# Learning Processes

Dr. Ratna Babu Chinnam

Industrial & Systems Engineering

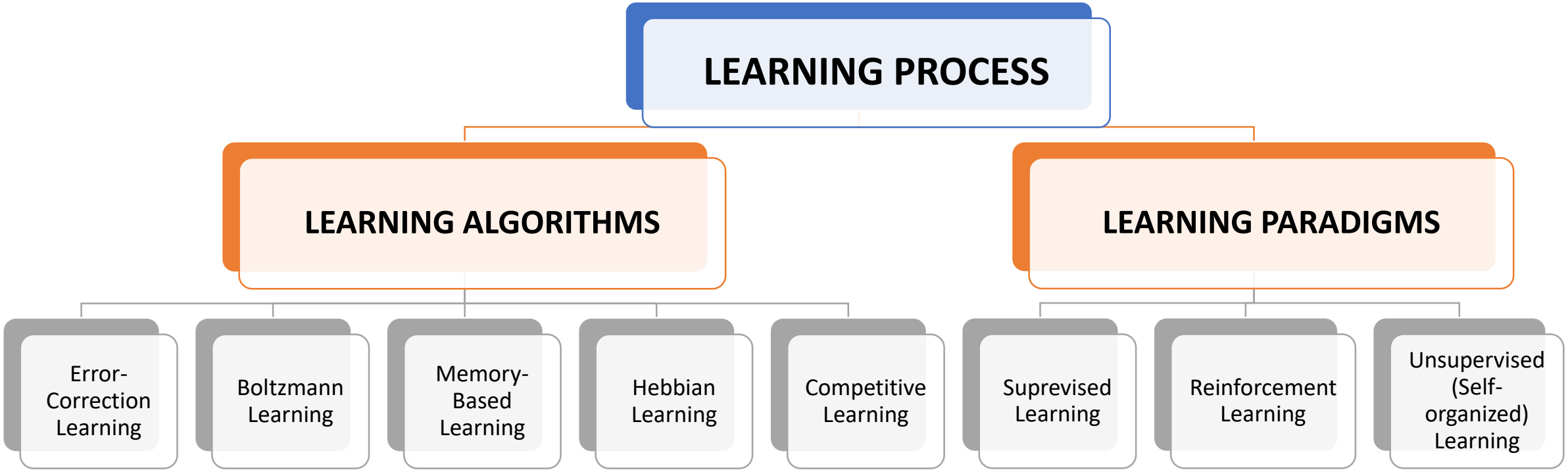
Wayne State University

Primary Source: Simon Haykin (2009)

# Learning Processes

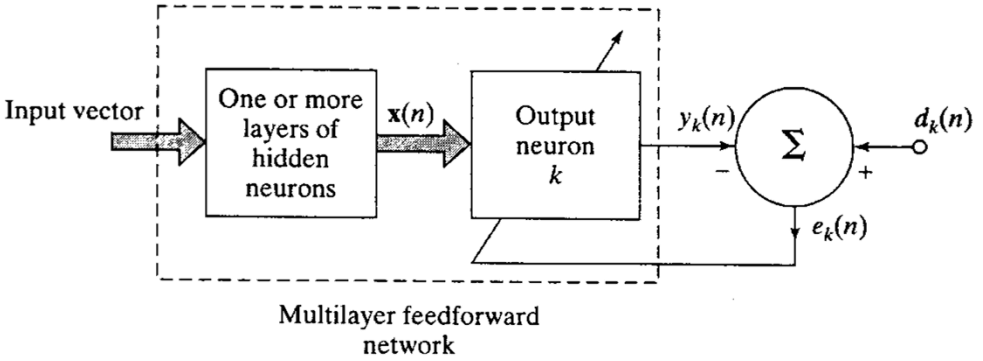
- **Learning** is the *process* by which the free *parameters* (weights) of a neural network are *adapted* through a continuing process of *stimulation* by the *environment* in which the network is embedded.
- A prescribed set of well-defined *rules for the solution of a learning problem* is called a **learning algorithm**.
- Learning algorithms differ in the way neuron synaptic weight adjustments are formulated.
- **Learning paradigm** refers to a *model of the environment* in which the neural network operates.

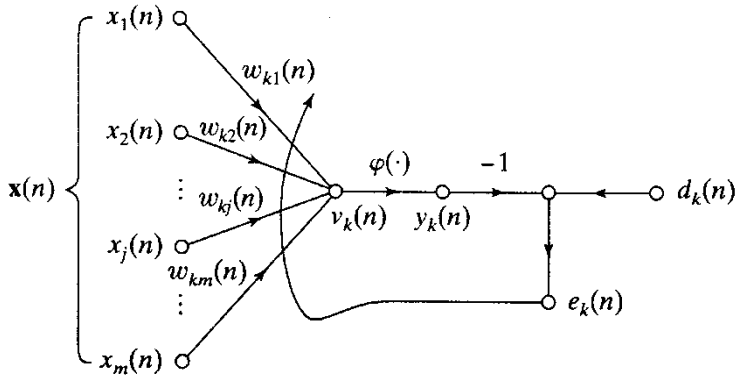
# Learning Processes



# Error-Correction Learning

- **Setting**: Consider an output neuron $k$ in a FFN
- Let $x(n)$, $y_k(n)$, and $d_k(n)$ denote the input vector, output, and desired target for neuron $k$
- $n$ denotes the iteration of learning process
- **Objective**: Minimize cost function, $J$, defined in terms of error $e_k(n) = d_k(n) - y_k(n)$
- One popular cost function is: $J = E\left[{}^1/_2 \sum_k e_k^2(n)\right]$
- **Learning**: *Gradient descent* is a possible approach to minimizing $J$
- Minimizing $J$ requires knowledge of statistical properties of the underlying process (difficult)
- Several alternatives; Use *instantaneous value* of sum of squared error: $\varepsilon(n) = {}^1/_2 \sum_k e_k^2(n)$

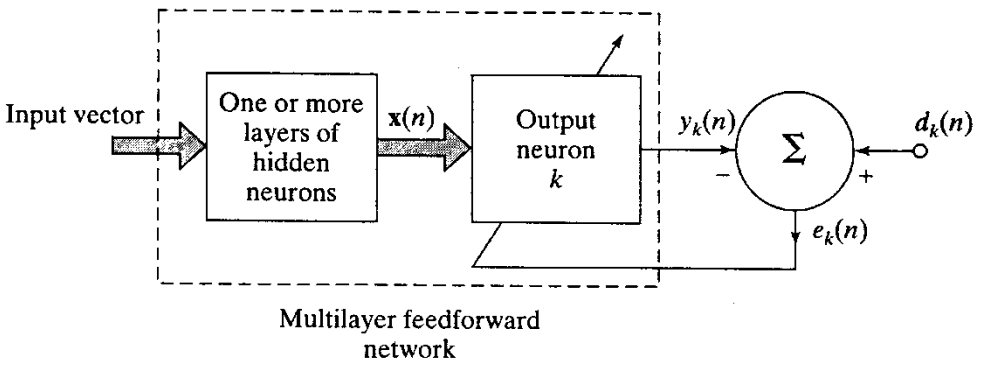


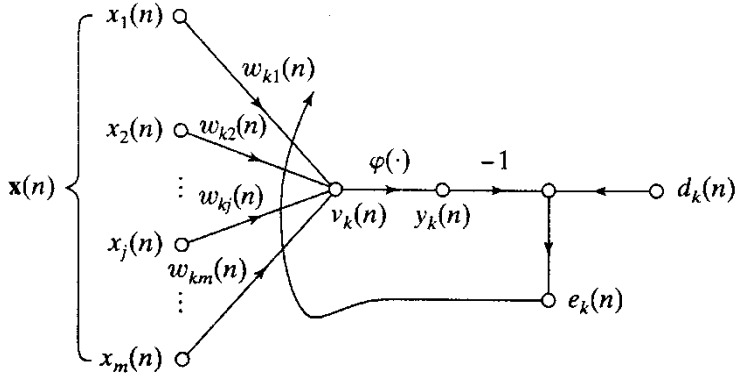



# Error-Correction Learning …

- $\varepsilon(n)$ can be minimized using the delta rule or Widrow-Hoff rule: $\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$
- Rule presumes error signal is measurable (neuron $k$ is visible to outside world)
- Updated weight: $w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$
- Learning rate $\eta$ has to be selected carefully to ensure stability or convergence
- **Notes**:
  - Cost function can be convex with a unique optimal solution or non-convex with local optimal solutions
  - Delta rule is essentially a gradient-descent method; Proof?
  - All gradient-descent methods lead to local optimal solutions
  - Gradient-descent techniques primarily update parameters in the following manner:

$$w_{kj}(n+1) = w_{kj}(n) - \eta \frac{\partial \varepsilon(n)}{\partial w_{kj}(n)}$$



# Memory-Based Learning

- All (or most) of the *past experiences are "explicitly" stored* in memory of correctly classified (or learnt) input-output examples: $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$
- When classifying a test vector $\mathbf{x}_{\text{test}}$ (not seen before), algorithm responds by retrieving and analyzing training data in the "local neighborhood" of $\mathbf{x}_{\text{test}}$
- *Memory-based learning* can be thought of as a *local-approximation learning* while *error-correction learning* can be thought of as a *global-approximation learning*
- All memory-based learning algorithms involve two essential ingredients:
  1. Criterion used for defining the local neighborhood of test vector $\mathbf{x}_{\text{test}}$
  2. Learning rule applied to training examples in local neighborhood of $\mathbf{x}_{\text{test}}$
- Example: $k$-Nearest Neighbor Algorithms

# Hebbian Learning

- **Hebb's Postulate of Learning** [Hebb, 1949]: "When an axon on a cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased."
- **Key properties**: Time-dependent mechanism, Local mechanism, Interactive mechanism, and Conjunctional or correlational mechanism.
- **Mathematical Models of Hebbian Modifications:**
  - Consider a synaptic weight $w_{kj}$ of neuron $k$ with presynaptic and postsynaptic signals denoted by $x_j$ and $y_k$, respectively.
  - Adjustment to the synaptic weight at time step $n$ can be expressed in the general form: $\Delta w_{kj}(n) = \eta F(y_k(n), x_j(n))$
- ***Activity Product Rule***: $\Delta w_{kj}(n) = \eta y_k(n) x_j(n)$
  - Repeated application can lead to exponential growth of $w_{kj}$ (saturation)
- ***Covariance Hypothesis***: $\Delta w_{kj}(n) = \eta(y_k(n) - \bar{y}_k)(x_j(n) - \bar{x}_j)$

# Competitive Learning

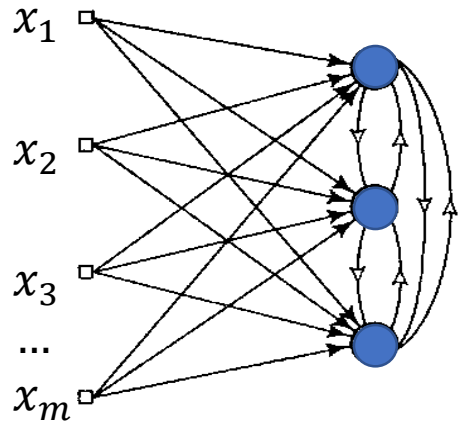
- **Setting**: Output neurons compete with one winner
- **Three Basic Elements** [Rumelhart and Zipser, 1985]:
  1. Neurons are identical except for some random synaptic weights
  2. A limit imposed on the strength of each neuron
  3. A mechanism for the competition
- **Winner**: Neuron $k$ with largest induced field $v_k$ for input $x$
- Output $y_k$ of winner is set to one; others to zero:

$$y_k = \begin{cases} 1 & \text{if } v_k > v_j \forall j, j \neq k \\ 0 & \text{otherwise} \end{cases}$$

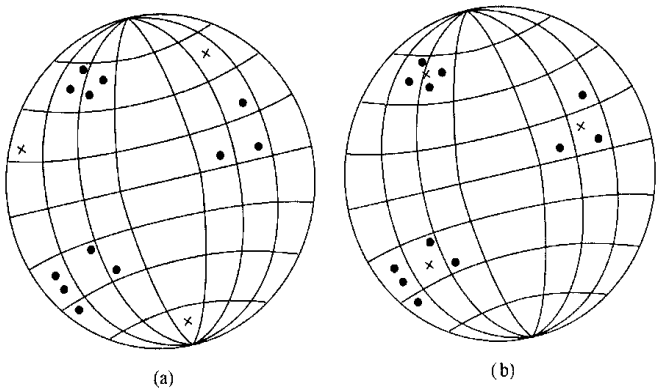
- Normally, neurons get fixed synaptic weight: $\sum_j w_{kj} = 1 \; \forall k$
- **Learning**: According to standard competitive learning rule

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_{kj}) & \text{if neuron } k \text{ wins competition} \\ 0 & \text{if neuron } k \text{ loses competition} \end{cases}$$

- Self-Organizing-Maps employ competitive learning for clustering




A simple competitive learning network with three output neurons



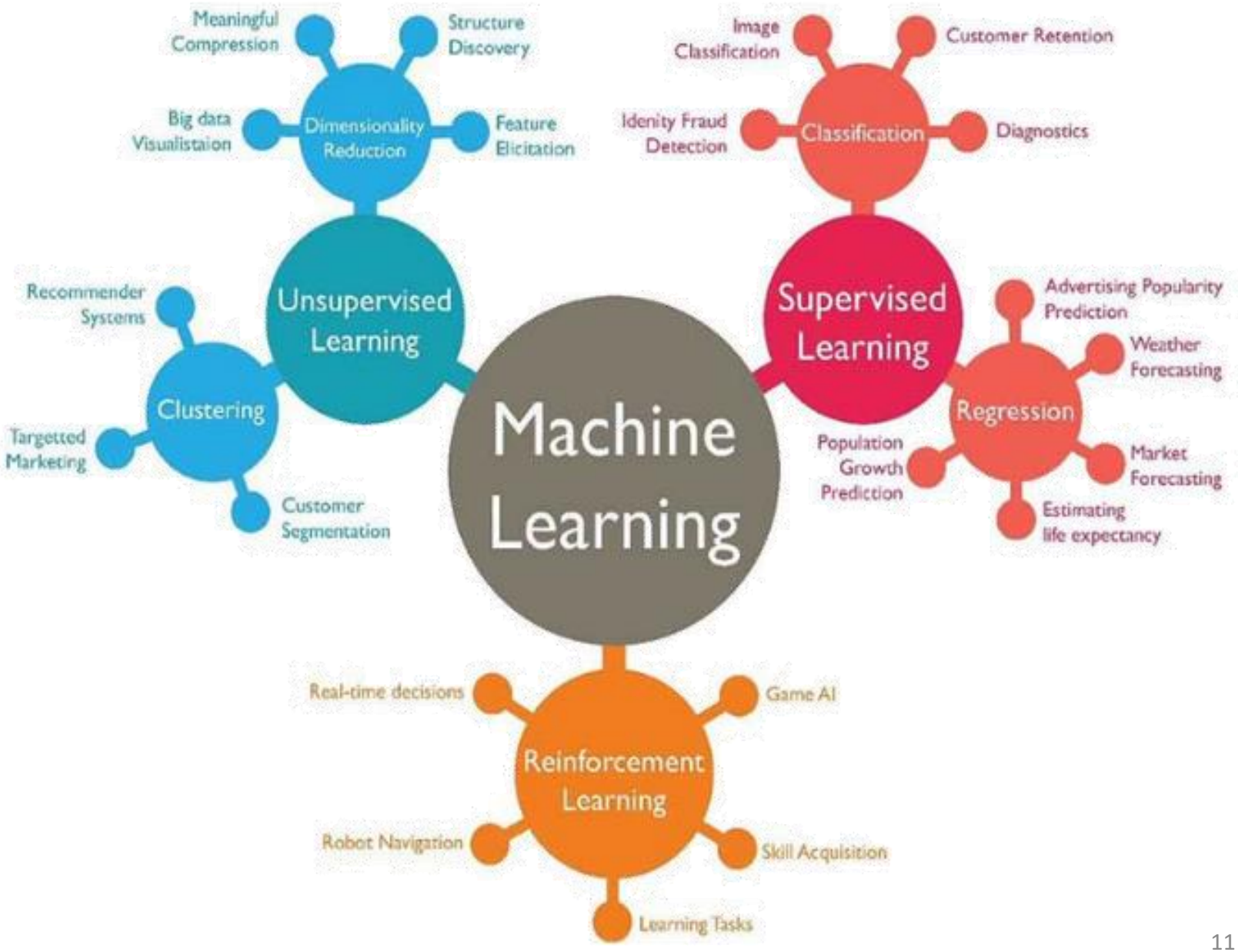

Geometric interpretation of the competitive learning process. (a) Initial state. (b) Final state.

# Credit-Assignment Problem

- Is the problem of *assigning credit or blame* for overall outcomes to each of the *internal decisions* made by a learning machine
- In many cases, this problem can be decomposed into two sub-problems [Sutton, 1984]:
  1. *Assignment of credit for outcomes to actions*. **Temporal credit-assignment problem** in that it involves the instants of time when the actions that deserve credit were actually taken.
  2. *Assignment of credit for actions to internal decisions*. **Structural credit-assignment problem** in that it involves assigning credit to internal structures of actions generated by the system.
- Credit-assignment problem faces any distributed learning machine that attempts to improve its performance in situations involving temporally extended behavior [Williams, 1988]

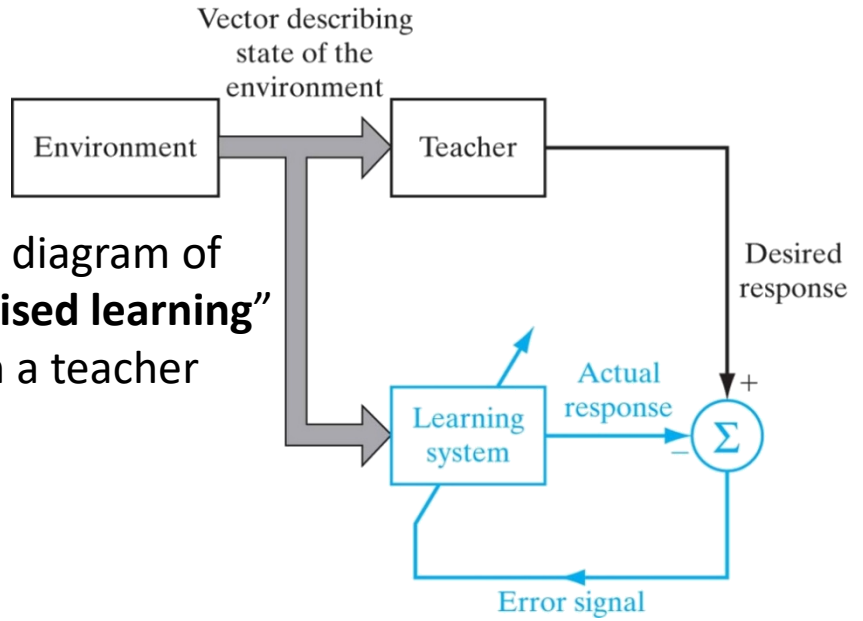
# Learning Paradigms

- Learning With A Teacher: ***Supervised Learning***
- Learning Without A Teacher: ***Unsupervised Learning***
- Learning Under Delayed Feedback: ***Reinforcement Learning***

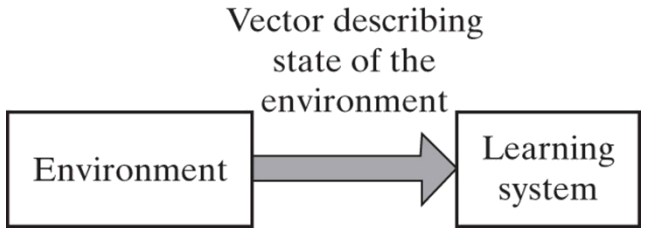


# Learning Paradigms

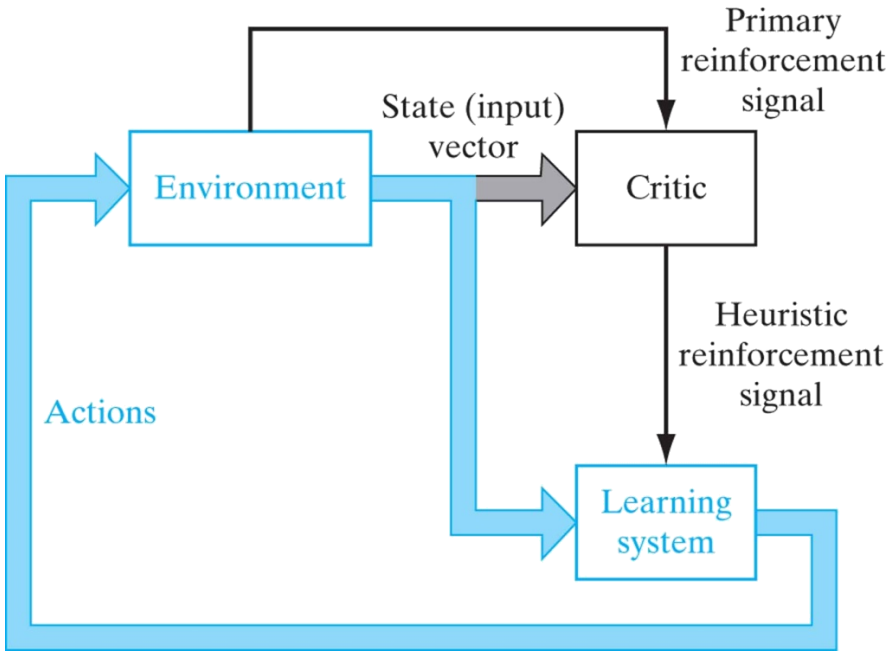
- Learning With A Teacher: ***Supervised Learning***
- Learning Without A Teacher: ***Unsupervised Learning***
- Learning Under Delayed Feedback: ***Reinforcement Learning***




Block diagram of "**supervised learning**" with a teacher




Block diagram of "**unsupervised learning**"




Block diagram of one form of "**reinforcement learning**"

# Learning Tasks

- Choice of learning algorithm depends on learning task
- **Pattern Association**:
  - Autoassociation (unsupervised)
  - Heteroassociation (supervised)
- **Pattern Recognition** (Classification): Received pattern is assigned to one of a prescribed number of classes (supervised)
- **Function Approximation**: Learn an unknown input output mapping (linear/nonlinear) with or without noise (supervised)
- **Forecasting**: Train network to forecast the future (supervised)
- **Control**: Goal is to develop a model for controlling systems using a control action in an optimum manner
- **Filtering**: Establish a "best estimate" for the true value of some system from an incomplete, potentially noisy set of observations

# Statistical Nature of Learning (Context: Regression)

- **Focus**: Final deviation between a "true" function $f(\mathbf{x})$ and "learnt" function $F(\mathbf{x}, \mathbf{w})$
- **Data**: Suppose there is a finite dataset: $\Im = \{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$
- **"True" but Unknown Model**: $D = f(\mathbf{X}) + \varepsilon$

  $\varepsilon$ - random error: 1) $E(\varepsilon|\mathbf{x}) = 0$; 2) $E[\varepsilon f(\mathbf{X})] = 0$
- **"Learnt" Model**: $F(\mathbf{x}, \mathbf{w})$ obtained by minimizing loss function

$$\xi(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}(d_i - F(\mathbf{x}_i, \mathbf{w}))^2 \text{ or } \xi(\mathbf{w}) = \frac{1}{2}E_{\Im}[(d - F(\mathbf{x}, \mathbf{w}))^2]$$

$E_{\Im}$ denotes *average operator* taken over $\Im$

# Statistical Nature of Learning (Context: Regression) …

One can rearrange terms of the model loss function:

$$\xi(\mathbf{w}) = \frac{1}{2} E_{\Im}[(d - F(\mathbf{x}, \mathbf{w}))^2]$$

$$= \frac{1}{2} E_{\Im}[\{(d - f(\mathbf{x})) + (f(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))\}^2]$$

$$= \frac{1}{2} E_{\Im}[\{\varepsilon + (f(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))\}^2]$$

$$= \frac{1}{2} E_{\Im}[\varepsilon^2] + \frac{1}{2} E_{\Im}[(f(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))^2] + E_{\Im}[\cancel{\varepsilon(f(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))}]$$

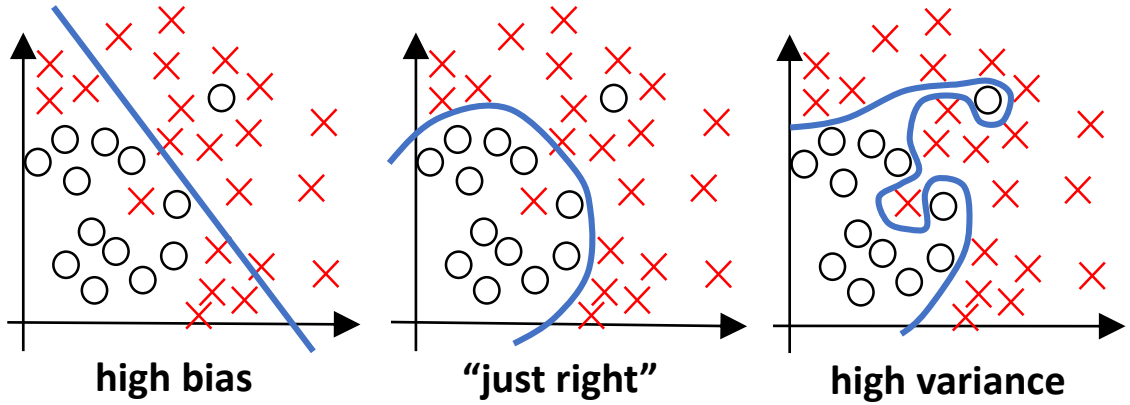
$$= \underbrace{\frac{1}{2} E_{\Im}[\varepsilon^2]}_{\text{Intrinsic Error}} + \underbrace{\frac{1}{2} E_{\Im}[(f(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))^2]}_{\text{Natural Measure of Effectivness}}$$

**Cannot be minimized**

**Can be minimized by learning algorithm**

# Statistical Nature of Learning (Context: Regression) …

**Bias-Variance Dilemma:**




$$E_{\mathfrak{J}}[(f(\mathbf{x}) - F(\mathbf{x},\mathbf{w}))^2]$$

$$= E_{\mathfrak{J}}[(E[D|\mathbf{X}=\mathbf{x}] - F(\mathbf{x},\mathbf{w}))^2]$$

$$= E_{\mathfrak{J}}[\{(E[D|\mathbf{X}=\mathbf{x}] - E_{\mathfrak{J}}[F(\mathbf{x},\mathbf{w})]) + (E_{\mathfrak{J}}[F(\mathbf{x},\mathbf{w})] - F(\mathbf{x},\mathbf{w}))\}^2]$$

$$= E_{\mathfrak{J}}[(E[D|\mathbf{X}=\mathbf{x}] - E_{\mathfrak{J}}[F(\mathbf{x},\mathbf{w})])^2] + E_{\mathfrak{J}}[(E_{\mathfrak{J}}[F(\mathbf{x},\mathbf{w})] - F(\mathbf{x},\mathbf{w}))^2] +$$

$$2E_{\mathfrak{J}}[(E[D|\mathbf{X}=\mathbf{x}] - E_{\mathfrak{J}}[F(\mathbf{x},\mathbf{w})])(E_{\mathfrak{J}}[F(\mathbf{x},\mathbf{w})] - F(\mathbf{x},\mathbf{w}))]$$

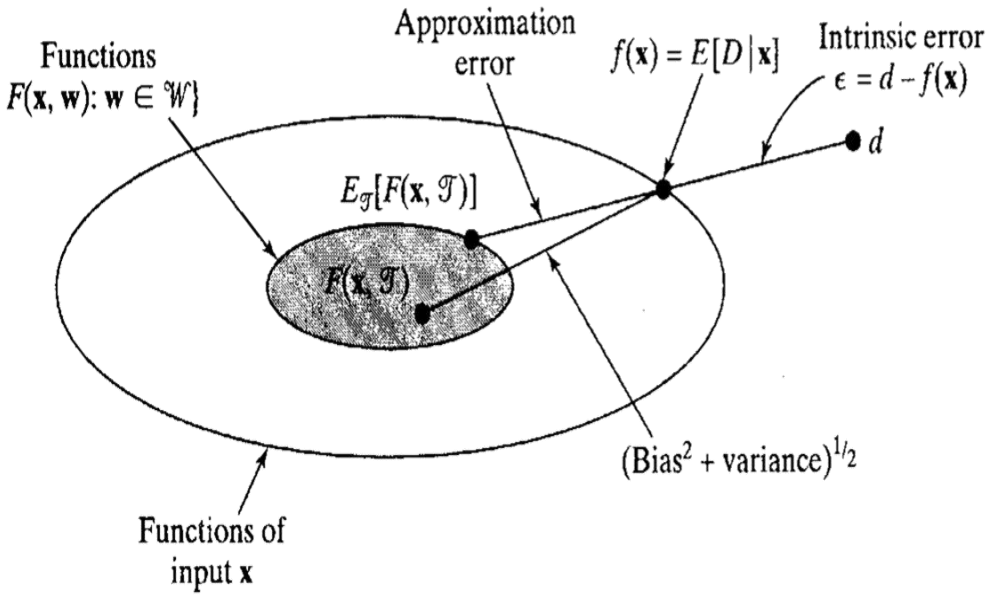
$$= \underbrace{(E[D|\mathbf{X}=\mathbf{x}] - E_{\mathfrak{J}}[F(\mathbf{x},\mathbf{w})])^2}_{Bias^2} + \underbrace{E_{\mathfrak{J}}[(E_{\mathfrak{J}}[F(\mathbf{x},\mathbf{w})] - F(\mathbf{x},\mathbf{w}))^2]}_{Variance}$$

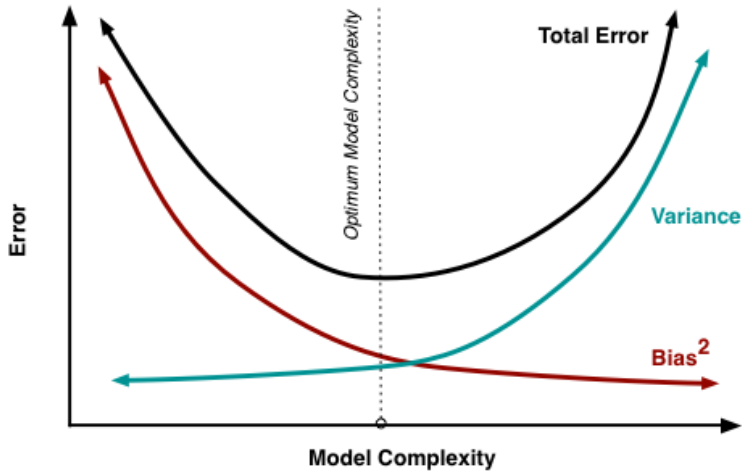
**Approximation Error** **Estimation Error**

# Statistical Nature of Learning (Context: Regression) …

**Bias-Variance Dilemma:**

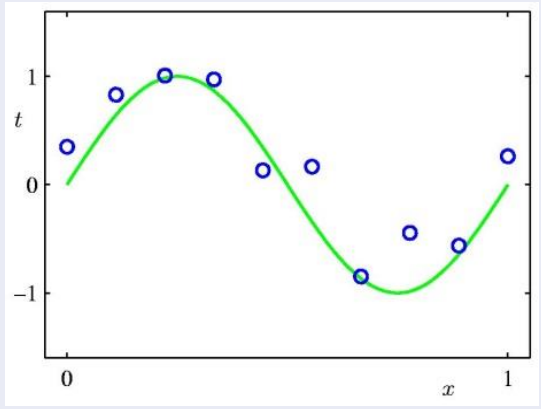
- Bias and variance should be balanced
- Under small datasets, price for achieving a small bias is large variance
  - *Larger neural networks (more parameters) produce smaller bias but larger variance*
  - *Smaller neural networks (fewer parameters) produce larger bias but smaller variance*
- Only when training sample becomes (infinitely) large can we completely eliminate both bias and variance



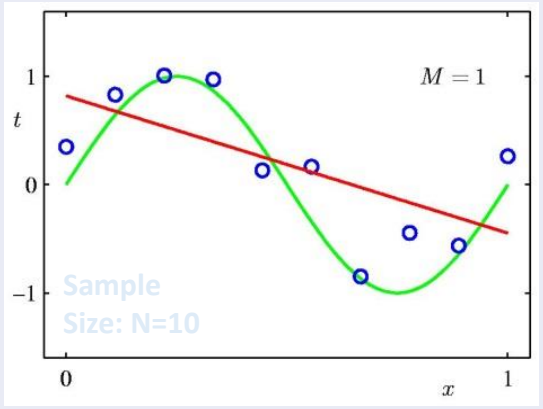



# Bias-Variance Dilemma: Polynomial Curve Fitting

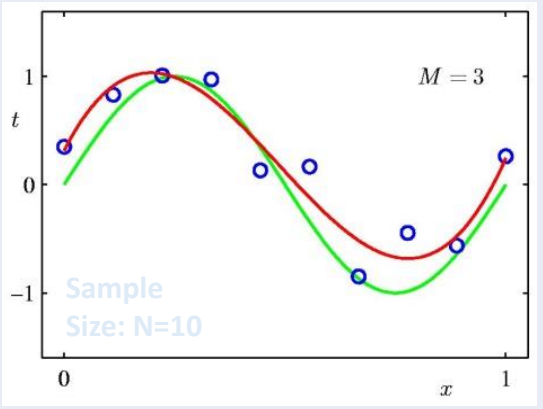
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$



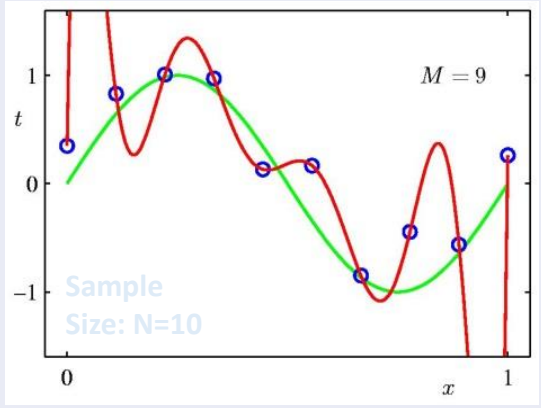
True Function (Green); Observations (Blue)



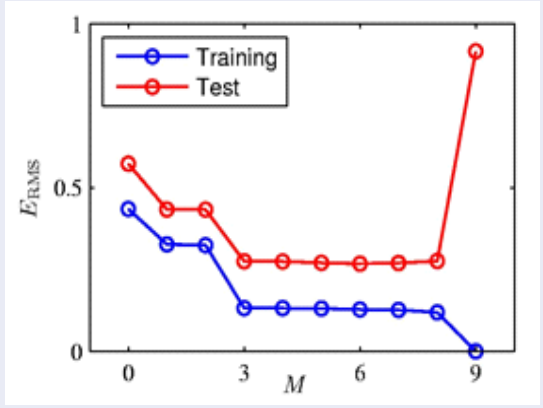

1st Order Polynomial



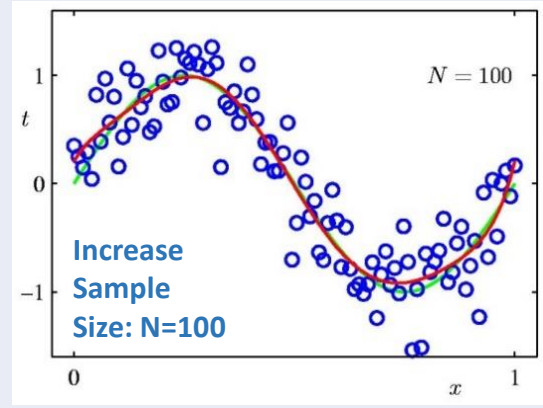

3rd Order Polynomial




9th Order Polynomial




Over-Fitting




9th Order Polynomial

| | $M=0$ | $M=1$ | $M=3$ | $M=9$ |
|---|---|---|---|---|
| $w_0^\star$ | 0.19 | 0.82 | 0.31 | 0.35 |
| $w_1^\star$ | | -1.27 | 7.99 | 232.37 |
| $w_2^\star$ | | | -25.43 | -5321.83 |
| $w_3^\star$ | | | 17.37 | 48568.31 |
| $w_4^\star$ | | | | -231639.30 |
| $w_5^\star$ | | | | 640042.26 |
| $w_6^\star$ | | | | -1061800.52 |
| $w_7^\star$ | | | | 1042400.18 |
| $w_8^\star$ | | | | -557682.99 |
| $w_9^\star$ | | | | 125201.43 |

Polynomial Model Coefficients

**How to fix this problem of overfitting?**

(Source: C. Bishop)

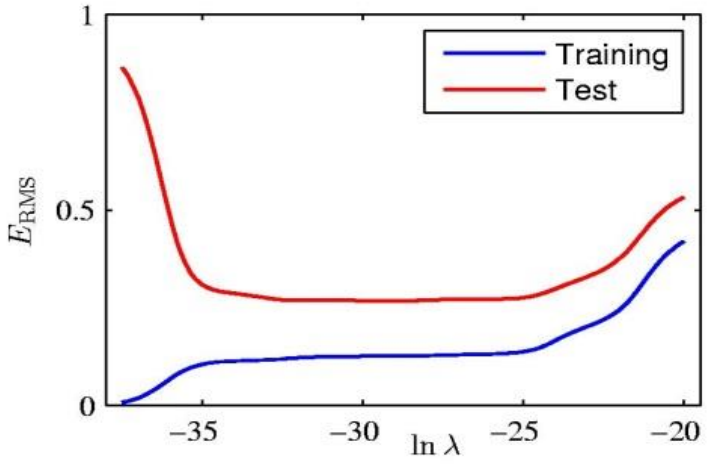
# Bias-Variance Dilemma: Polynomial Curve Fitting …

"**Regularization**" to Deal with Bias/Variance Dilemma: Penalize Large Coefficient Values

**Ridge Regression**

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

$l_1$ Norm penalty leads to LASSO Regression




| | $\ln\lambda = -\infty$ | $\ln\lambda = -18$ | $\ln\lambda = 0$ |
|---|---|---|---|
| $w_0^\star$ | 0.35 | 0.35 | 0.13 |
| $w_1^\star$ | 232.37 | 4.74 | -0.05 |
| $w_2^\star$ | -5321.83 | -0.77 | -0.06 |
| $w_3^\star$ | 48568.31 | -31.97 | -0.05 |
| $w_4^\star$ | -231639.30 | -3.89 | -0.03 |
| $w_5^\star$ | 640042.26 | 55.28 | -0.02 |
| $w_6^\star$ | -1061800.52 | 41.32 | -0.01 |
| $w_7^\star$ | 1042400.18 | -45.95 | -0.00 |
| $w_8^\star$ | -557682.99 | -91.53 | 0.00 |
| $w_9^\star$ | 125201.43 | 72.68 | 0.01 |

Polynomial Coefficients under Regularization

**No Free Lunch:** Regularization Distorts the Loss Function (introduces some bias)!

(Source: C. Bishop)

# Linear Regression: Regularization Options & Tradeoffs

- **Three Norms:** $l_0, l_1, l_2$

$$\|\beta\|_0 = \sum_{j=1}^{p} 1\{\beta_j \neq 0\}, \quad \|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|, \quad \|\beta\|_2 = \left( \sum_{j=1}^{p} \beta_j^2 \right)^{1/2}.$$

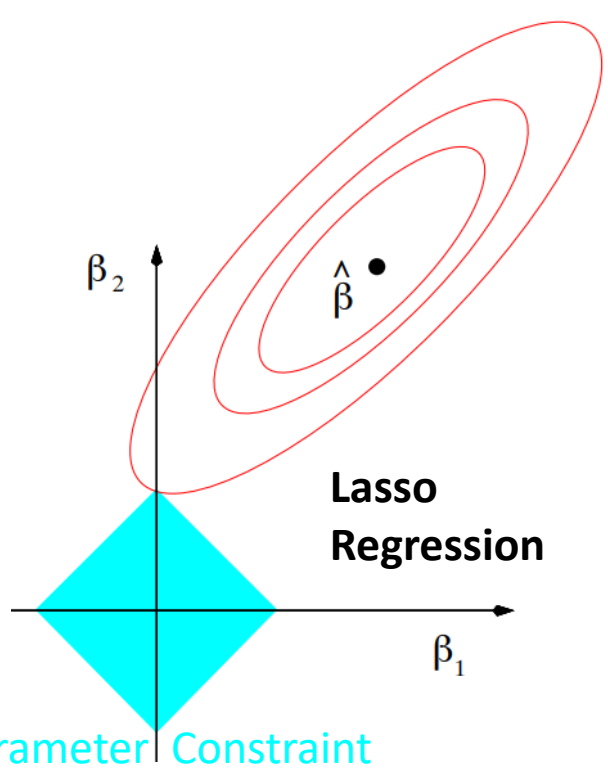
- **In constrained form:**
  - **Best Subset Selection**: $\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 \text{ subject to } \|\beta\|_0 \leq k$ (1)
  - **Lasso Regression**: $\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 \text{ subject to } \|\beta\|_1 \leq t$ (2)
  - **Ridge Regression**: $\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 \text{ subject to } \|\beta\|_2^2 \leq t$ (3)

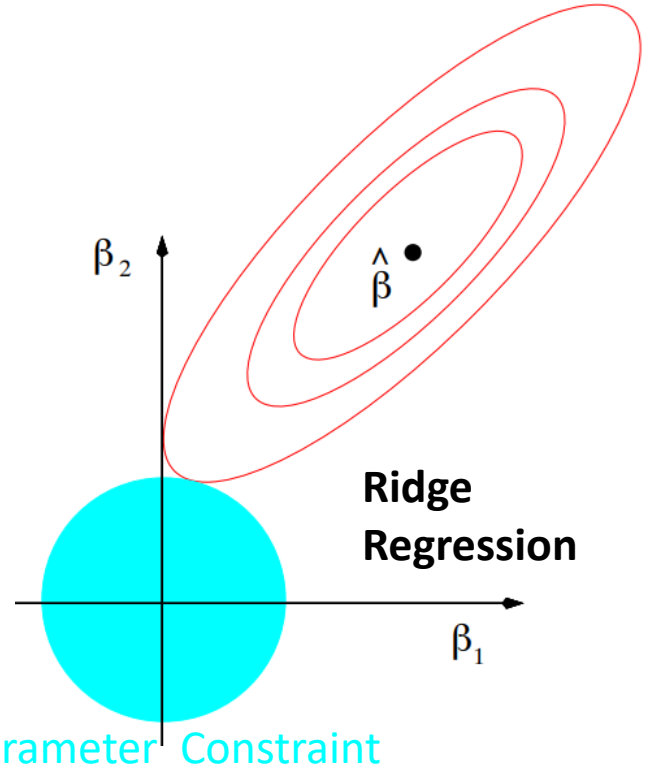
Source: Tibshirani – A Closer Look at Sparse Regression | Link

# Linear Regression: Regularization Options & Tradeoffs

- **Lasso Regression**: $\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 \quad \text{subject to} \quad \|\beta\|_1 \leq t$
- **Ridge Regression**: $\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 \quad \text{subject to} \quad \|\beta\|_2^2 \leq t$

**Lasso likely to yield "sparse" (fewer non-zero) parameters**




Source: Tibshirani – A Closer Look at Sparse Regression | Link

# Linear Regression: Regularization Options & Tradeoffs

- **Best Subset Selection:** $\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 \text{ subject to } \|\beta\|_0 \leq k$ (1)
- **Lasso Regression:** Convex $\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 \text{ subject to } \|\beta\|_1 \leq t$ (2)
- **Ridge Regression:** Convex $\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 \text{ subject to } \|\beta\|_2^2 \leq t$ (3)

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_0 \quad (4)$$

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1 \quad (5)$$

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2 \quad (6)$$

In Penalized Form

- **Equivalence**:
  - Problems (2) and (5) are equivalent (for any $t \geq 0$ and solution $\beta$ in (2), there is a value of $\lambda \geq 0$ such that $\beta$ also solves (5), and vice versa)
  - Problems (3) and (6) are equivalent as well.
  - Problems (1) and (4) are *not equivalent* and *non-convex*.
- **Sparsity & Convexity**: Only Lasso leads to both "sparse" and "convex" formulations!

Source: Tibshirani – A Closer Look at Sparse Regression | Link