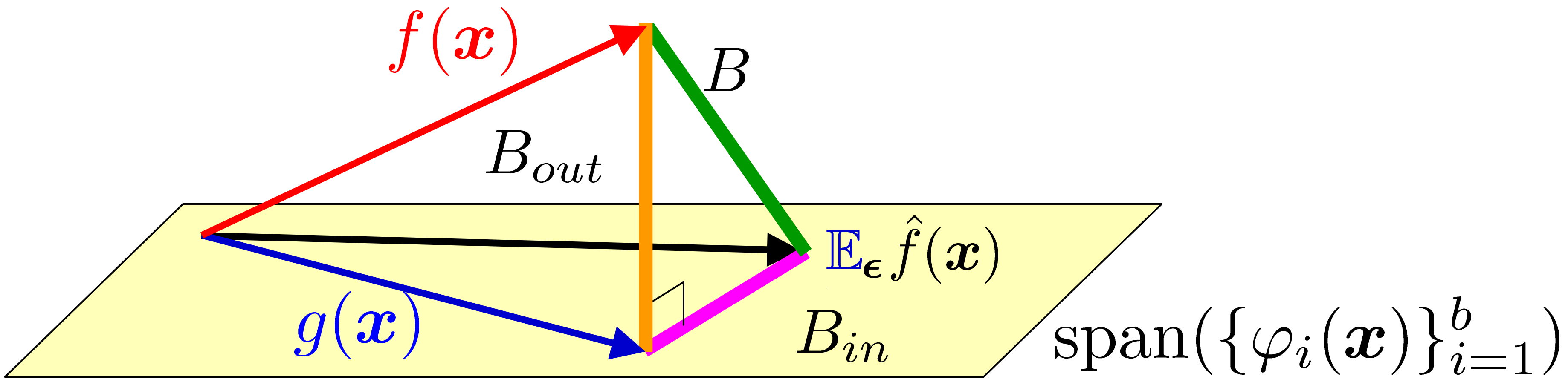
- Approximately correct model: $r(\boldsymbol{x}) \approx 0$

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi_i(\boldsymbol{x})$$



Further Decomposition of Bias

- Bias: $$B = \int \left(\mathbb{E}_{\boldsymbol{\epsilon}} \widehat{f}(\boldsymbol{x}) - f(\boldsymbol{x}) \right)^2 p_{test}(\boldsymbol{x}) d\boldsymbol{x}$$
  $$= B_{out} + B_{in}$$
- Out-model bias: $$B_{out} = \int \left(g(\boldsymbol{x}) - f(\boldsymbol{x}) \right)^2 p_{test}(\boldsymbol{x}) d\boldsymbol{x}$$
- In-model bias: $$B_{in} = \int \left(\mathbb{E}_{\boldsymbol{\epsilon}} \widehat{f}(\boldsymbol{x}) - g(\boldsymbol{x}) \right)^2 p_{test}(\boldsymbol{x}) d\boldsymbol{x}$$



Difficulty of AL for Misspecified Models

$$B = B_{out} + B_{in}$$

- Out-model bias remains, so bias cannot be zero.
- Out-model bias is constant, so it can be ignored.
- However, OLS does not reduce in-model bias to zero.

$$B_{in} \neq 0$$

- "Covariate shift" is the cause!

Covariate Shift

- Training and test inputs follow different distributions:

$$p_{train}(\boldsymbol{x}) \neq p_{test}(\boldsymbol{x})$$

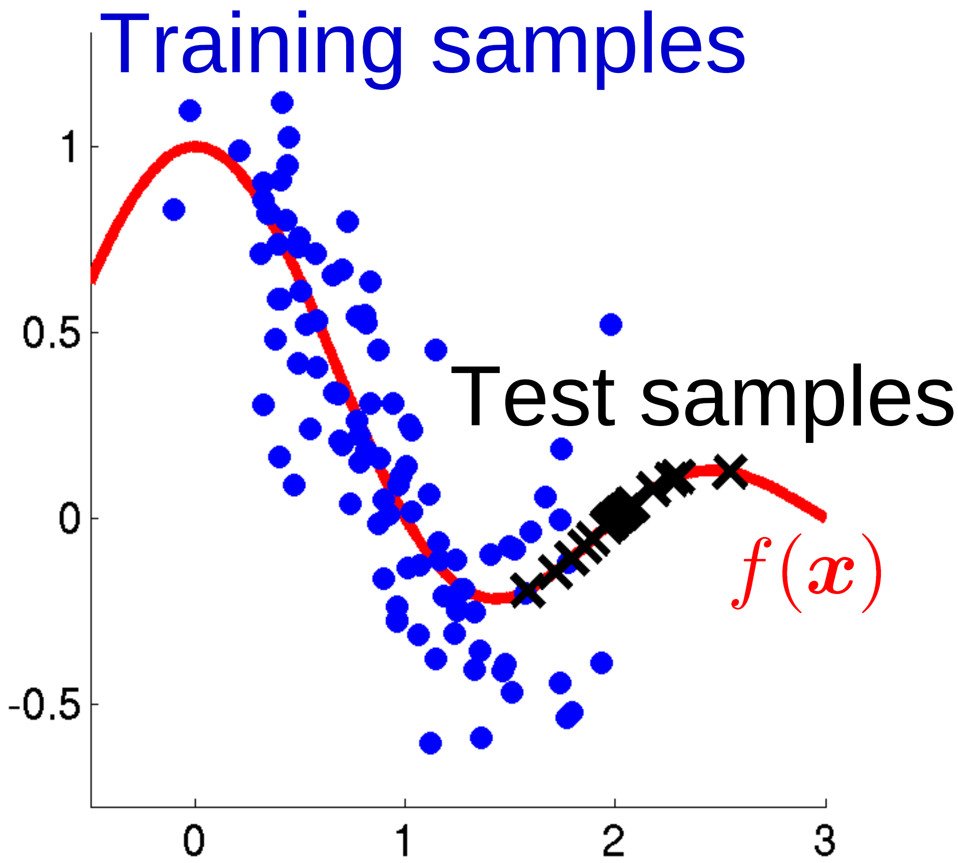
Covariate = Input

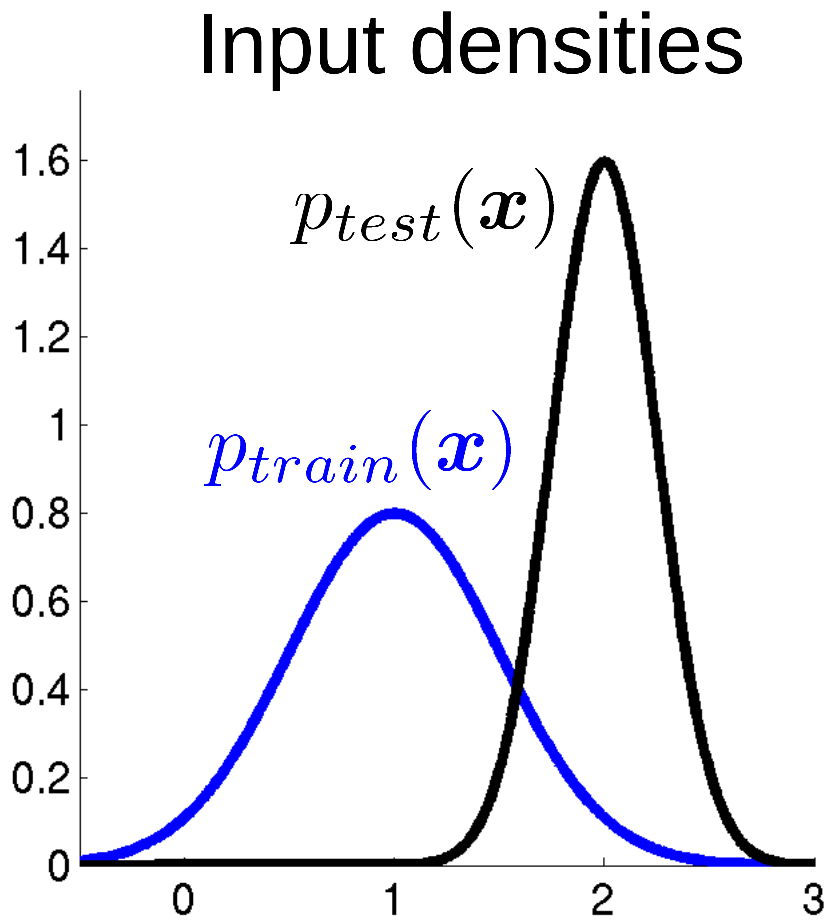
- In AL, covariate shift always occurs!
- Difference of input distributions causes OLS not to reduce in-model bias to zero.

$$\mathbb{E}_{\boldsymbol{\epsilon}} \widehat{\boldsymbol{\alpha}} \neq \boldsymbol{\alpha}^*$$

Shimodaira, Improving predictive inference under covariate shift by weighting the log-likelihood function, *Journal of Statistical Planning and Inference*, vol. 90, pp. 227-244, 2000.

Example of Covariate Shift





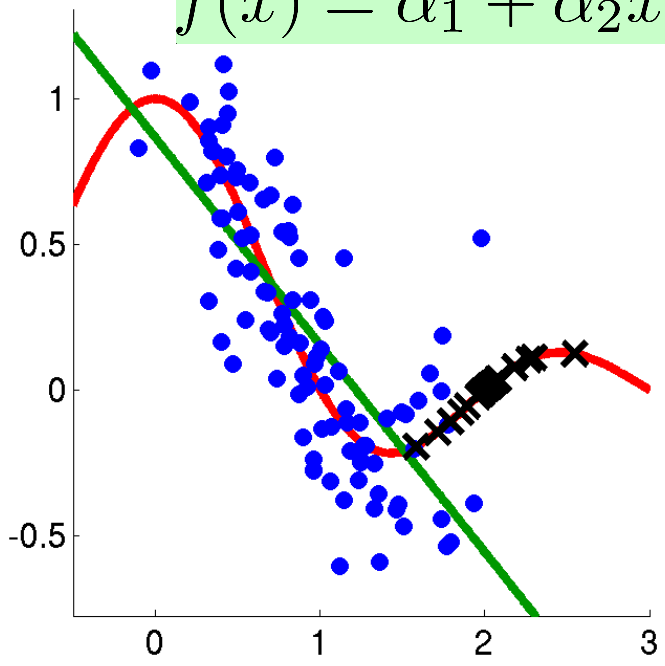
Bias of OLS under Covariate Shift

$$\min_{\boldsymbol{\alpha}} \left[\sum_{i=1}^{n} \left(\hat{f}(\boldsymbol{x}_i) - y_i \right)^2 \right]$$

- OLS:
  - Unbiased for correctly specified models.
  - For misspecified models, in-model bias remains even asymptotically.

$$\lim_{n \to \infty} B_{in} \neq 0$$

$$\hat{f}(x) = \alpha_1 + \alpha_2 x$$



The Law of Large Numbers

- Sample average converges to the population mean:

$$\frac{1}{n}\sum_{i=1}^{n}\mathrm{loss}(\boldsymbol{x}_i) \longrightarrow \int \mathrm{loss}(\boldsymbol{x})p_{train}(\boldsymbol{x})d\boldsymbol{x}$$

$$\boldsymbol{x}_i \overset{i.i.d.}{\sim} p_{train}(\boldsymbol{x})$$

- We want to estimate the expectation over test distribution using training samples (following training distribution).

$$\int \mathrm{loss}(\boldsymbol{x})p_{test}(\boldsymbol{x})d\boldsymbol{x}$$

Importance-Weighted Average

- Importance: the ratio of input densities

$$\frac{p_{test}(\boldsymbol{x})}{p_{train}(\boldsymbol{x})}$$

- Importance-weighted average:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{p_{test}(\boldsymbol{x}_i)}{p_{train}(\boldsymbol{x}_i)}\mathrm{loss}(\boldsymbol{x}_i) \qquad \boldsymbol{x}_i \overset{i.i.d.}{\sim} p_{train}(\boldsymbol{x})$$

$$\longrightarrow \int \frac{p_{test}(\boldsymbol{x})}{p_{train}(\boldsymbol{x})}\mathrm{loss}(\boldsymbol{x})p_{train}(\boldsymbol{x})d\boldsymbol{x}$$

$$= \int \mathrm{loss}(\boldsymbol{x})p_{test}(\boldsymbol{x})d\boldsymbol{x}$$

(cf. importance sampling)

Importance-Weighted LS (WLS)

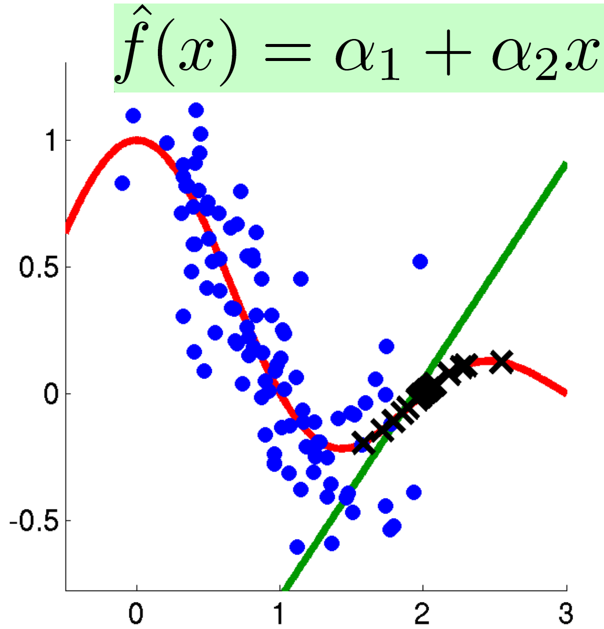
$$\min_{\boldsymbol{\alpha}} \left[\sum_{i=1}^{n} \frac{p_{test}(\boldsymbol{x}_i)}{p_{train}(\boldsymbol{x}_i)} \left(\widehat{f}(\boldsymbol{x}_i) - y_i \right)^2 \right]$$

- WLS:
  - Even for misspecified models, in-model bias vanishes asymptotically.

    $$\lim_{n \to \infty} B_{in} = 0$$

  - For approximately correct models, in-model bias is very small.

    $$0 \approx B_{in} \ll V$$



Importance-Weighted LS (WLS) 0

- WLS is linear:

$$\widehat{\boldsymbol{\alpha}} = \boldsymbol{L}\boldsymbol{y}$$

$$\boldsymbol{L} = (\boldsymbol{X}^\top \boldsymbol{D}\boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{D}$$

$$\boldsymbol{X}_{i,j} = \varphi_j(\boldsymbol{x}_i) \qquad \boldsymbol{y} = (y_1, \ldots, y_n)^\top$$

$$\boldsymbol{D} = \mathrm{diag}\left(\frac{p_{test}(\boldsymbol{x}_1)}{p_{train}(\boldsymbol{x}_1)}, \ldots, \frac{p_{test}(\boldsymbol{x}_n)}{p_{train}(\boldsymbol{x}_n)}\right)$$

- Thus variance is given by

$$V = \sigma^2 \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top) \propto \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)$$

AL for Approximately Correct Models using WLS

- Use WLS for learning:

$$\mathbb{E}_{\boldsymbol{\epsilon}} G = \underbrace{B_{out}}_{\text{Constant}} + \underbrace{B_{in}}_{\ll V} + \underbrace{V}_{\propto \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)}$$

- Thus

$$\underset{p_{train}}{\mathrm{argmin}}\, \mathbb{E}_{\boldsymbol{\epsilon}} G \approx \underset{p_{train}}{\mathrm{argmin}}\, \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)$$

Sugiyama, Active learning in approximately linear regression based on conditional expectation of generalization error, *Journal of Machine Learning Research*, vol.7, pp.141-166, 2006.

Obtained Generalization Error

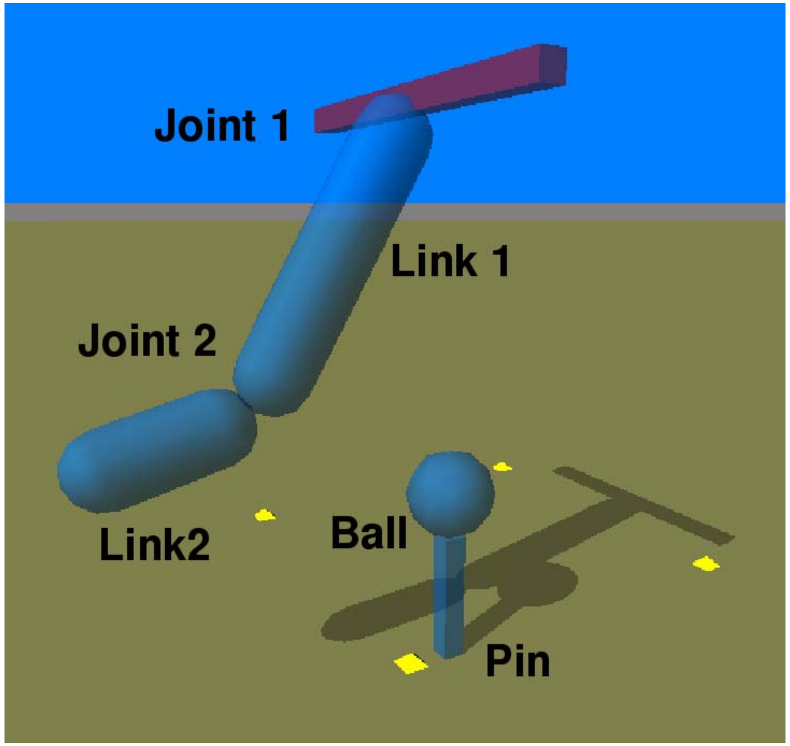
Mean±Std (1000 trials) T-test (95%)

	$\delta = 0$	$\delta = 0.03$	$\delta = 0.3$
WLS-AL	2.07±1.90	2.09±1.90	4.28±2.02
OLS-AL	1.45±1.82	2.56±2.24	113±63.7
Passive	3.10±2.61	3.13±2.61	5.75±3.09

- When model is exactly correct, OLS-AL works well.
- However, when model is misspecified, it is totally unreliable.
- WLS-AL works well even when model is misspecified.

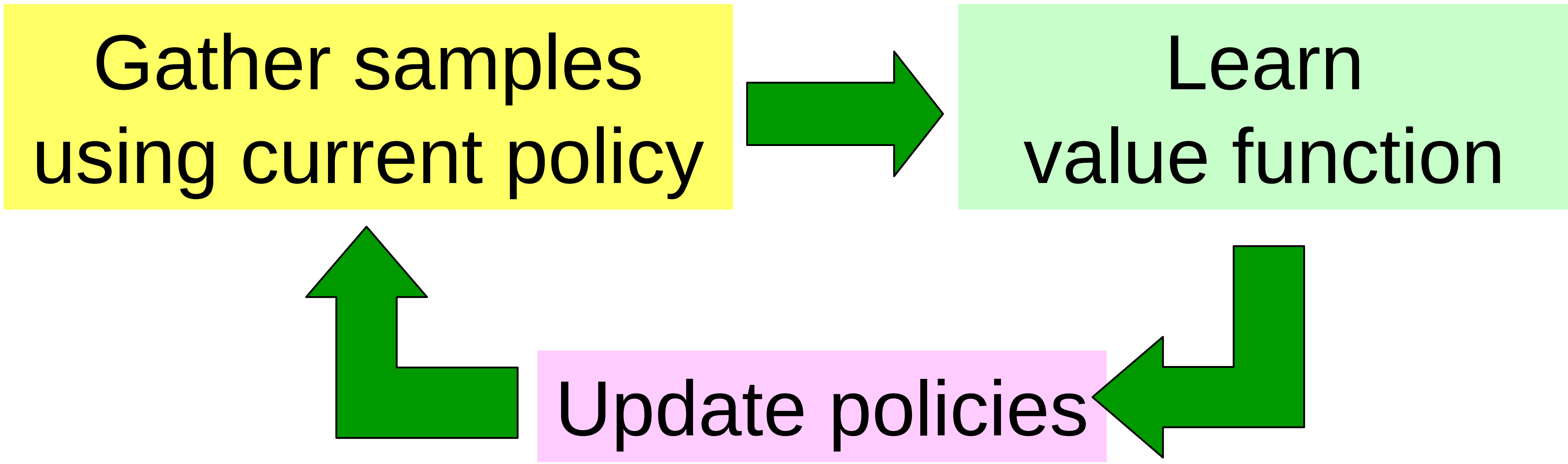
Application to Robot Control

- Golf robot: control the robot arm so that the ball is driven as far as possible.
  - State s : joint angles, angular velocities
  - Action a : torque to be applied to joints
- We use reinforcement learning (RL).
- In RL, reward r (carry distance of the ball) is given to the robot.
- Robot updates its control policy π so that the maximum amount of rewards is obtained.



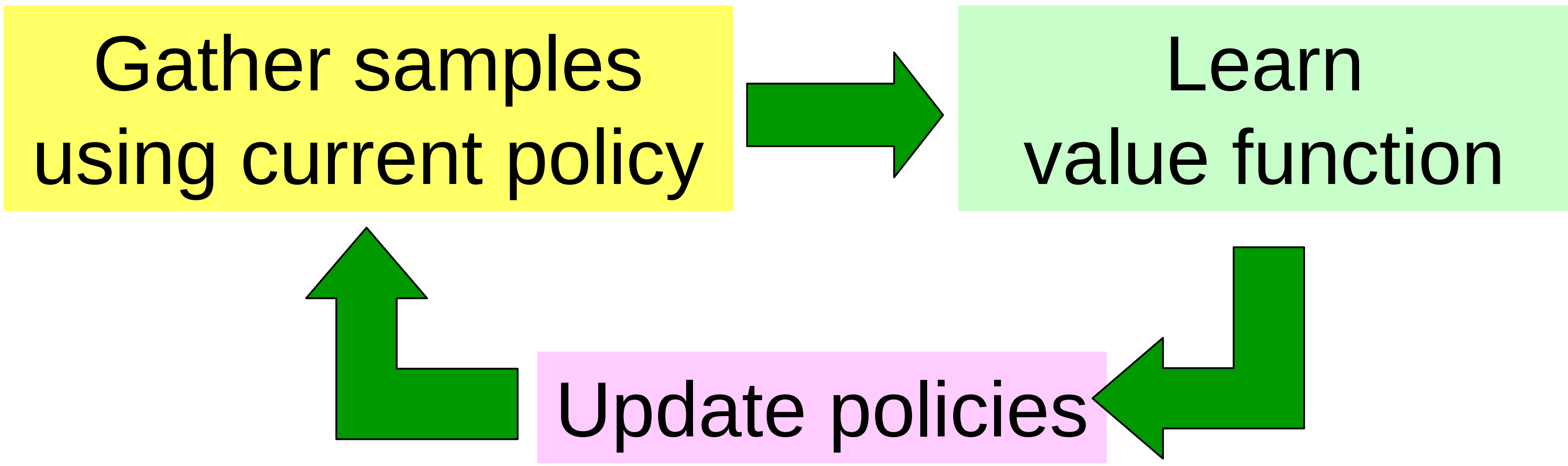
Policy Iteration

- Value function $Q^{\pi}(s,a)$: sum of rewards r when taking action a at state s and then following policy π.



Sutton & Barto, *Reinforcement Learning: An Introduction,* MIT Press, 1998.

Covariate Shift in Policy Iteration

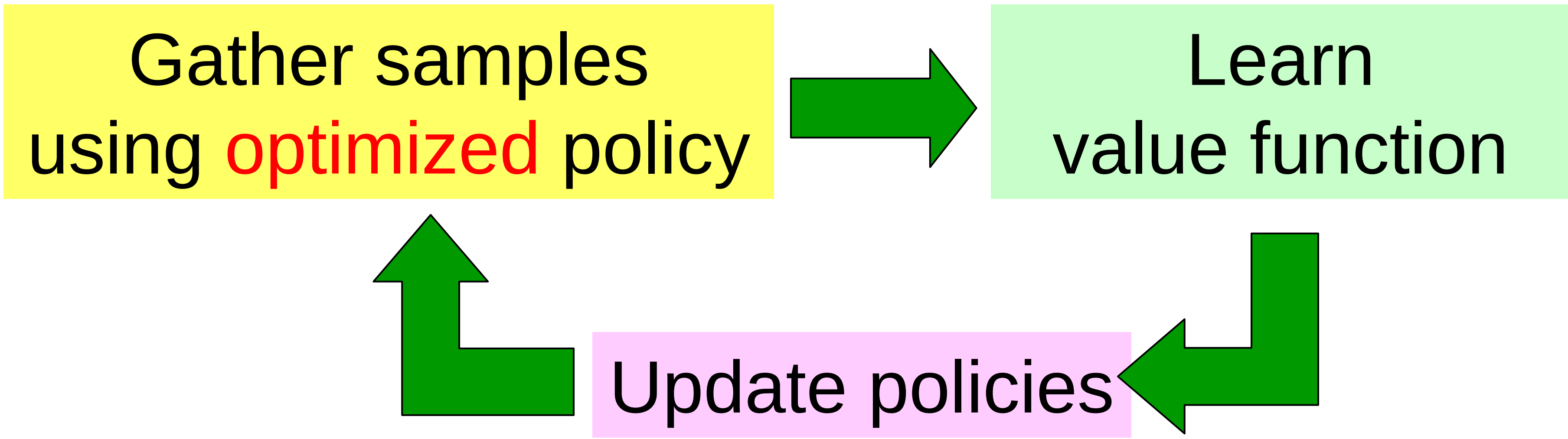


- When policies are updated, the distribution of s and a changes.
- Thus we need to use importance weighting for being consistent.

Hachiya, Akiyama, Sugiyama & Peters.
Adaptive importance sampling for value function approximation in off-policy reinforcement learning. *Neural Networks*, to appear

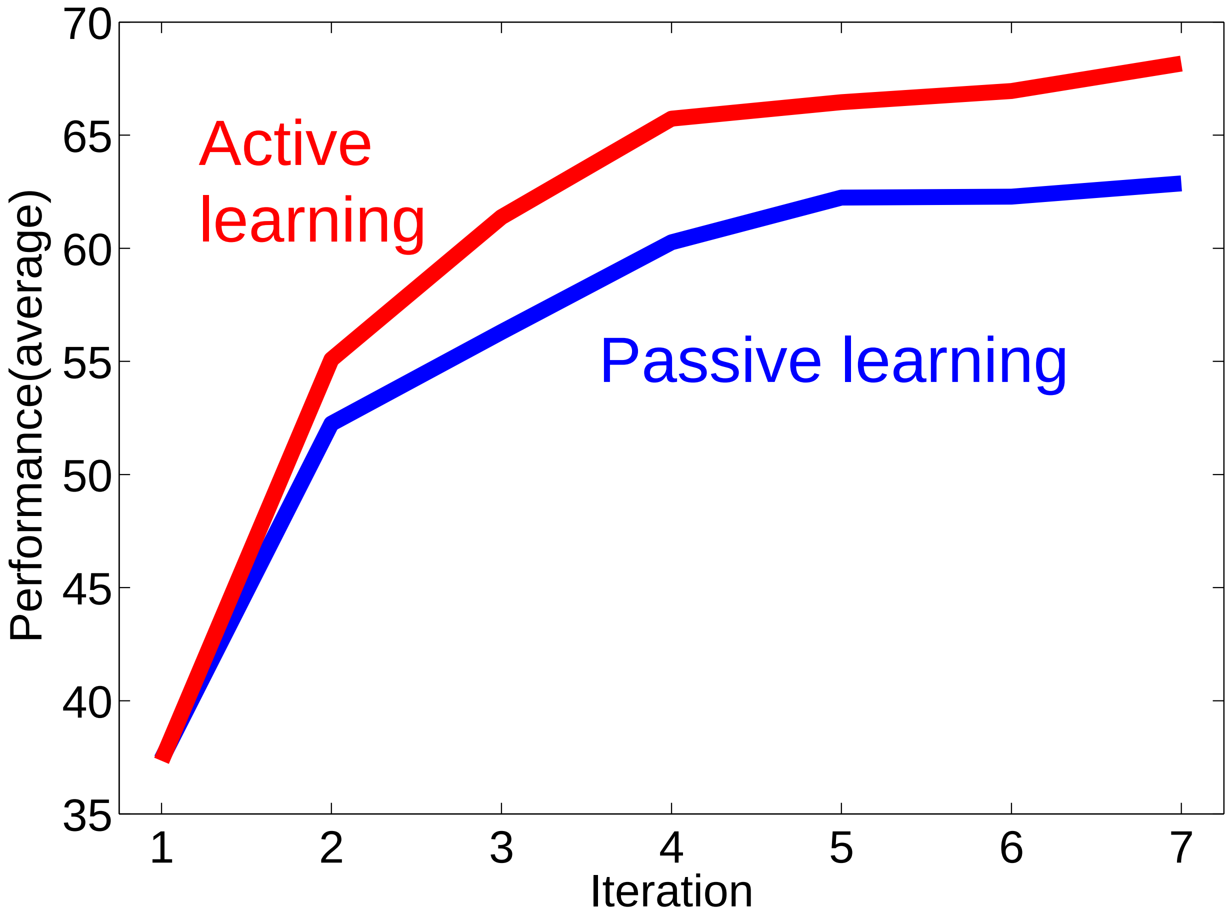
AL in Policy Iteration

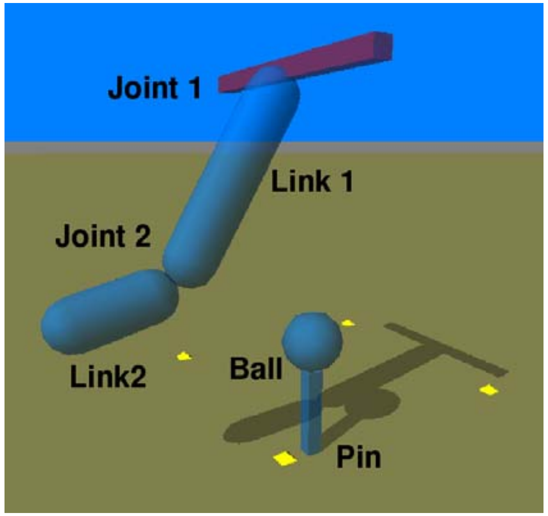
- Sampling cost is high in golf robot control (manually measuring carry distance is painful).



Akiyama, Hachiya & Sugiyama.
Active policy iteration, *IJCAI2009*.

Experimental Results

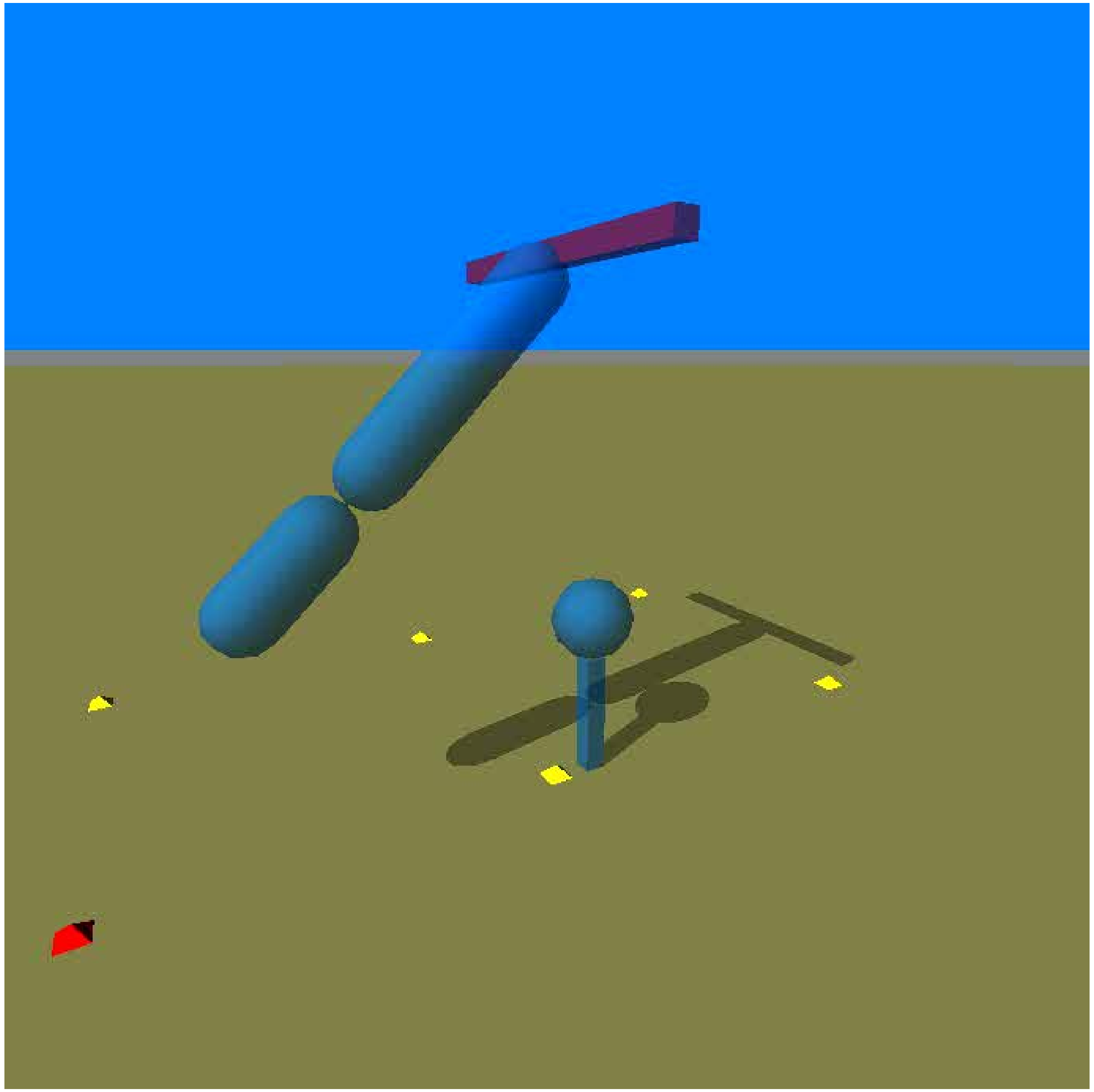




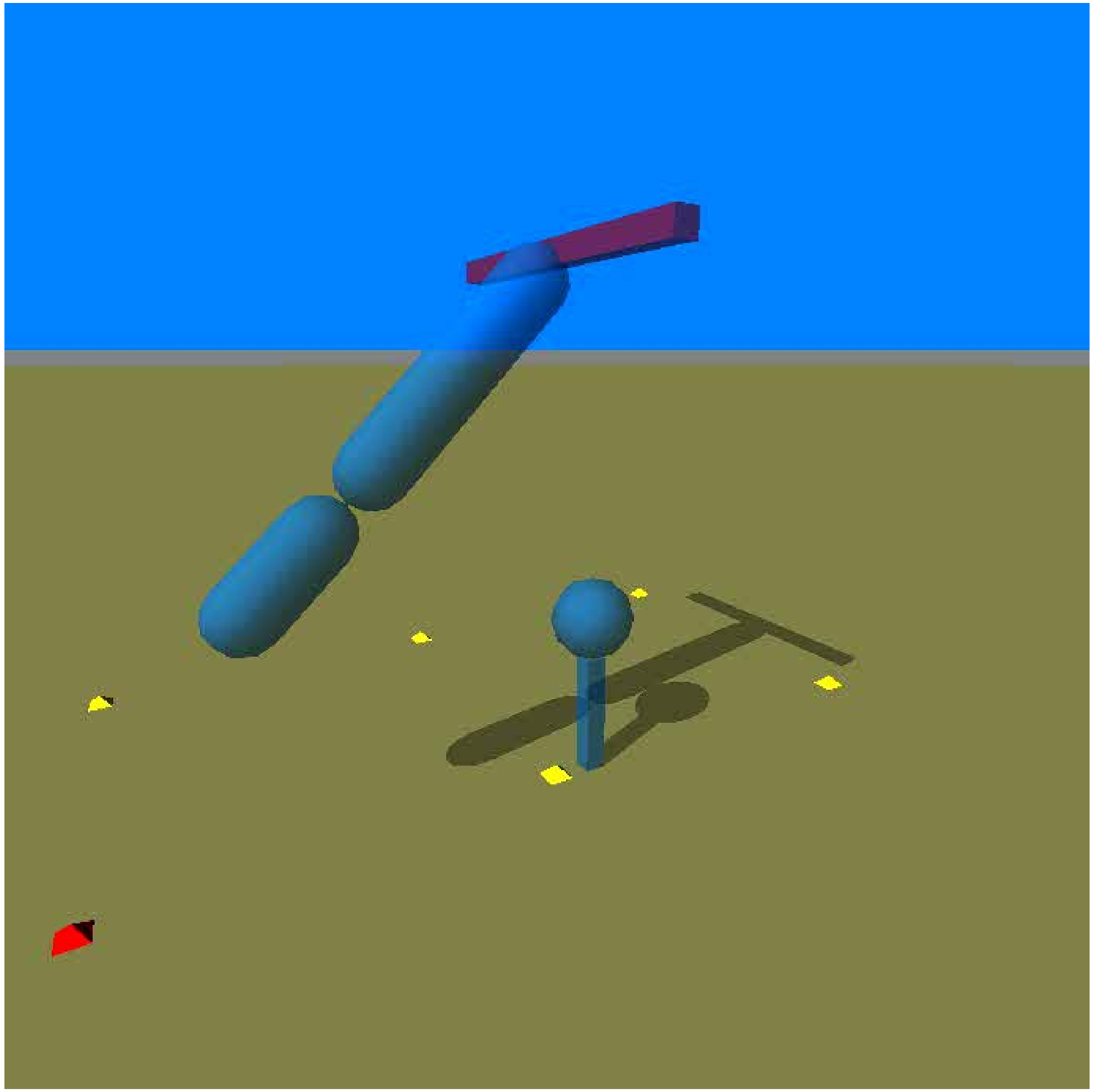
The difference of the performances at 7-th iteration is statistically significant by the t-test at the significance level 1%.

- AL improves the performance!

Passive Learning



Active Learning



WLS-based AL: Summary

$$\min_{p_{train}} \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)$$

$$\boldsymbol{U}_{i,j} = \int \varphi_i(\boldsymbol{x})\varphi_j(\boldsymbol{x})p_{test}(\boldsymbol{x})d\boldsymbol{x}$$

$$\boldsymbol{L} = (\boldsymbol{X}^\top\boldsymbol{D}\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{D}$$

$$\boldsymbol{X}_{i,j} = \varphi_j(\boldsymbol{x}_i)$$

$$\{\boldsymbol{x}_i\}_{i=1}^n \overset{i.i.d.}{\sim} p_{train}(\boldsymbol{x})$$

$$\boldsymbol{D} = \mathrm{diag}\left(\frac{p_{test}(\boldsymbol{x}_1)}{p_{train}(\boldsymbol{x}_1)}, \ldots, \frac{p_{test}(\boldsymbol{x}_n)}{p_{train}(\boldsymbol{x}_n)}\right)$$

- Pros:
  - Robust against model misspecification.
  - Easy to implement.
- Cons:
  - Test input density $p_{test}(\boldsymbol{x})$ could be unknown in practice.

Organization of My Talk

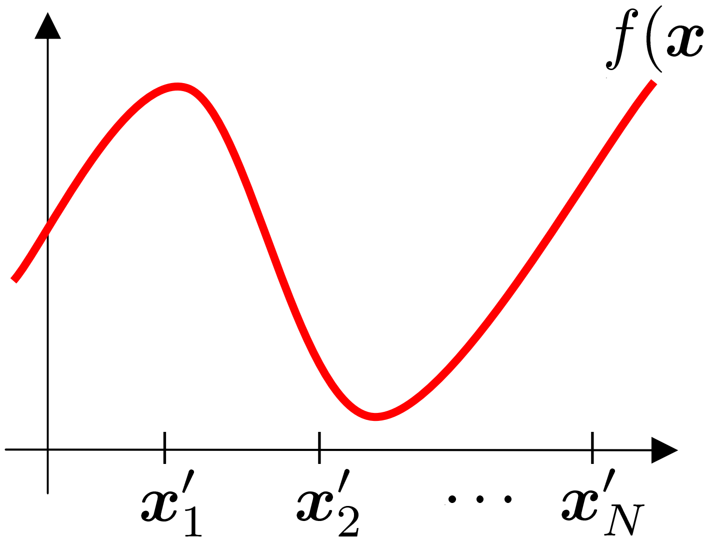
1. Formulation.
2. AL for correctly specified models.
3. AL for misspecified models.
4. Choosing inputs from unlabeled samples.
5. AL with model selection.

Pool-based AL: Setup

- Test input density $p_{test}(\boldsymbol{x})$ is unknown.
- A pool of input samples following $p_{test}(\boldsymbol{x})$ is available.

$$\{\boldsymbol{x}'_i\}_{i=1}^{N} \overset{i.i.d.}{\sim} p_{test}(\boldsymbol{x}) \qquad n \leq N$$

- From the pool, we choose sample $\{\boldsymbol{x}_i\}_{i=1}^{n}$ and gather output values $\{y_i\}_{i=1}^{n}$.



Difficulty of Pool-based AL

$$\{\boldsymbol{x}'_i\}_{i=1}^{N} \overset{i.i.d.}{\sim} p_{test}(\boldsymbol{x})$$

- $p_{test}(\boldsymbol{x})$ in $\boldsymbol{U}, \boldsymbol{D}$ are unknown, so AL criterion cannot be directly computed.

$$\min_{p_{train}} \mathrm{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)$$

$$\boldsymbol{U}_{i,j} = \int \varphi_i(\boldsymbol{x})\varphi_j(\boldsymbol{x})p_{test}(\boldsymbol{x})d\boldsymbol{x}$$

$$\boldsymbol{L} = (\boldsymbol{X}^\top \boldsymbol{D}\boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{D}$$

$$\boldsymbol{X}_{i,j} = \varphi_j(\boldsymbol{x}_i)$$

$$\{\boldsymbol{x}_i\}_{i=1}^{n} \overset{i.i.d.}{\sim} p_{train}(\boldsymbol{x})$$

$$\boldsymbol{D} = \mathrm{diag}\left(\frac{p_{test}(\boldsymbol{x}_1)}{p_{train}(\boldsymbol{x}_1)}, \ldots, \frac{p_{test}(\boldsymbol{x}_n)}{p_{train}(\boldsymbol{x}_n)}\right)$$

Naïve Approach

$$\{\boldsymbol{x}'_i\}_{i=1}^{N} \overset{i.i.d.}{\sim} p_{test}(\boldsymbol{x})$$

- Estimate test density from $\{\boldsymbol{x}_i\}_{i=1}^{N}$.
- Plug-in the estimator $\widehat{p}_{test}(\boldsymbol{x})$:

$$\boldsymbol{U}_{i,j} \approx \int \varphi_i(\boldsymbol{x})\varphi_j(\boldsymbol{x})\widehat{p}_{test}(\boldsymbol{x})d\boldsymbol{x}$$

$$\boldsymbol{D} \approx \mathrm{diag}\left(\frac{\widehat{p}_{test}(\boldsymbol{x}_1)}{p_{train}(\boldsymbol{x}_1)}, \ldots, \frac{\widehat{p}_{test}(\boldsymbol{x}_n)}{p_{train}(\boldsymbol{x}_n)}\right)$$

- However, density estimation is hard and thus this approach is not reliable.

Better Approach

- $\boldsymbol{U}$: empirical approximation

$$\widehat{\boldsymbol{U}}_{i,j} = \frac{1}{N}\sum_{i=1}^{N}\varphi_i(\boldsymbol{x}'_i)\varphi_j(\boldsymbol{x}'_i) \qquad \{\boldsymbol{x}'_i\}_{i=1}^{N} \overset{i.i.d.}{\sim} p_{test}(\boldsymbol{x})$$

- $\boldsymbol{D}$: define resampling probability over pool

$$p_{train}(\boldsymbol{x}_i) = p_{test}(\boldsymbol{x}_i) r(\boldsymbol{x}_i) \qquad \sum_{i=1}^{N} r(\boldsymbol{x}'_i) = 1, \ \ r(\boldsymbol{x}'_i) \geq 0$$

$$\Rightarrow \frac{p_{test}(\boldsymbol{x}_i)}{p_{train}(\boldsymbol{x}_i)} = \frac{1}{r(\boldsymbol{x}_i)}$$

$$\Rightarrow \boldsymbol{D} = \mathrm{diag}\left(\frac{1}{r(\boldsymbol{x}_1)}, \ldots, \frac{1}{r(\boldsymbol{x}_n)}\right)$$

This is exact!

Sugiyama & Nakajima.
Pool-based active learning in approximate linear regression.
Machine Learning, vol.75, no.3, pp.249-274, 2009.

Benchmark Datasets (8-dim)

Mean (std.) of normalized test error.

Red: Significantly better by 95% Wilcoxon test, Blue: Worth than baseline passive

Dataset	Pool / WLS-AL	Pool / OLS-AL	Population / WLS-AL	Passive
Bank-8fm	0.89(0.14)	0.91(0.14)	1.16(0.26)	1.00(0.19)
Bank-8fh	0.86(0.14)	0.85(0.14)	0.97(0.20)	1.00(0.20)
Bank-8nm	0.89(0.16)	0.91(0.18)	1.18(0.28)	1.00(0.21)
Bank-8nh	0.88(0.16)	0.87(0.16)	1.02(0.28)	1.00(0.21)
Kin-8fm	0.78(0.22)	0.87(0.22)	0.39(0.20)	1.00(0.25)
Kin-8fh	0.80(0.17)	0.85(0.17)	0.54(0.16)	1.00(0.23)
Kin-8nm	0.91(0.14)	0.92(0.14)	0.97(0.18)	1.00(0.17)
Kin-8nh	0.90(0.13)	0.90(0.13)	0.95(0.17)	1.00(0.17)
Pumadyn-8fm	0.89(0.13)	0.89(0.12)	0.93(0.16)	1.00(0.18)
Pumadyn-8fh	0.89(0.13)	0.88(0.12)	0.93(0.15)	1.00(0.17)
Pumadyn-8nm	0.91(013.)	0.92(0.13)	1.03(0.18)	1.00(0.18)
Pumadyn-8nh	0.91(013.)	0.91(0.13)	0.98(0.16)	1.00(0.17)
Average	0.87(0.16)	0.89(0.15)	0.92(0.30)	1.00(0.20)

- “Pool/WLS” is consistently better than “Passive”.
- “Pool/OLS” is still useful.
- “Population/WLS” is unstable.

Benchmark Datasets (32-dim)

Mean (std.) of normalized test error.

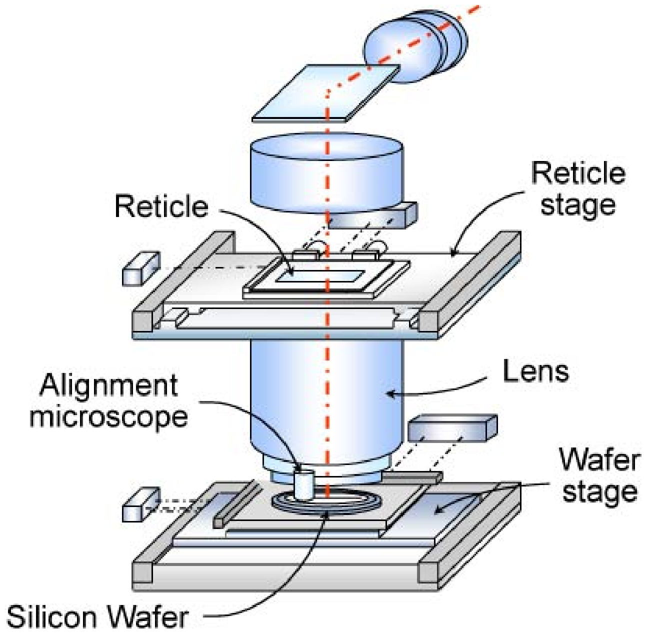
Red: Significantly better by 95% Wilcoxon test, Blue: Worth than baseline passive

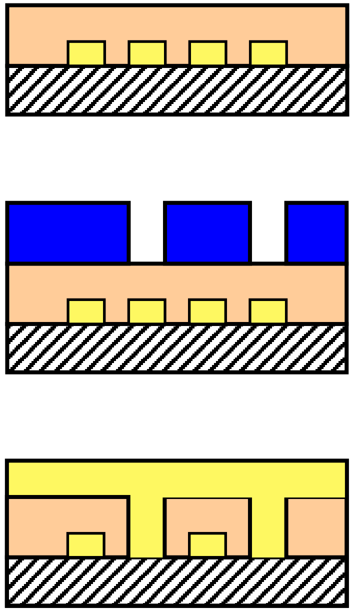
Dataset	Pool / WLS-AL	Pool / OLS-AL	Population / WLS-AL	Passive
Bank-32fm	0.97(0.05)	0.96(0.04)	1.04(0.06)	1.00(0.06)
Bank-32fh	0.98(0.05)	0.96(0.04)	1.01(0.05)	1.00(0.05)
Bank-32nm	0.98(0.06)	0.96(0.05)	1.03(0.07)	1.00(0.07)
Bank-32nh	0.97(0.05)	0.96(0.05)	0.99(0.05)	1.00(0.06)
Kin-32fm	0.79(0.07)	1.53(0.14)	0.98(0.09)	1.00(0.11)
Kin-32fh	0.79(0.07)	1.40 (0.12)	0.98(0.09)	1.00(0.10)
Kin-32nm	0.95(0.04)	0.93(0.04)	1.03(0.05)	1.00(0.05)
Kin-32nh	0.95(0.04)	0.92(0.03)	1.02(0.04)	1.00(0.05)
Pumadyn-32fm	0.98(0.12)	1.15(0.15)	0.96(0.12)	1.00(0.13)
Pumadyn-32fh	0.96(0.04)	0.95(0.04)	0.97(0.04)	1.00(0.05)
Pumadyn-32nm	0.96(0.04)	0.93(0.03)	0.96(0.03)	1.00(0.05)
Pumadyn-32nh	0.96(0.03)	0.92(0.03)	0.97(0.04)	1.00(0.04)
Average (32d)	0.94(0.09)	1.05(0.21)	1.00(0.07)	1.00(0.07)

- "Pool/WLS" is consistently better than "Passive".
- "Pool/OLS" and "population/WLS" are unstable.

Wafer Alignment in Semiconductor Exposure Apparatus

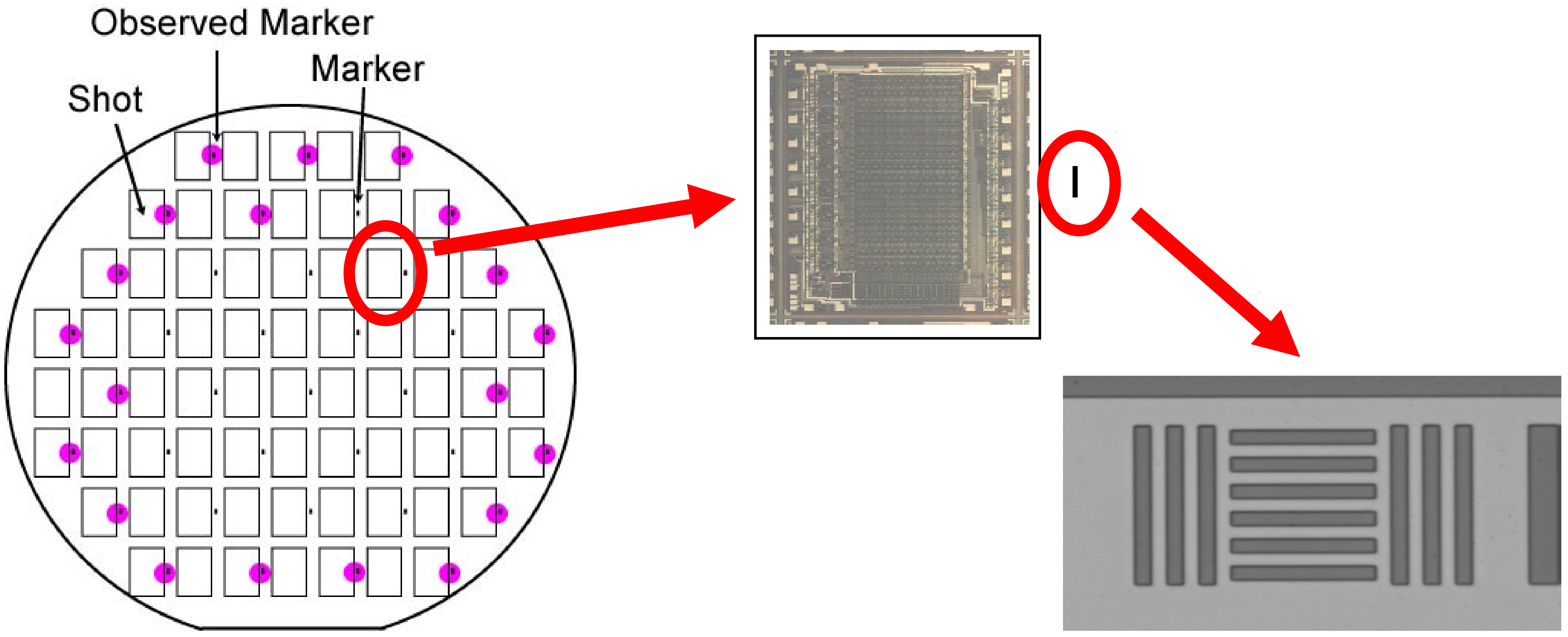
- Recent silicon wafers have layer structure.
- Circuit patterns are exposed multiple times.
- Exact alignment of wafers is necessary.





Markers on Wafer

- Wafer alignment process:
  - Measure marker location printed on wafers.
  - Shift and rotate the wafer to minimize the gap.
- For speeding up, reducing the number of markers to measure is highly important.

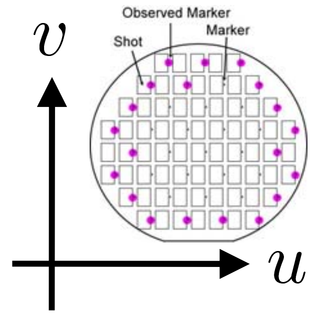


Non-linear Alignment Model

- When the gap is caused only by shift and rotation, linear model is exact:

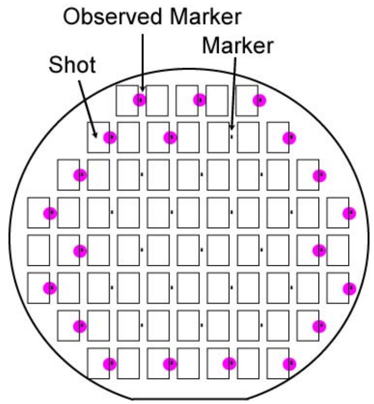
$$\Delta u \text{ or } \Delta v = \theta_0 + \theta_1 u + \theta_2 v$$

- However, non-linear factors exist, e.g.,
  - Warp
  - Biased characteristic of measurement apparatus
  - Different temperature conditions
- Exactly modeling non-linear factors is not possible in practice!



Experimental Results

- 20 markers (out of 38) are chosen by AL.
- Gaps of all markers are predicted.
- Repeated for 220 different wafers.
- Mean (standard deviation) of the gap prediction error
- Red: Significantly better by 95% Wilcoxon test
- Blue: Worse than the baseline passive method



Model	WLS-AL	OLS-AL	"Outer" heuristic AL	Passive (Random)
Order 1	2.27(1.08)	2.37(1.15)	2.36(1.15)	2.32(1.11)
Order 2	1.93(0.89)	1.96(0.91)	2.13(1.08)	2.32(1.15)

Order 1: $\Delta u \text{ or } \Delta v = \theta_0 + \theta_1 u + \theta_2 v$

Order 2: $\Delta u \text{ or } \Delta v = \theta_0 + \theta_1 u + \theta_2 v + \theta_3 uv + \theta_4 u^2 + \theta_5 v^2$

- WLS-based method works well.

Pool-based AL: Summary

$$\min_{b} \mathrm{tr}(\widehat{\boldsymbol{U}}\boldsymbol{L}\boldsymbol{L}^\top)$$

$$\widehat{U}_{i,j} = \frac{1}{N}\sum_{i=1}^{N} \varphi_i(\boldsymbol{x}'_i)\varphi_j(\boldsymbol{x}'_i)$$

$$\boldsymbol{L} = (\boldsymbol{X}^\top \boldsymbol{D}\boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{D}$$

$$\boldsymbol{X}_{i,j} = \varphi_j(\boldsymbol{x}_i)$$

$$\{\boldsymbol{x}_i\}_{i=1}^{n} \overset{i.i.d.}{\sim} \{r(\boldsymbol{x}'_i)\}_{i=1}^{N}$$

$$\boldsymbol{D} = \mathrm{diag}\left(\frac{1}{r(\boldsymbol{x}_1)}, \ldots, \frac{1}{r(\boldsymbol{x}_n)}\right)$$

- Pros:
  - Robust against model misspecification.
  - $p_{test}(\boldsymbol{x})$ can be unknown.
  - Easy to implement.
- Cons:
  - WLS has a larger variance.

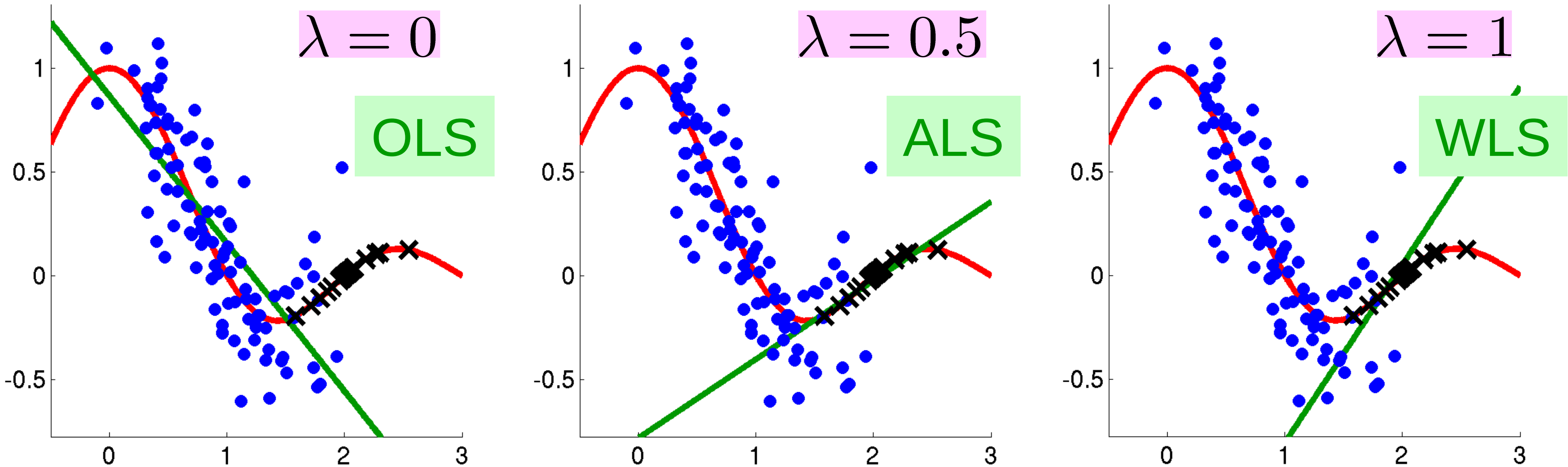
Organization of My Talk

1. Formulation.
2. AL for correctly specified models.
3. AL for misspecified models.
4. Choosing inputs from unlabeled samples.
5. AL with model selection.

Adaptive WLS (ALS)

- "flattening" importance for variance reduction.

$$\min_{\boldsymbol{\alpha}} \left[\sum_{i=1}^{n} \left(\frac{p_{test}(\boldsymbol{x}_i)}{p_{train}(\boldsymbol{x}_i)} \right)^{\lambda} \left(\widehat{f}(\boldsymbol{x}_i) - y_i \right)^2 \right]$$



Bias: Large
Variance: Small

Bias: Small
Variance: Large

Flattening Parameter Choice

- Performance of ALS depends on flattening parameter value λ
- Several model sele
  covariat

Not useful in AL

of generalization
& Decisions, vol.23, no.4,

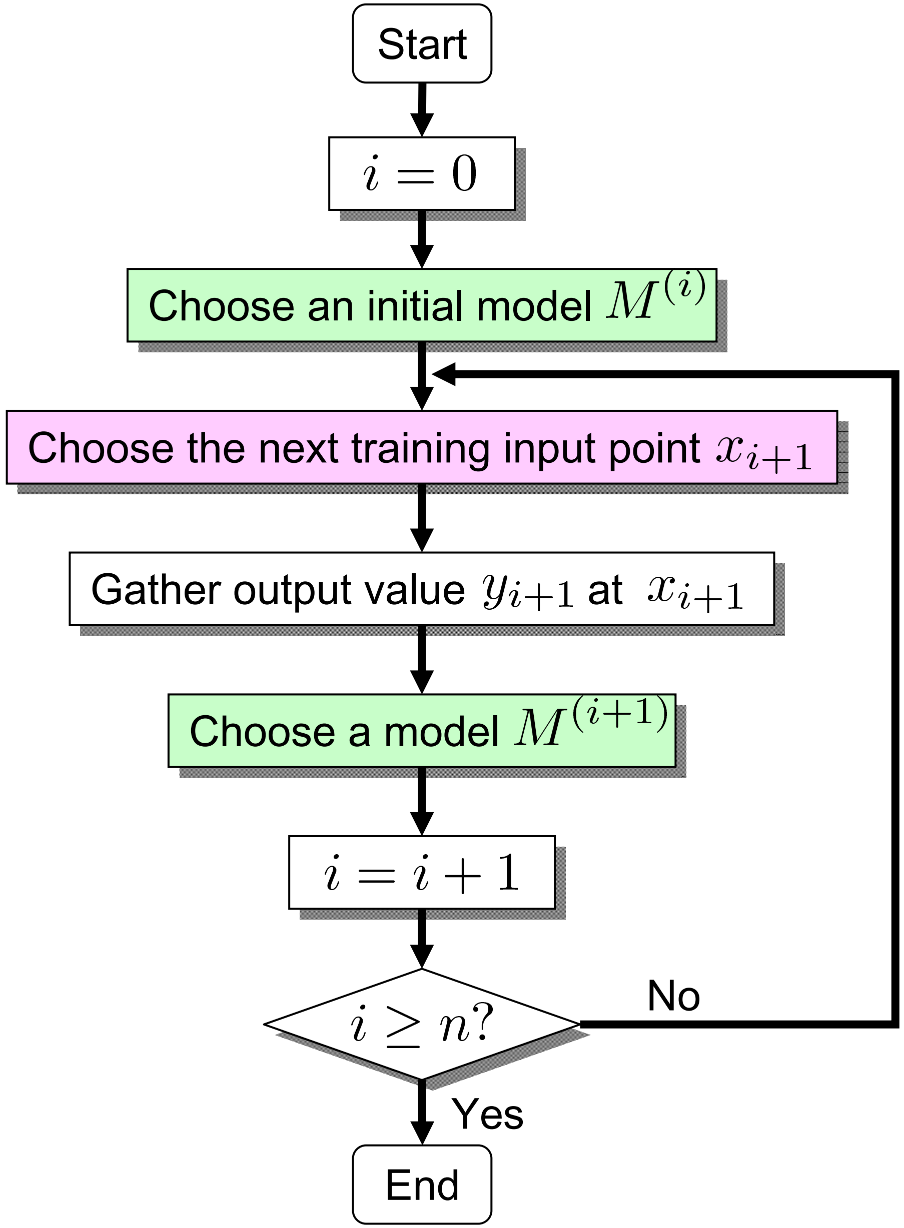
S audedat & Müller, Covariate shift adaptation by in portance weighted cross validation, *Journal of Machine Learning Research*, vol.8, pp.985-1005, 2007.

MS/AL Dilemma

- Model selection (MS):
  - Choose models using input-output training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$.
  - Thus MS is possible only after AL.
- Active learning (AL):
  - Choose input points $\{\boldsymbol{x}_i\}_{i=1}^{n}$ for a fixed model.
  - Thus AL is possible only after MS.
- MS and AL cannot be carried out by simply combining existing MS and AL methods.

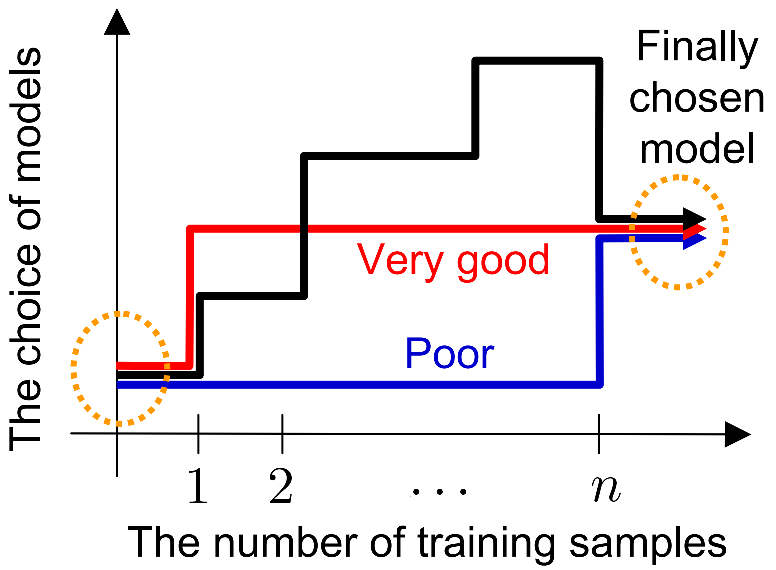
Sequential Approach

- Iteratively choose
  - a training input point (or a small portion)
  - a model
- This is commonly used in practice.



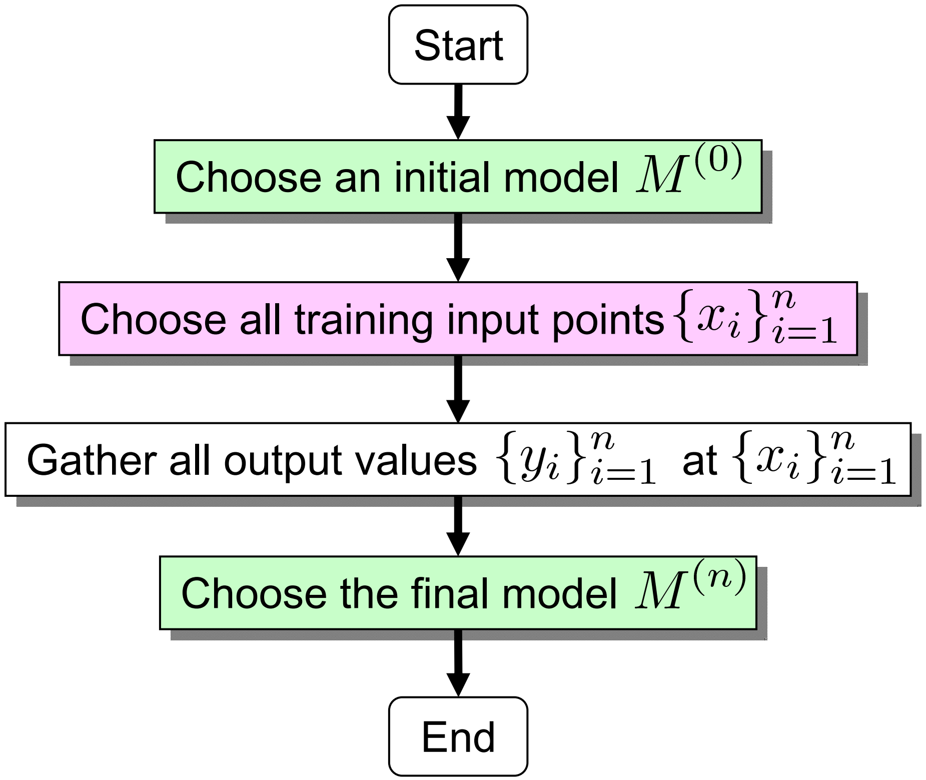
Model Drift

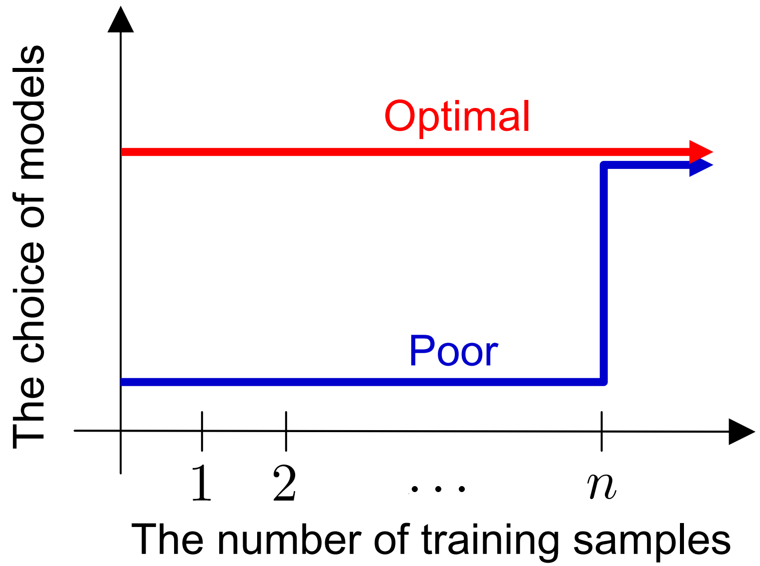
- However, sequential approach is not effective.
  - Target model varies through learning process.
  - Good training input density depends heavily on the target model.
  - Training input points determined in early stages could be poor for finally chosen model.
  - AL overfits to target models.



Batch Approach

- Perform batch AL for an initially chosen model.
- This does not suffer from model drift.





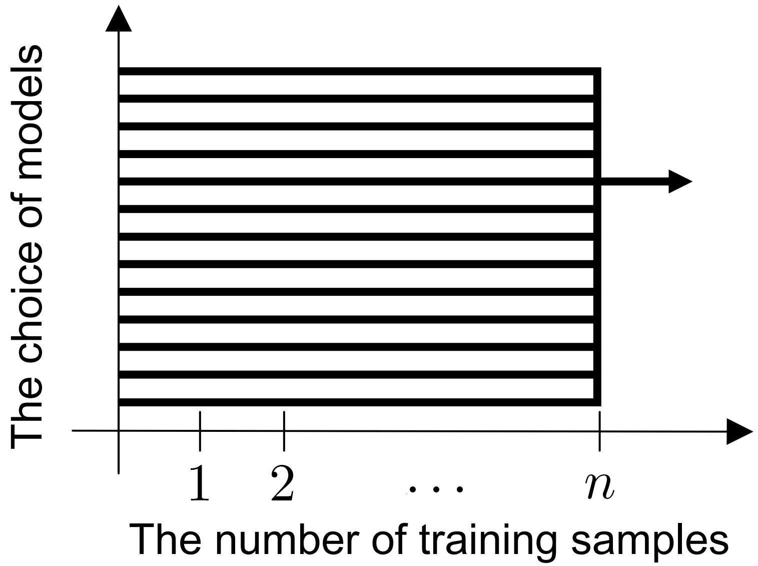
Difficulty in Initial Model Choice

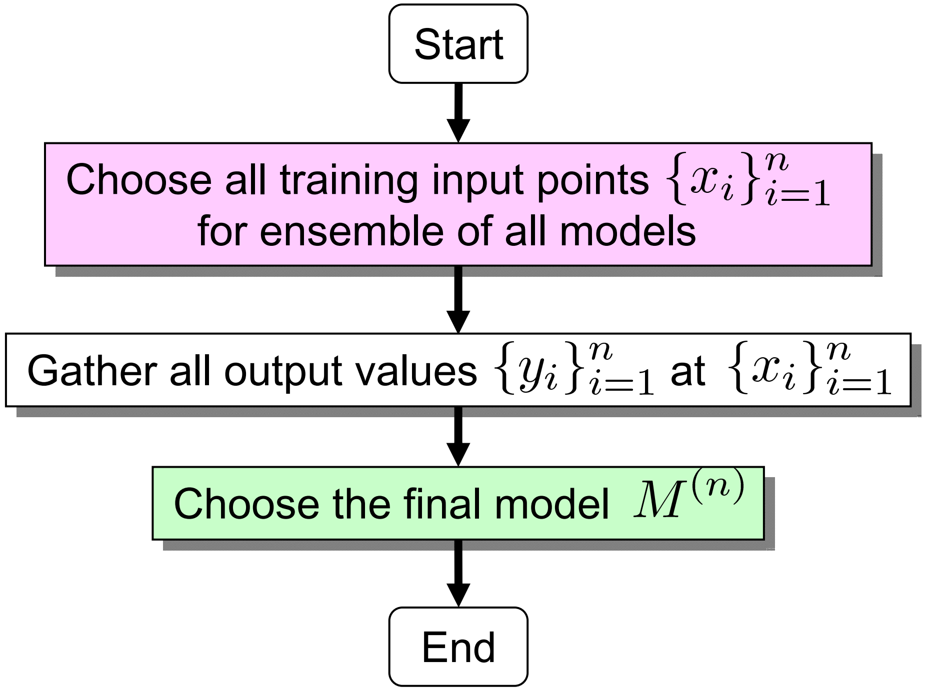
- We need to choose an initial model before observing training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$.
  - MS is not possible.
  - Variance-only AL is possible in principle, but the simplest model is always chosen.
- In practice, we may have to determine the initial model randomly.
- Therefore, batch approach is not reliable.

Ensemble Active Learning (EAL)

- Idea: perform AL for a set of model candidates

$$\min_{p_{train}} \mathbb{E}_{\mathcal{M}} \operatorname{tr}(\boldsymbol{U}\boldsymbol{L}\boldsymbol{L}^\top)$$





Sugiyama & Rubens. A batch ensemble approach to active learning with model selection. *Neural Networks*, vol.21, pp.1278-1286, 2008.

Simulation Results

Wilcoxon test (95%)

Dataset	Passive	Sequential	Batch	Ensemble
Bank-8fm	1.00(1.22)	0.59(0.85)	0.46(0.25)	0.45(0.28)
Bank-8fh	1.00(0.42)	0.53(0.22)	0.46(0.18)	0.44(0.11)
Bank-8nm	1.00(0.76)	0.63(0.19)	0.58(0.21)	0.56(0.10)
Bank-8nh	1.00(0.28)	0.61(0.19)	0.53(0.14)	0.51(0.11)
Pumadyn-8fm	1.00(0.22)	0.83(0.36)	0.92(0.68)	0.91(0.73)
Pumadyn-8fh	1.00(0.17)	0.80(0.17)	0.76(0.22)	0.71(0.19)
Pumadyn-8nm	1.00(0.18)	0.86(0.15)	0.85(0.20)	0.81(0.18)
Pumadyn-8nh	1.00(0.19)	0.85(0.14)	0.81(0.17)	0.77(0.15)

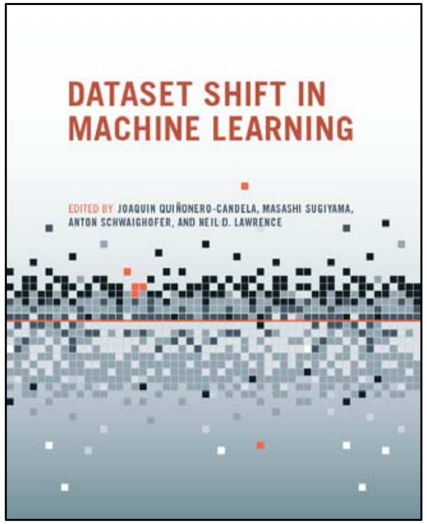
- All methods outperform passive.
- Ensemble method works the best!

Conclusions

- Active learning (AL) is useful when sampling cost is high.
- OLS-AL: good for correct models.
- WLS-AL: good for misspecified models.
- Pool-based AL: unlabeled samples are utilized.
- Ensemble AL: also choosing models.

Books

- Quiñonero-Candela, Sugiyama, Schwaighofer & Lawrence (Eds.), Dataset Shift in Machine Learning, MIT Press, 2009.



- Sugiyama, von Bünau, Kawanabe & Müller, Covariate Shift Adaptation in Machine Learning, MIT Press (in preparation)
- Sugiyama, Suzuki & Kanamori, Density Ratio Estimation in Machine Learning, Cambridge University Press (in preparation)