Chapter 5: Machine Learning Basics

Deep Learning Textbook Study Group, SF

Safak Ozkan

March 27, 2017

Chapter 5: Machine Learning Basics

- Learning Algorithms (eg Linear Regression)
- Capacity, Overfitting and Underfitting
- The No Free Lunch Theorem
- Regularization
- Hyperparameters and Cross-validation
- Bias and Variance
- MLE
- Bayesian Statistics
- Supervised and Unsupervised Algorithms
- Stochastic Gradient Descent
- Building a ML Algorithm
- The Curse of Dimensionality
- Manifold Learning

Learning

- Experience: ***E***
  Task (Problem): ***T***
  Performance measure: ***P***

  The algorithm that's designed to do task ***T*** is said to be learning if its performance ***P*** improves with experience ***E***.

$$y = f(x) + \epsilon$$

Irreducible Error

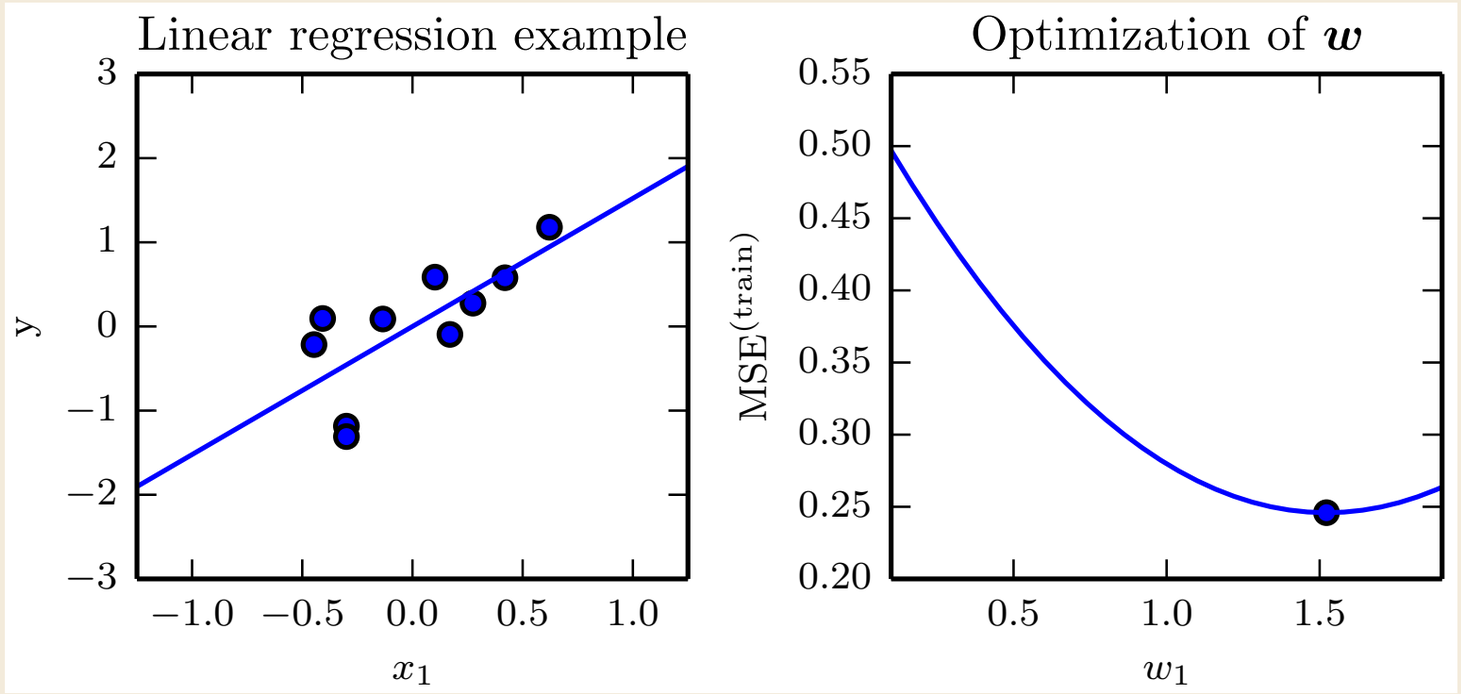
Learning

- TASKS, *T*
  - Classification
  - Regression
  - Transcription
  - Machine Translation
  - Anomaly Detection

- PERFORMANCE, *P*
  - log probability $log(p)$
  - MSE
  - accuracy, precision

- EXPERIENCE, *E*
  - processing the data set
  - Supervised: $(\mathbf{x}, \mathbf{y})$
  - Unsupervised: learning a $p(\mathbf{x})$

Linear Regression



$\hat{y} = \mathbf{w}^T\mathbf{x} + \mathbf{b}$ Task is to predict y from $\mathbf{x}$.

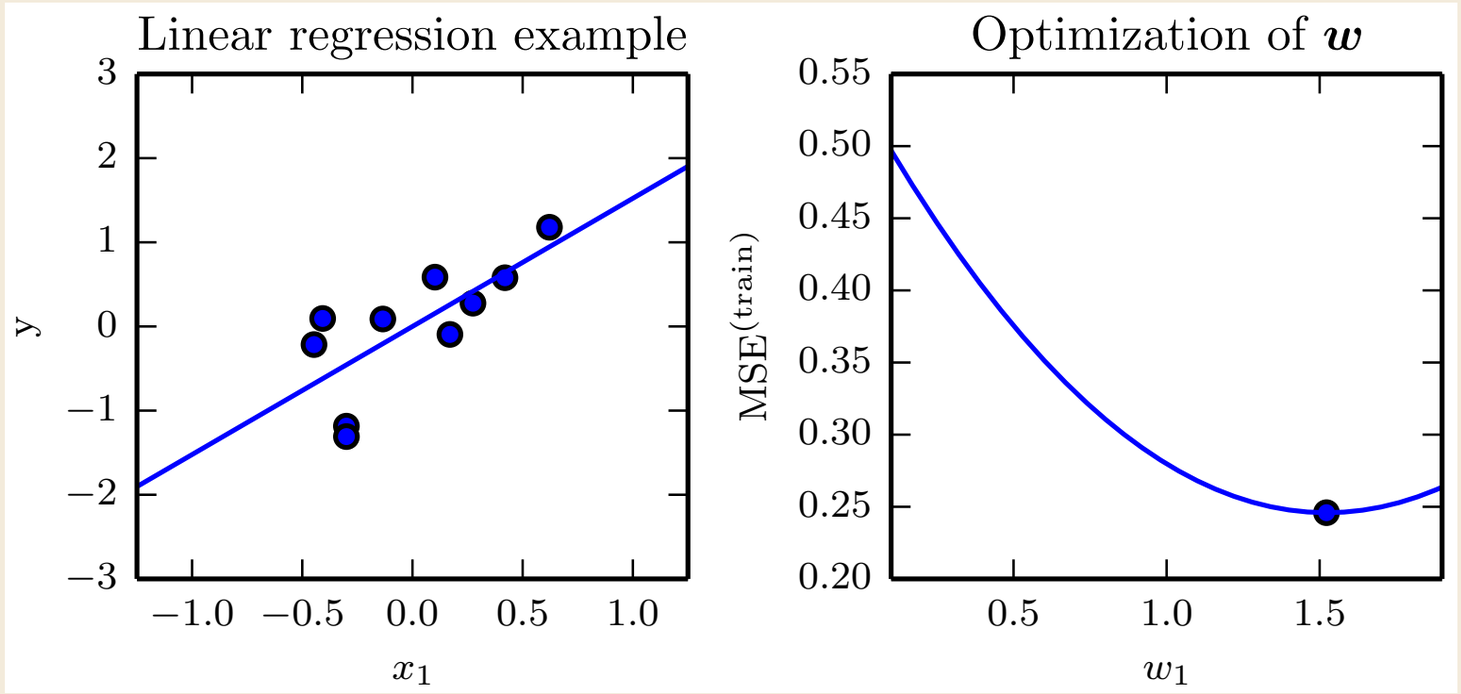
$X\mathbf{w} = \mathbf{y}$ ← overdetermined linear eqn

$X^T X\mathbf{w} = X^T\mathbf{y}$

$\mathbf{w} = (X^T X)^{-1}X^T\mathbf{y}$ ← Normal Equations

[Aside: $X\mathbf{w} = X(X^T X)^{-1}X^T\mathbf{y}$ ← Projection matrix]

Linear Regression

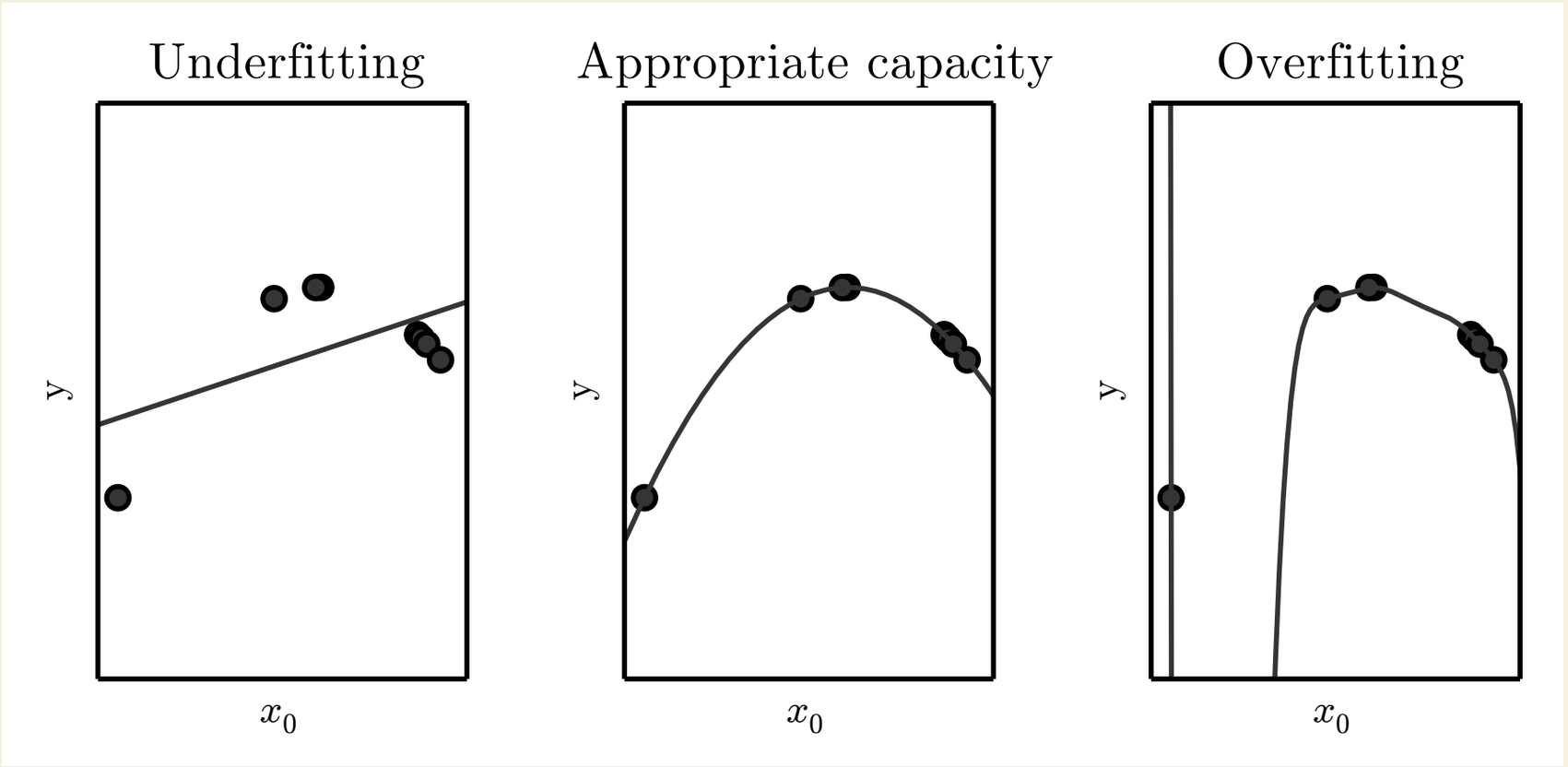


$\hat{y} = \mathbf{w}^T\mathbf{x} + \mathbf{b}$ Task is to predict y from $\mathbf{x}$.

$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$ Normal Equations

$arg\min_{\mathbf{w}} \frac{1}{2} \|A\mathbf{w} - \mathbf{y}\|_2^2$ MSE (Loss Func)

Overfitting and Underfitting



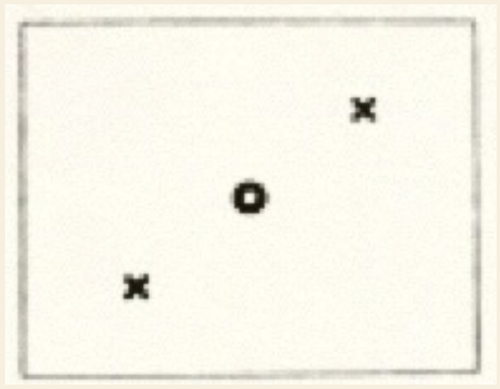
- **OCCAM's RAZOR:** **Amongst competing models that explain known observations equally well we should choose the simplest one.**

VC-Dimension (Capacity)

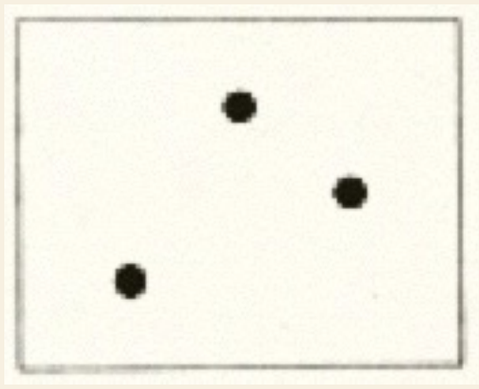
- **VC dimension (Capacity):** The **VC dimension** is the largest possible value of m for which there exists a training set of m different *data points* that the classifier can label *arbitrarily*.

e.g. 2D linear classifier

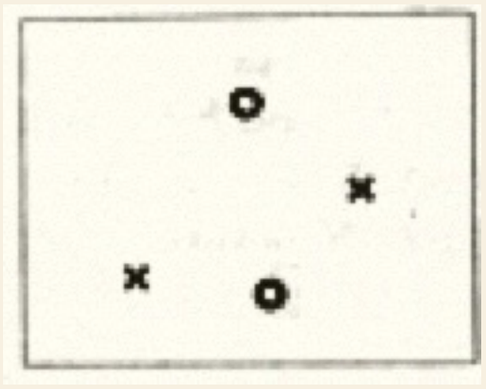
$$y = w_o + w_1 x_1 + w_2 x_2$$
$$y \in \{-1, +1\}$$



3 collinear data points can NOT be classified arbitrarily.



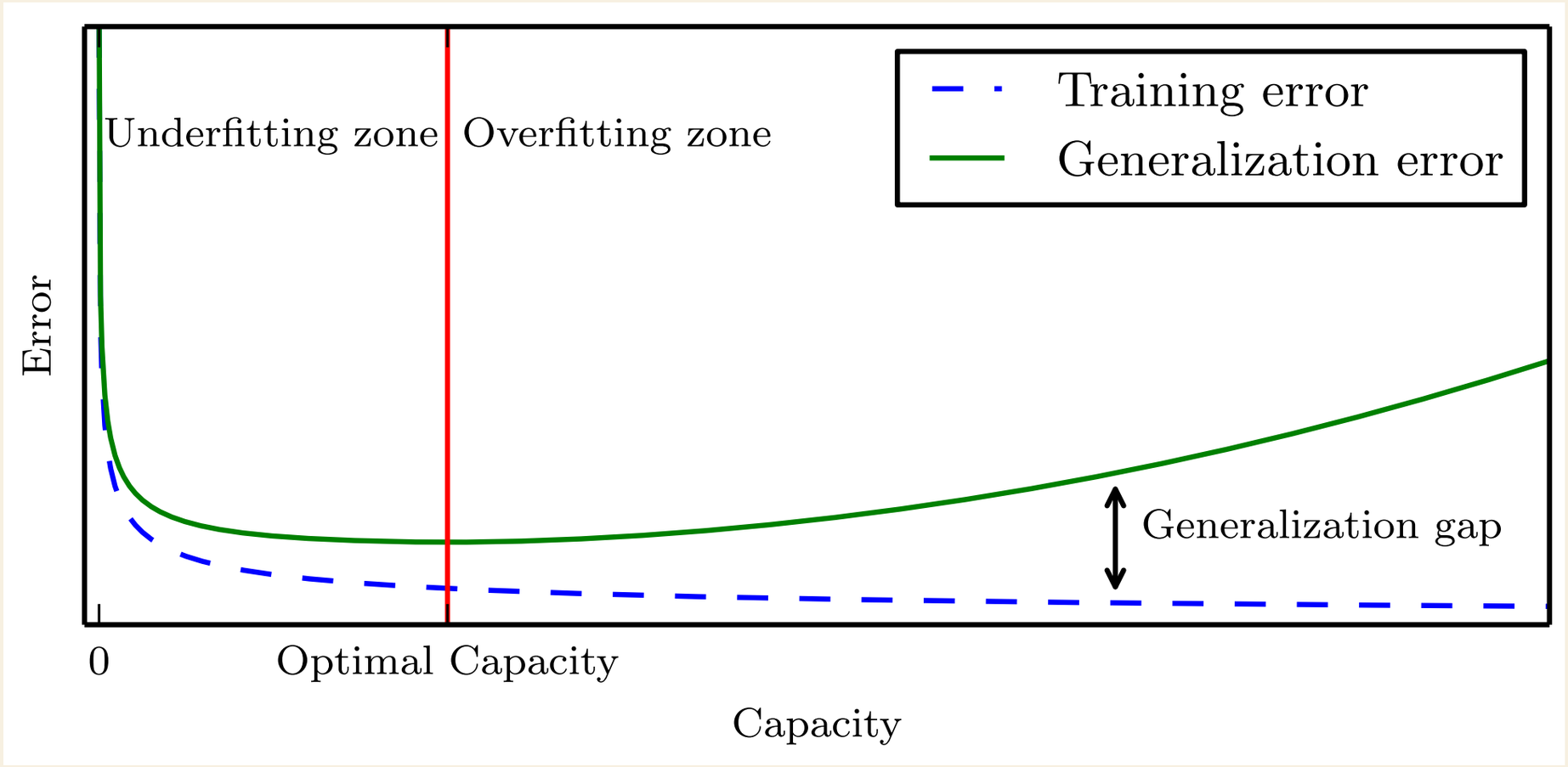
However, any 3 non-collinear datapoints can be classified arbitrarily.



There's **NO** 4 data points that can be classified arbitrarily.

Conclude: $d_{VC} = 3$

Generalization Error



- E_{in}
  "Training Error"
  "In-Sample Error"

- E_{out}
  "Test Error"
  "Out-of-Sample Error"

Generalization Error = E_{out} - E_{in}

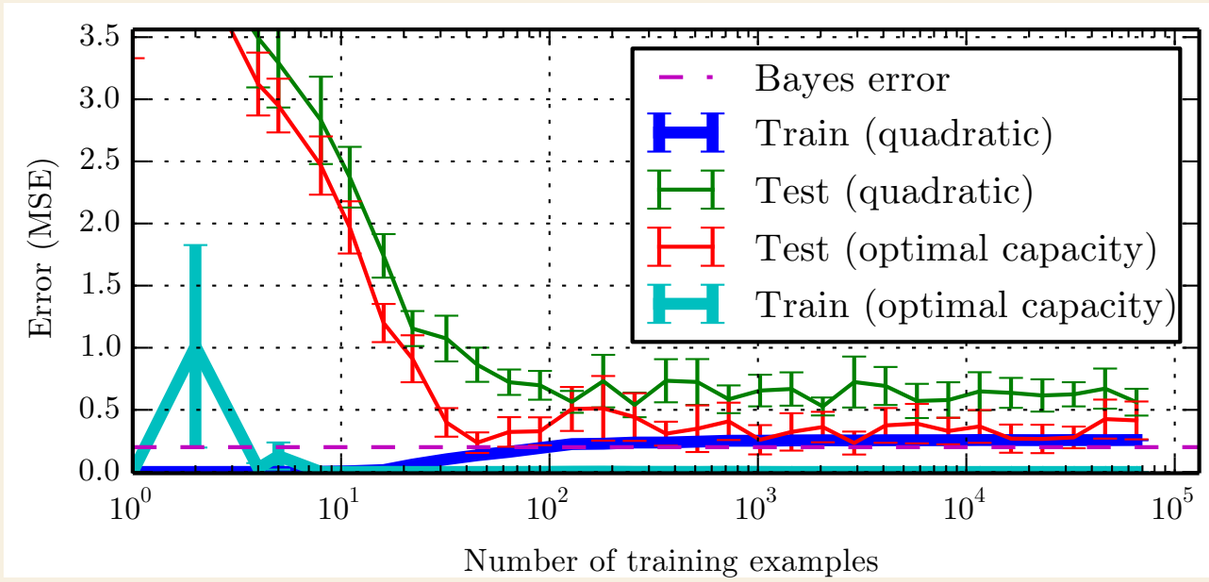
The No Free Lunch Theorem

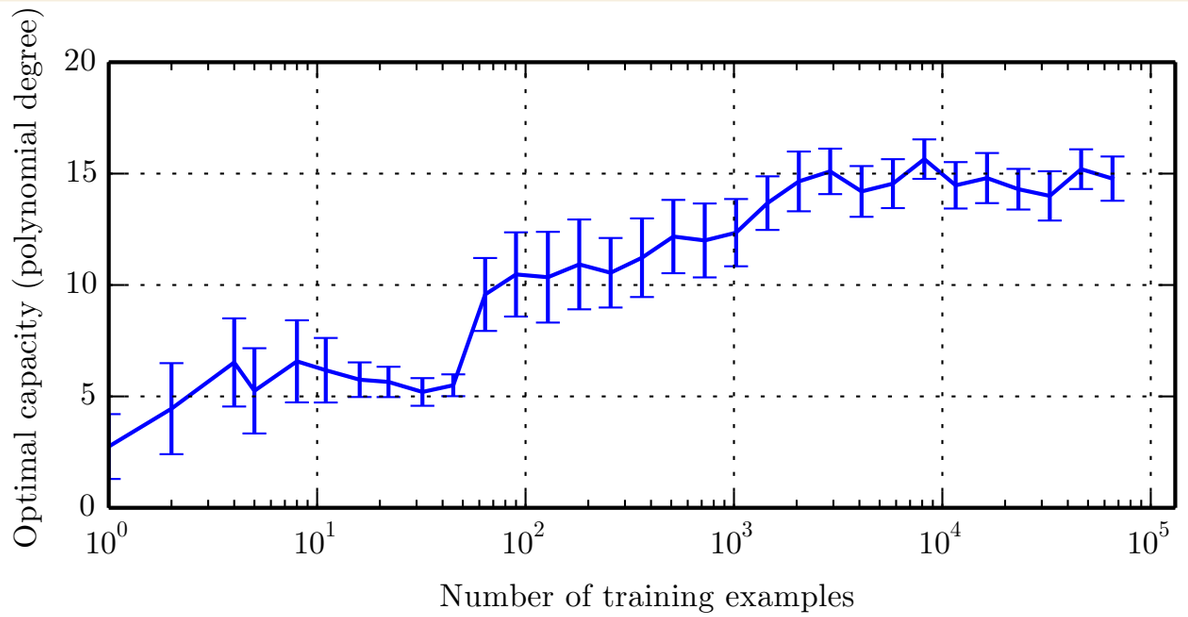
- *"Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points."*

- **Any tweaks you make to the ML model to explain a particular structure of the problem at hand, will make the model perform worse on other problems where that structure isn't present.**

- **We must design a specific ML model for a specific task.**

Training Dataset Size





Effect of training dataset size on MSE for a problem of a 5th degree polynomial

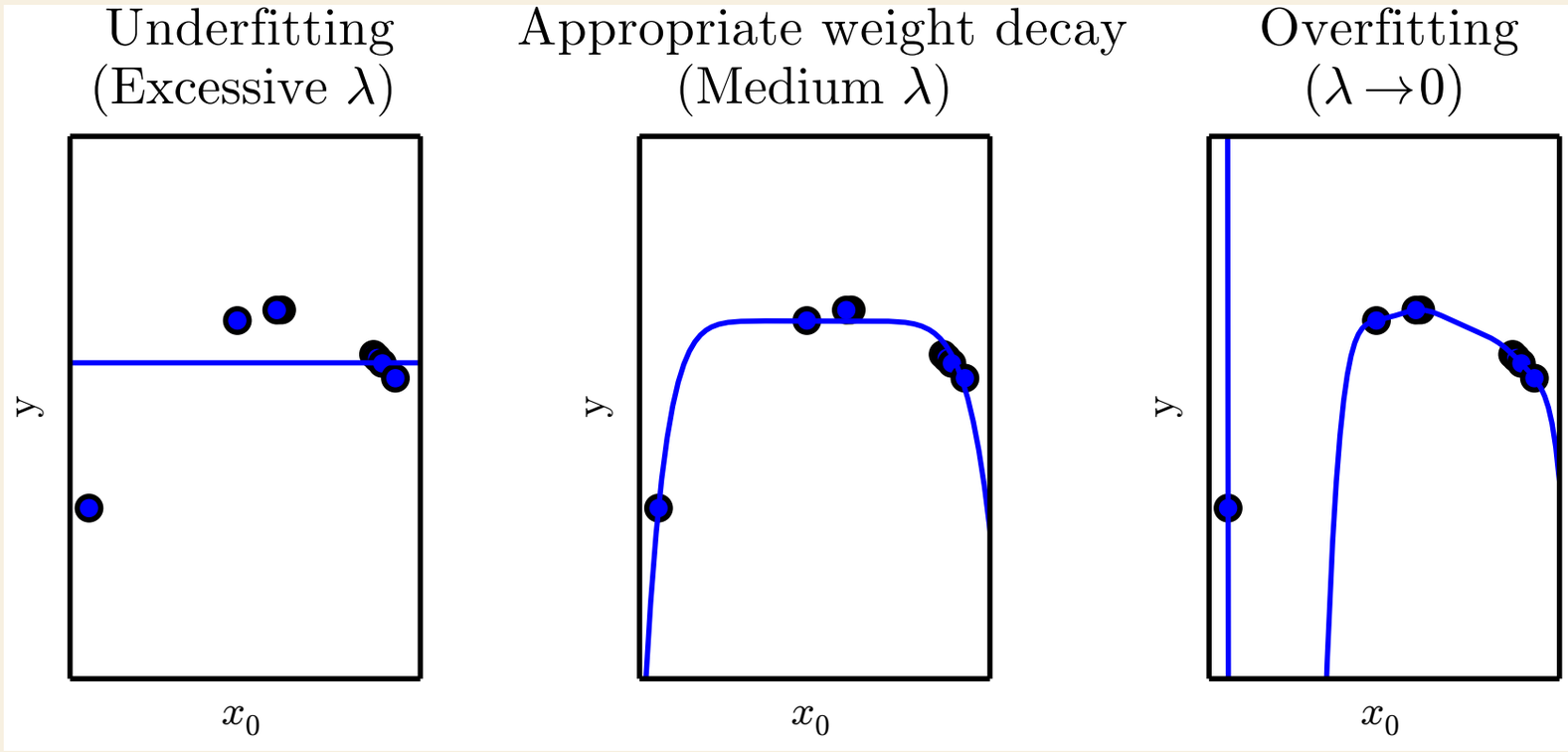
• **Bayes Error:** The error incurred by an ideal model making predictions from the true distribution $p(\mathbf{x}, y)$ is called the **Bayes Error** [Goodfellow].

It is the lowest possible error rate for any classifier and is analogous to the **irreducible error.**

$$y = f(\mathbf{x}) + \epsilon$$

Regularization

- **Regularization** is any modification we make to the learning algorithm that is intended to reduce its **generalization error** (**test error**) but **NOT** the **training error.** (NFL Theorem also applies to regularization)



$$J(\mathbf{w}) = J_{train}(\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}$$

L_2 norm of error on training data

Weight-decay regularization term

Bias-Variance Tradeoff

$$MSE = E_x[(g^D(x) - f(x))^2]$$

model (estimator of $f(x)$)

target func.

$$E_D[MSE] = E_D[E_x[(g^D(x) - f(x))^2]]$$

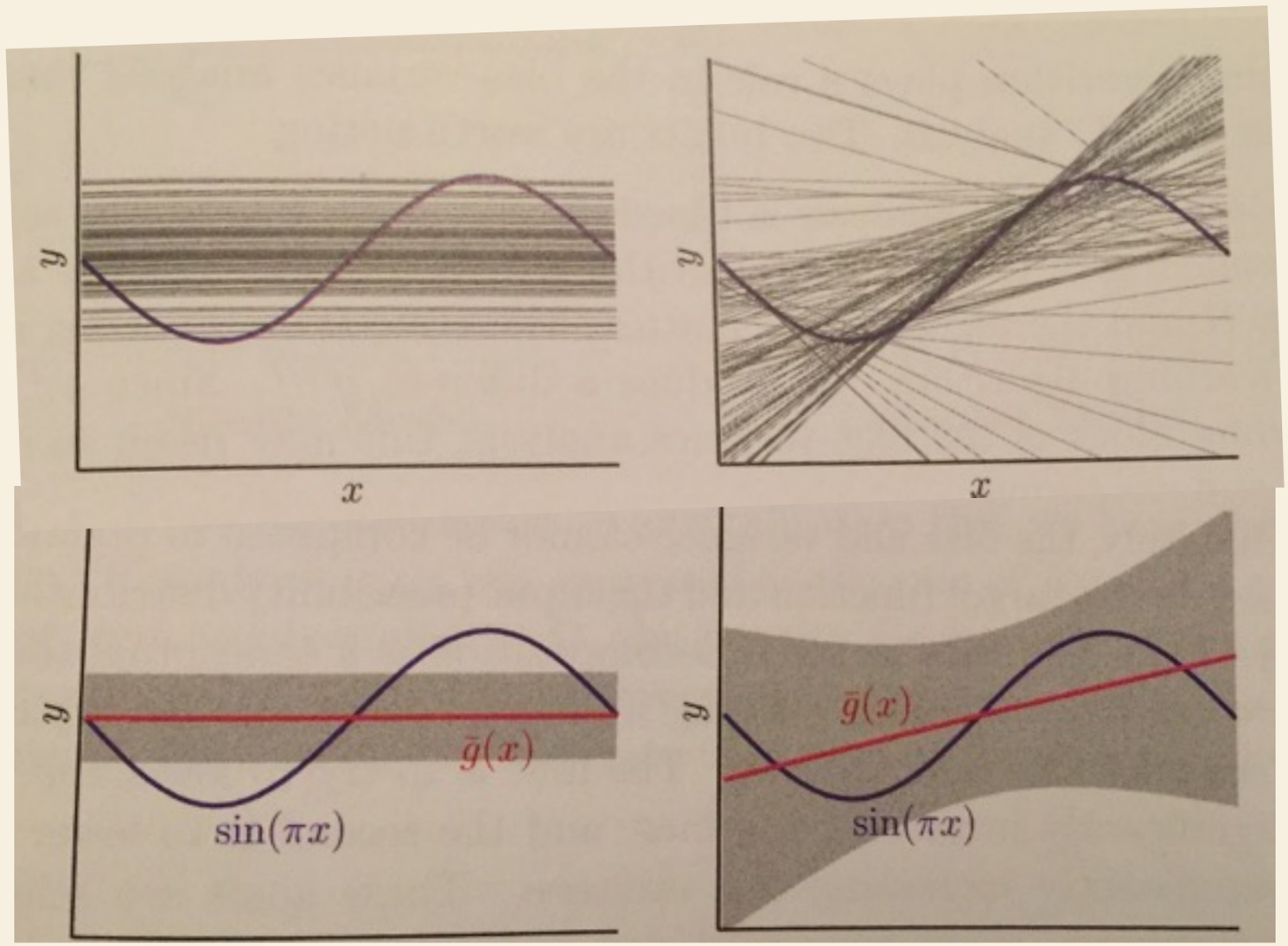
$$= E_x\left[E_D[g^D(x)^2] - 2\underbrace{E_D[g^D(x)]}_{\bar{g}(x)}f(x) + f(x)^2\right].$$

$$= E_x\left[\underbrace{E_D[g^D(x)^2] - \bar{g}(x)^2}_{E_D\left[\,(g^D(x) - \bar{g}(x))^2\,\right]} + \underbrace{\bar{g}(x)^2 + f(x)^2 - 2\bar{g}(x)f(x)}_{(\bar{g}(x) - f(x))^2}\right]$$

var(x) ***bias(x)^2***

Bias-Variance Tradeoff

- 2-data point samples



$E_x[bias(x)]^2 = 0.50$
$E_x[var(x)] = 0.25$

$E_x[bias(x)]^2 = 0.21$
$E_x[var(x)] = 1.69$

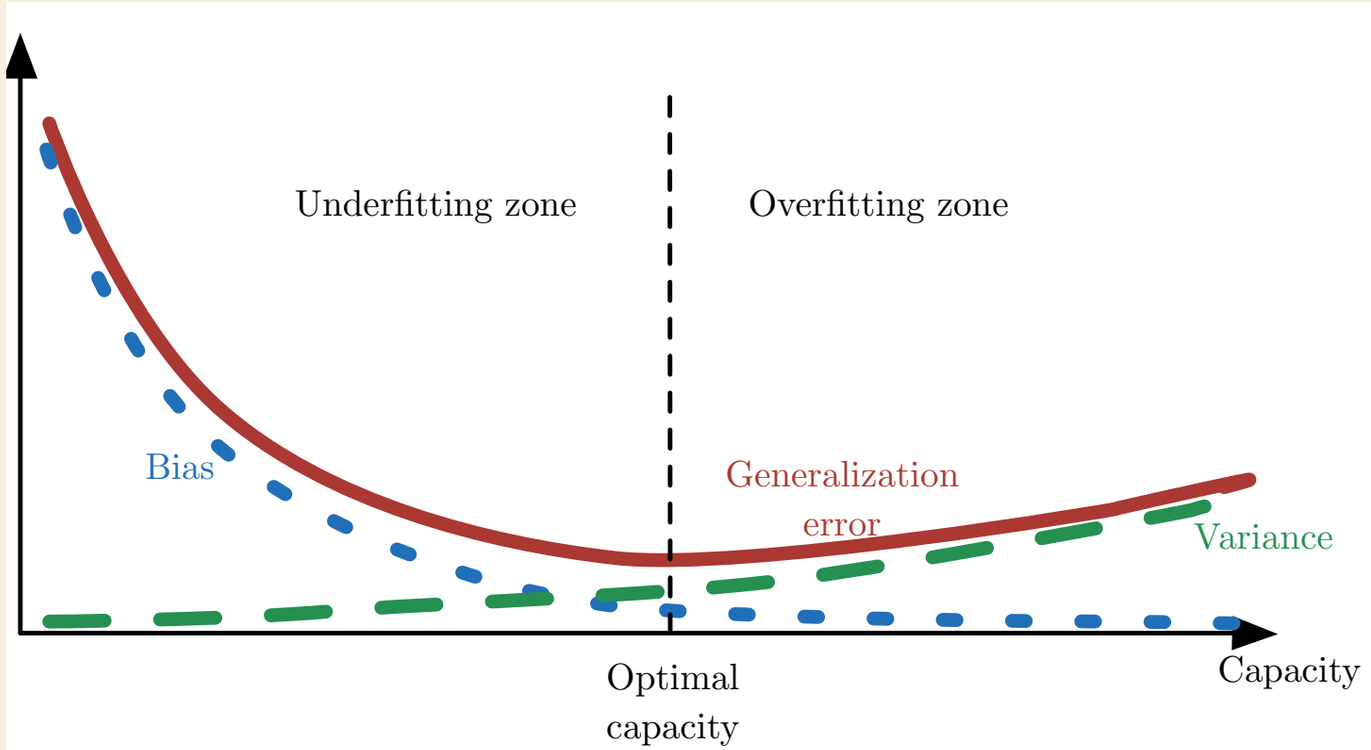
Bias and Variance

- $bias(\hat{\theta}_m) = \mathbb{E}(\hat{\theta}_m) - \theta$

$\hat{\theta}_m$: is a random variable, and it's computed over set of m data points

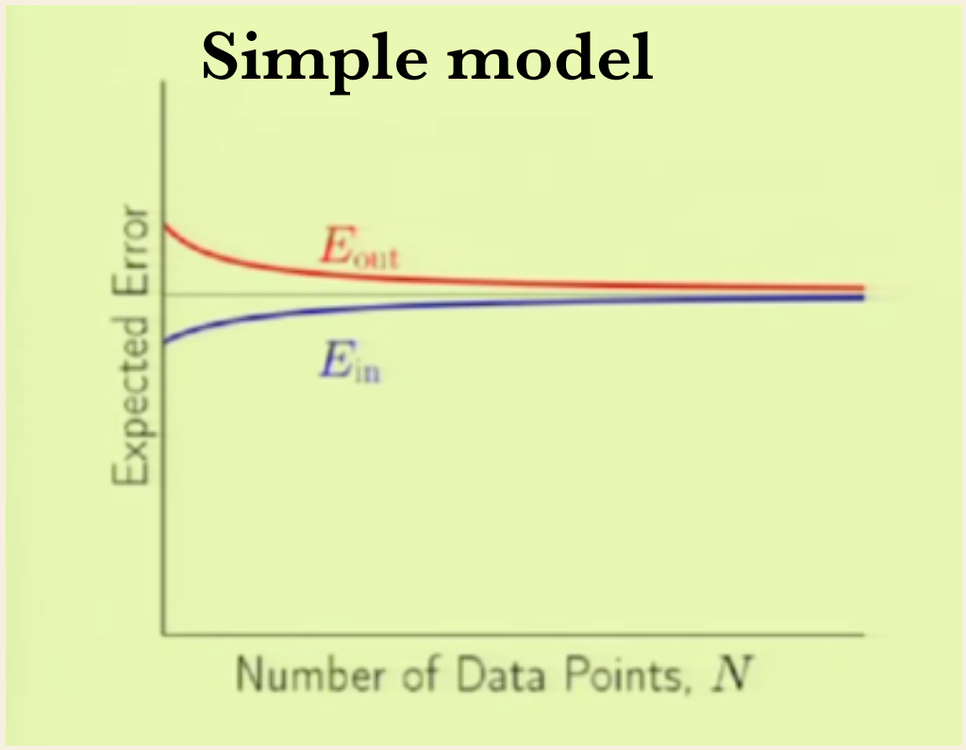
θ : True fixed value like (μ, σ) It's a deterministic scalar.

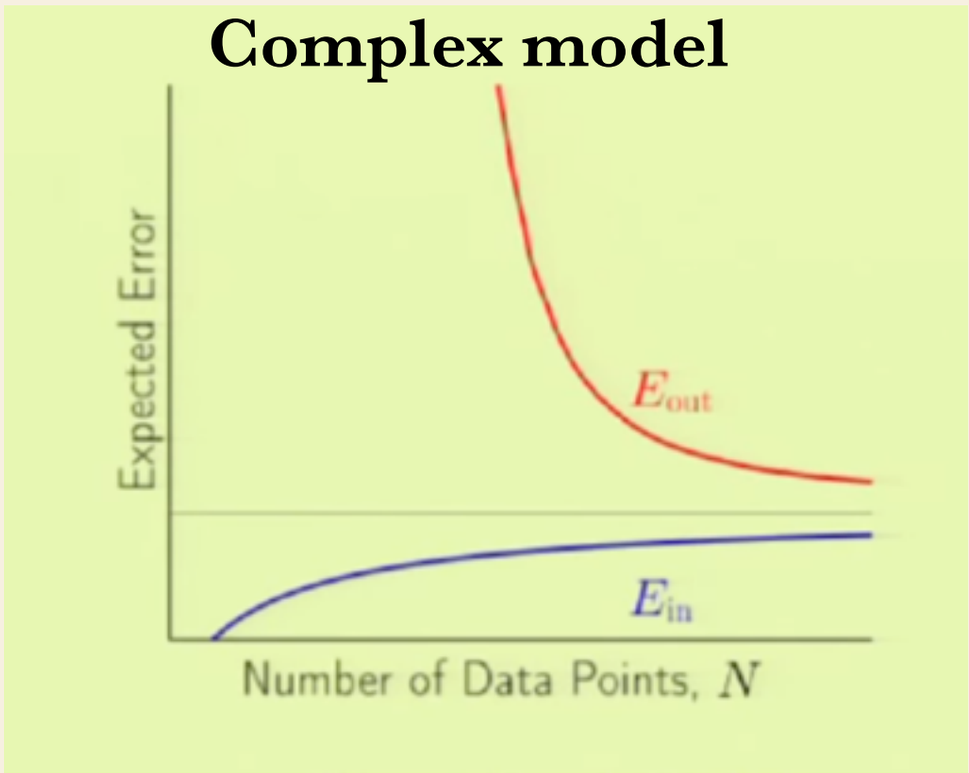
- $Var(\hat{\theta}_m)$

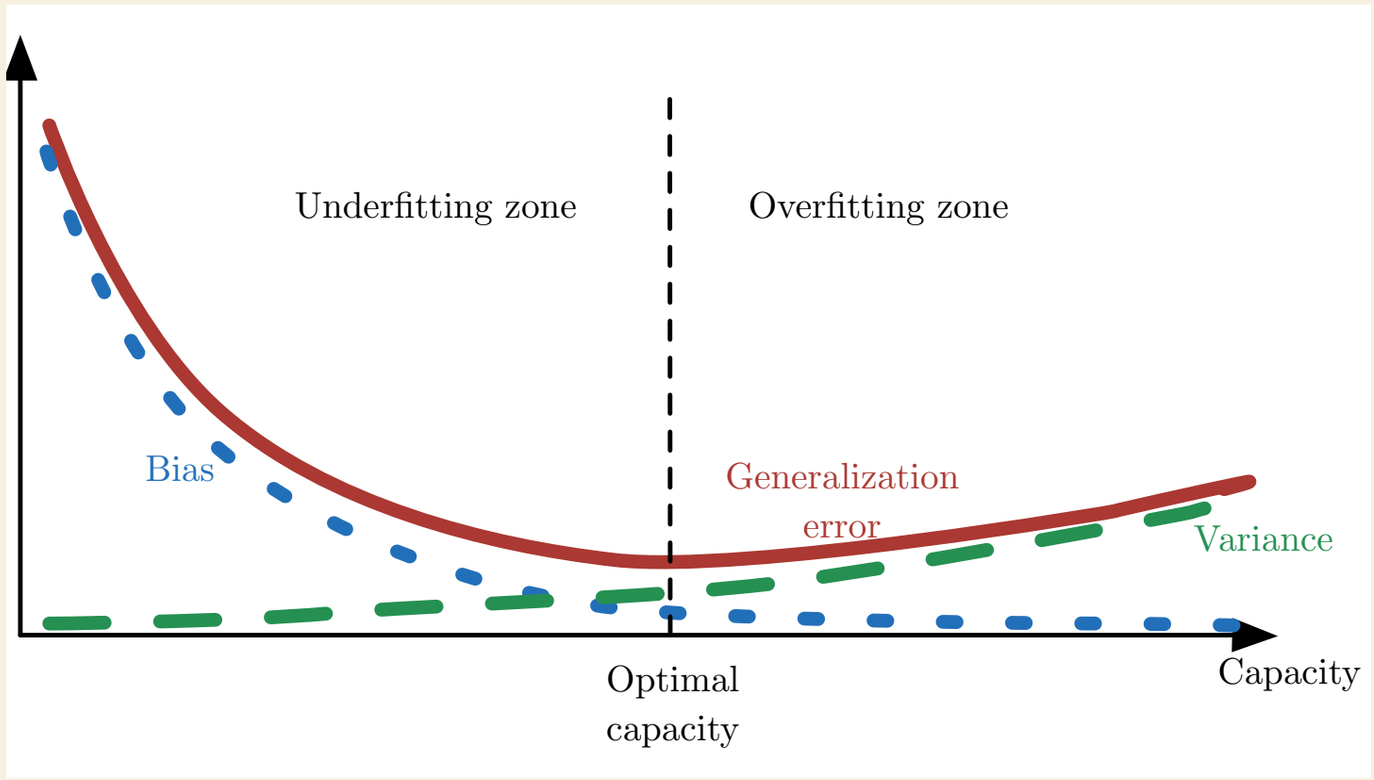


- **Very simple models** don't get too close to the truth, but their results don't change amongst different data sets.
- **Very complicated models** can be accurate on average but they tend to be sensitive to small changes in inputs.

Bias and Variance







- **Very simple models** don't get too close to the truth, but their results don't change amongst different data sets.
- **Very complicated models** can be accurate on average but they tend to be sensitive to small changes in inputs.

Maximum Likelihood Estimation

- MLE gives the **founding principles** for finding the best **estimator.**

$p_{data}(\boldsymbol{x})$ ← *data generating distribution*

$p_{model}(\boldsymbol{x}, \boldsymbol{w})$ ← *estimate*

$$= \boldsymbol{w}^T \boldsymbol{x}^{(i)} + \mathcal{N}(0, \sigma)$$

$$\hat{\boldsymbol{w}} = arg \max_{\theta} \prod_{i=1}^{m} p_{model}(\boldsymbol{x}^{(i)}, \boldsymbol{w})$$

$$\hat{\boldsymbol{w}} = arg \max_{\theta} \sum_{i=1}^{m} log\ p_{model}(\boldsymbol{x}^{(i)}, \boldsymbol{w})$$

- Connection with **KL Divergence**

$$D_{KL}(\hat{p}_{data} || p_{model}) = \mathbb{E}_X \left[log\ \hat{p}_{data} - log\ p_{model} \right]$$

$$\hat{\boldsymbol{w}} = arg \min_{\boldsymbol{w}} \left(-\mathbb{E}_X \left[log\ p_{model} \right] \right)$$

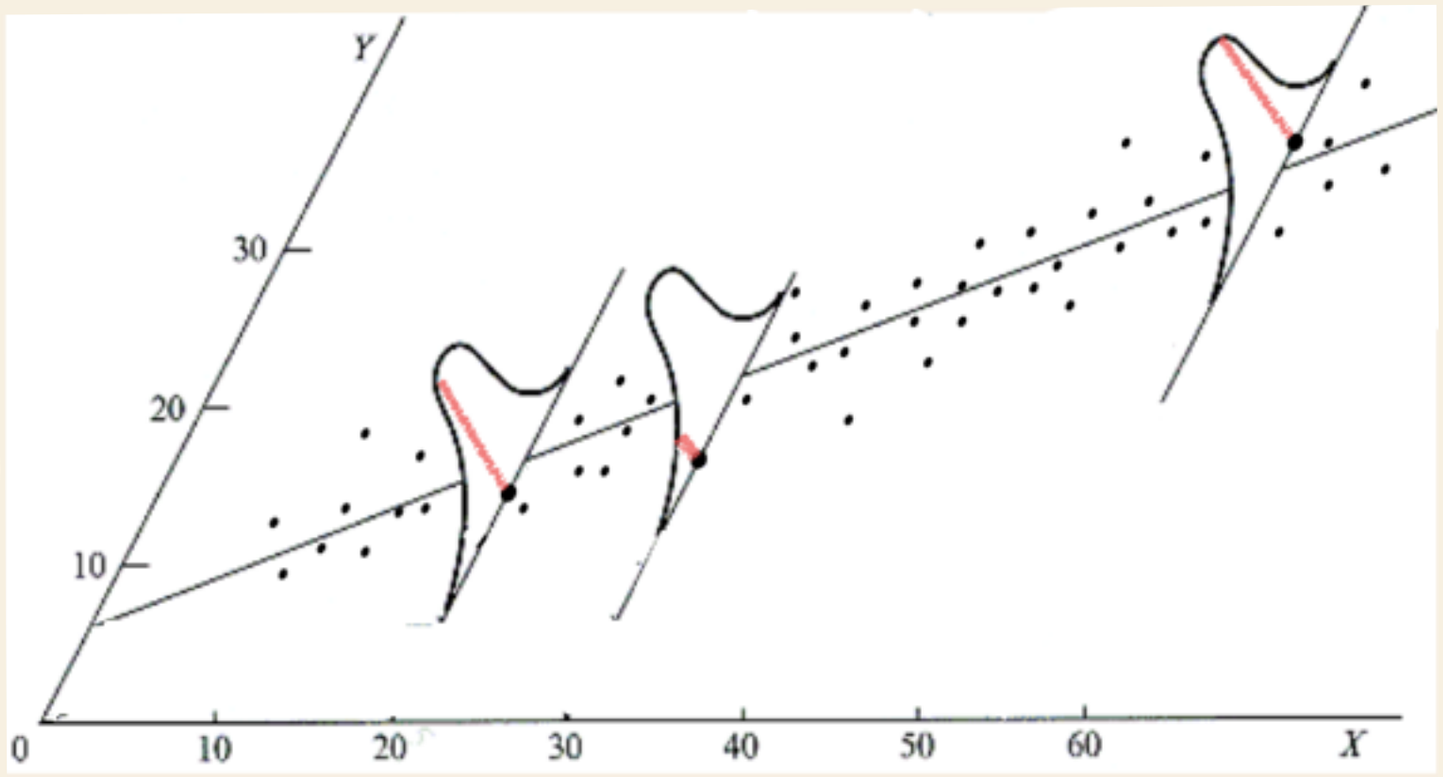
Maximum Likelihood Estimation

- e.g. Linear Regression Problem

$$\hat{y} = w_o + w_1 x$$

$$\hat{\boldsymbol{w}} = arg\ \max_{\boldsymbol{w}} \sum_{i=1}^{m} log\ p_{model}(\boldsymbol{x}^{(i)}, \boldsymbol{w})$$

data



$$p_{model}(x) = \hat{y}(x) + \epsilon \sim \mathcal{N}\left(\mu = \hat{y}(x), \sigma^2\right)$$

$$p_{model}(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\left(y^{(i)} - \hat{y}^{(i)}\right)^2 / 2\sigma^2}$$

$$MSE_{train} = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

In general:

$$p(y|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{cons.}\, e^{-loss(y|\boldsymbol{x}, \boldsymbol{w})}$$

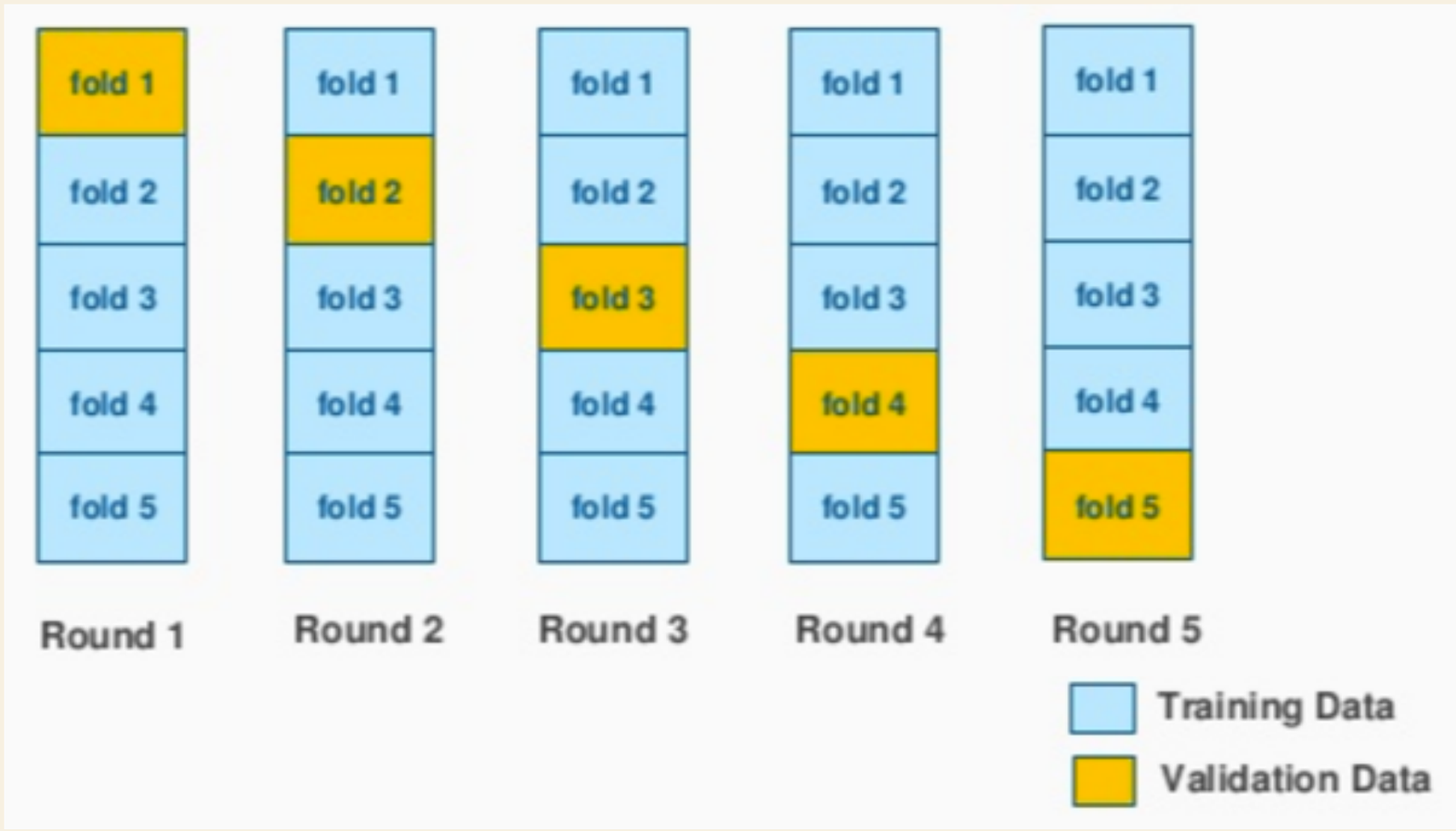
Hyper parameters and Cross-Validation

65 % → • **Training data:** For training the model weights, $\mathbf{w}$.

15 % → • **Cross-Validation data:** For tuning hyper parameters.

---

20 % → • **Test Data:** For assessing the model accuracy



Support Vector Machines

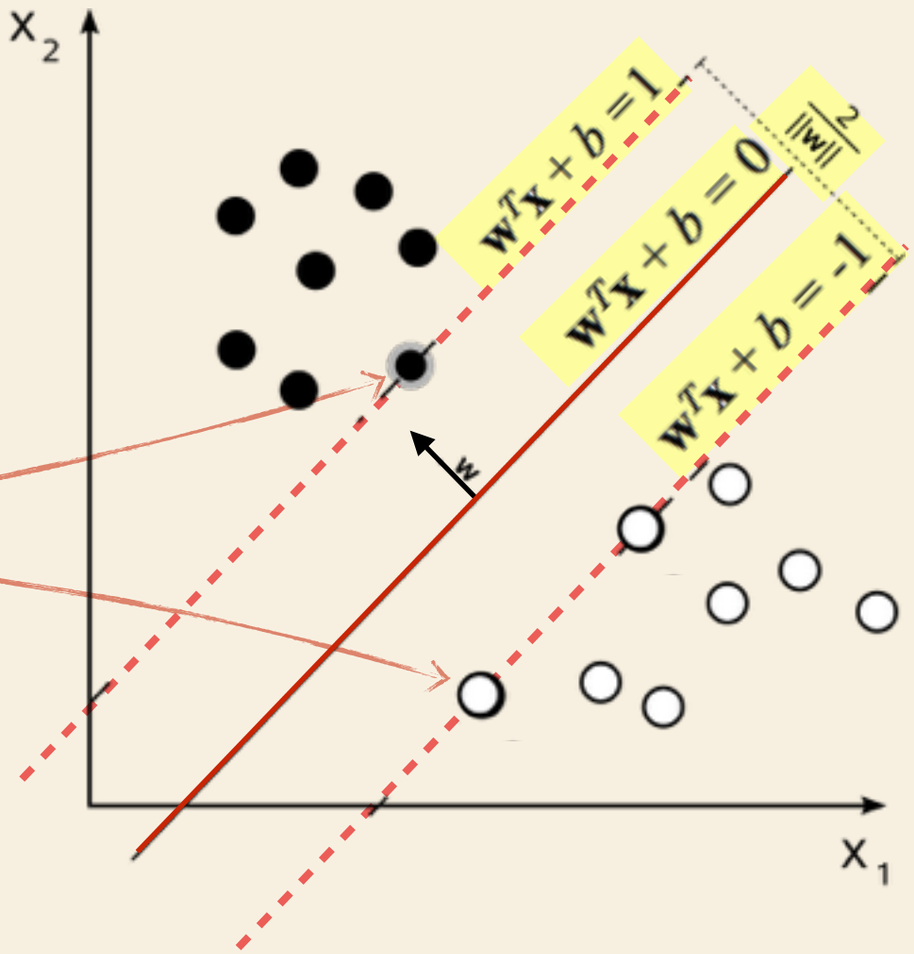
- **Two linearly separable classes**

$$(\boldsymbol{x}^{(i)}, y^{(i)}), \quad y^{(i)} \in \{-1, +1\}$$ **data**

$$\begin{cases} \gamma^{(i)} = \dfrac{\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b}{\|\boldsymbol{w}\|} & \text{distance of point } (i) \text{ to hyperplane} \\ \boldsymbol{w}^T \boldsymbol{x} - b = 1, -1. & \text{on the edges.} \end{cases}$$

$$max \ \frac{2}{\|\boldsymbol{w}^2\|}, \ such \ that \ y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b) \geqslant 1.$$

margin width **constraint**



Constrained Optimization Objective:

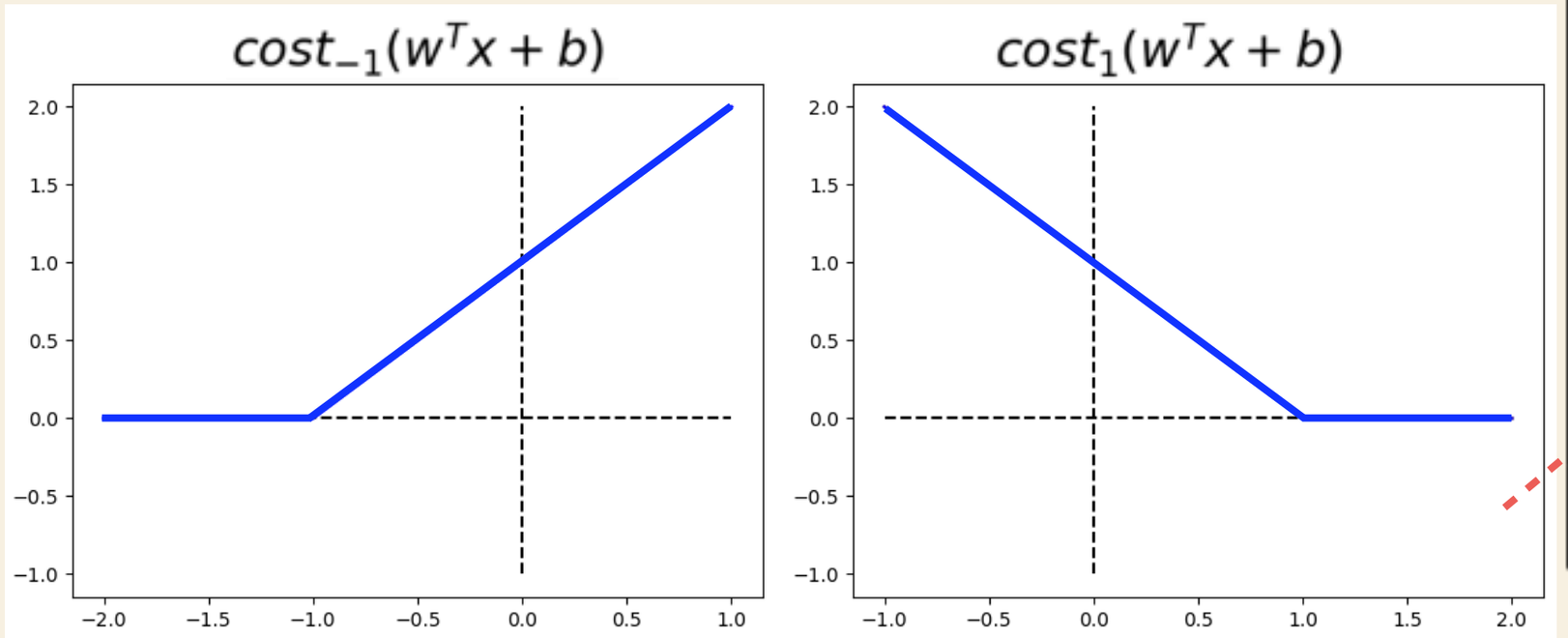
$$\mathcal{L}(\boldsymbol{w}, b, \alpha) = \tfrac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{m} \alpha_i [\, y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b) - 1].$$

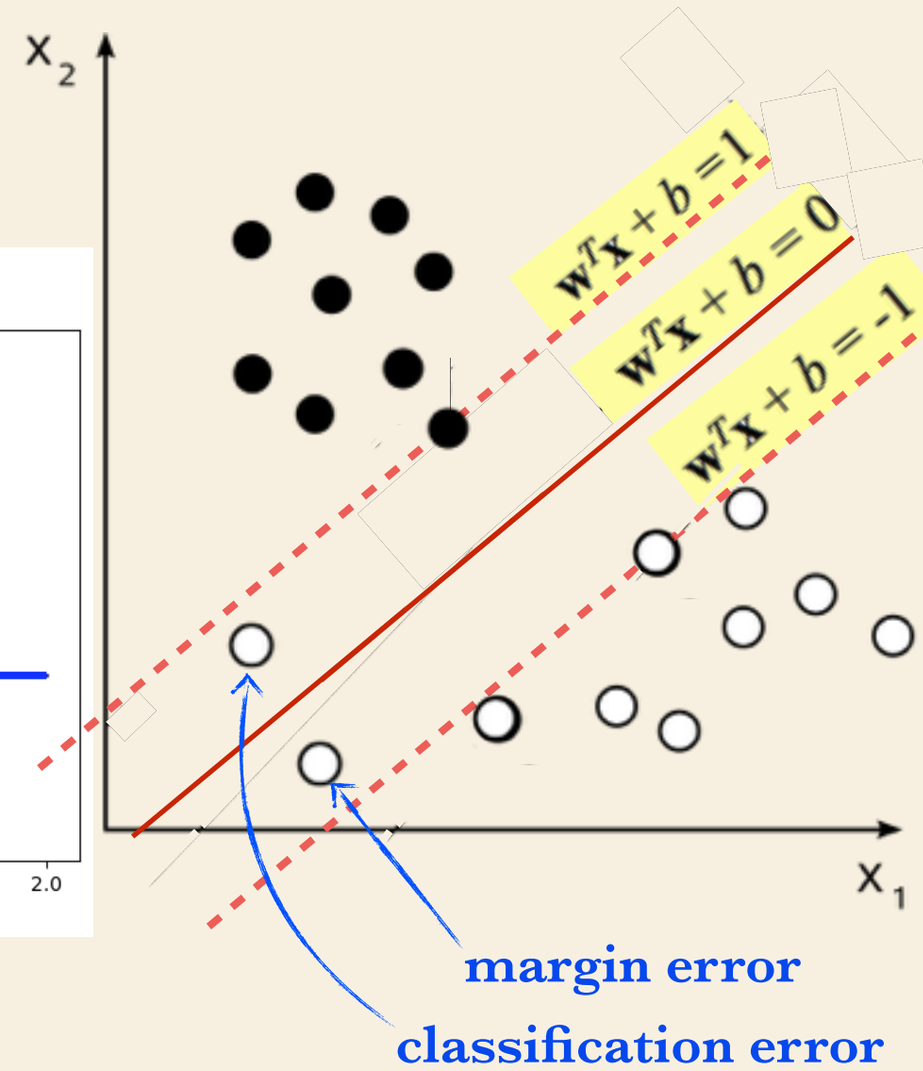
inequality constraint

$\nabla_{\boldsymbol{w}} \mathcal{L} = 0$ yields $\boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}$ terms.

Support Vector Machines

- Hinge loss





$$\hat{\boldsymbol{w}} = arg\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m} max(0, 1 - y^{(i)}(\boldsymbol{w}^T\boldsymbol{x} + b))$$

Regularization parameter

SVM loss

Support Vector Machines

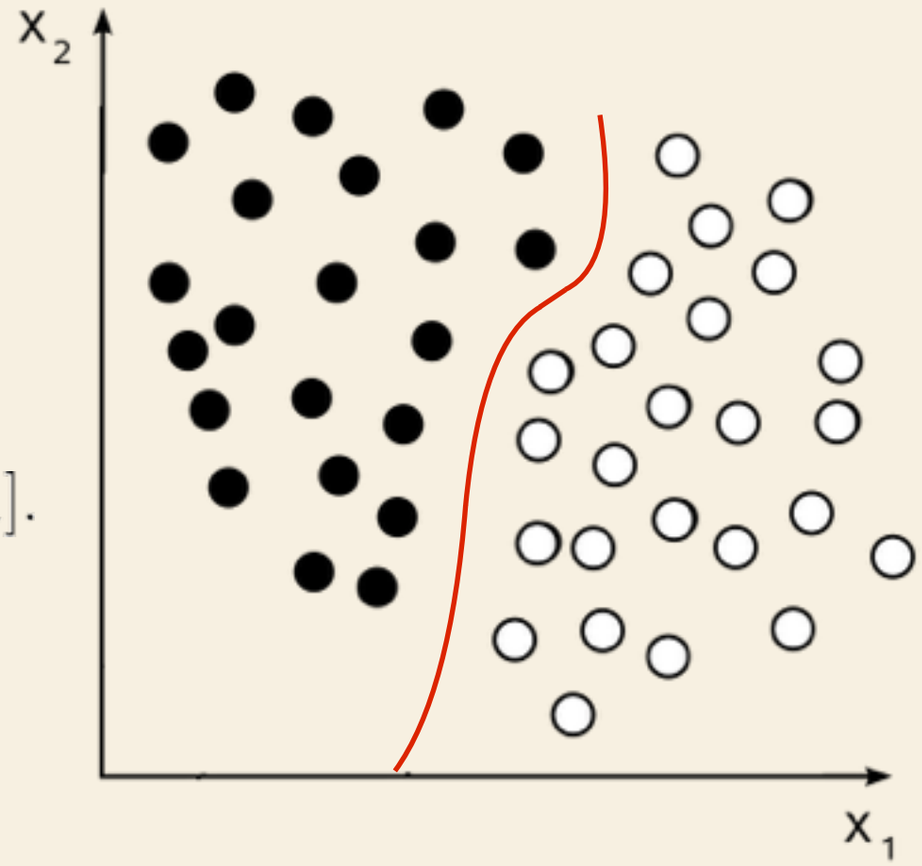
- **Linearly inseparable data**

m points in (*m*-1) dimensional space is always separable.

$$\boldsymbol{x}^{(i)} \rightarrow \phi(\boldsymbol{x}^{(i)})$$ Transform the data to a **higher dimensional** space.

$$\mathcal{L}(\boldsymbol{w}, b, \alpha) = \tfrac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{m} \alpha_i [\, y^{(i)}(\boldsymbol{w}^T \phi(\boldsymbol{x}^{(i)}) + b) - 1].$$

$\nabla_{\boldsymbol{w}} \mathcal{L} = 0$ yields $\phi(\boldsymbol{x}^{(i)}) \cdot \phi(\boldsymbol{x}^{(j)})$ terms.



- **Kernel Trick**

$K(\boldsymbol{u}, \boldsymbol{v}) = \phi(\boldsymbol{u}) \cdot \phi(\boldsymbol{v})$ ← $K(\boldsymbol{u}, \boldsymbol{v})$ calculates $\phi(\boldsymbol{u}) \cdot \phi(\boldsymbol{v})$ without having to calculate either $\phi(\boldsymbol{u})$ or $\phi(\boldsymbol{v})$.

$$K(\boldsymbol{u}, \boldsymbol{v}) = exp\left(-\frac{||\boldsymbol{u} - \boldsymbol{v}||^2}{2\sigma^2}\right)$$

- **Test Time**

$\alpha_i \neq 0$ only at **support vectors.**

Principal Components Analysis

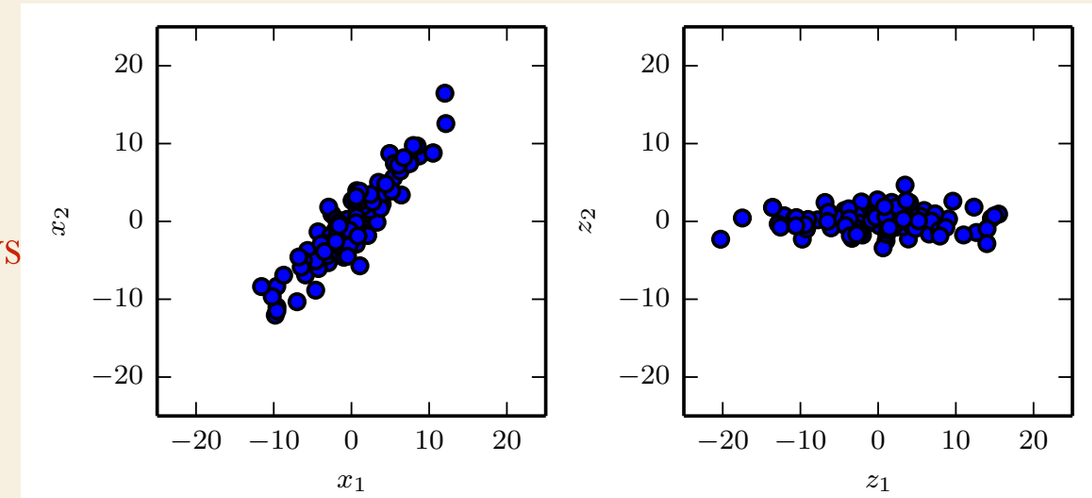
- **SVD**

$$X = U \Sigma V^T$$

$$Xv = \sigma u$$

$$X^T u = \sigma v$$

Singular vectors are always orthogonal for any matrix ***X***.



- **PCA**

$$\underbrace{X^T X} = (V \Sigma U^T)(U \Sigma V^T)$$
$$= V \Sigma^2 V^T$$

Symmetric Square Matrix

eigenvectors

eigenvalues

$$z = V^T x$$

representation of x in "eigen coordinates"

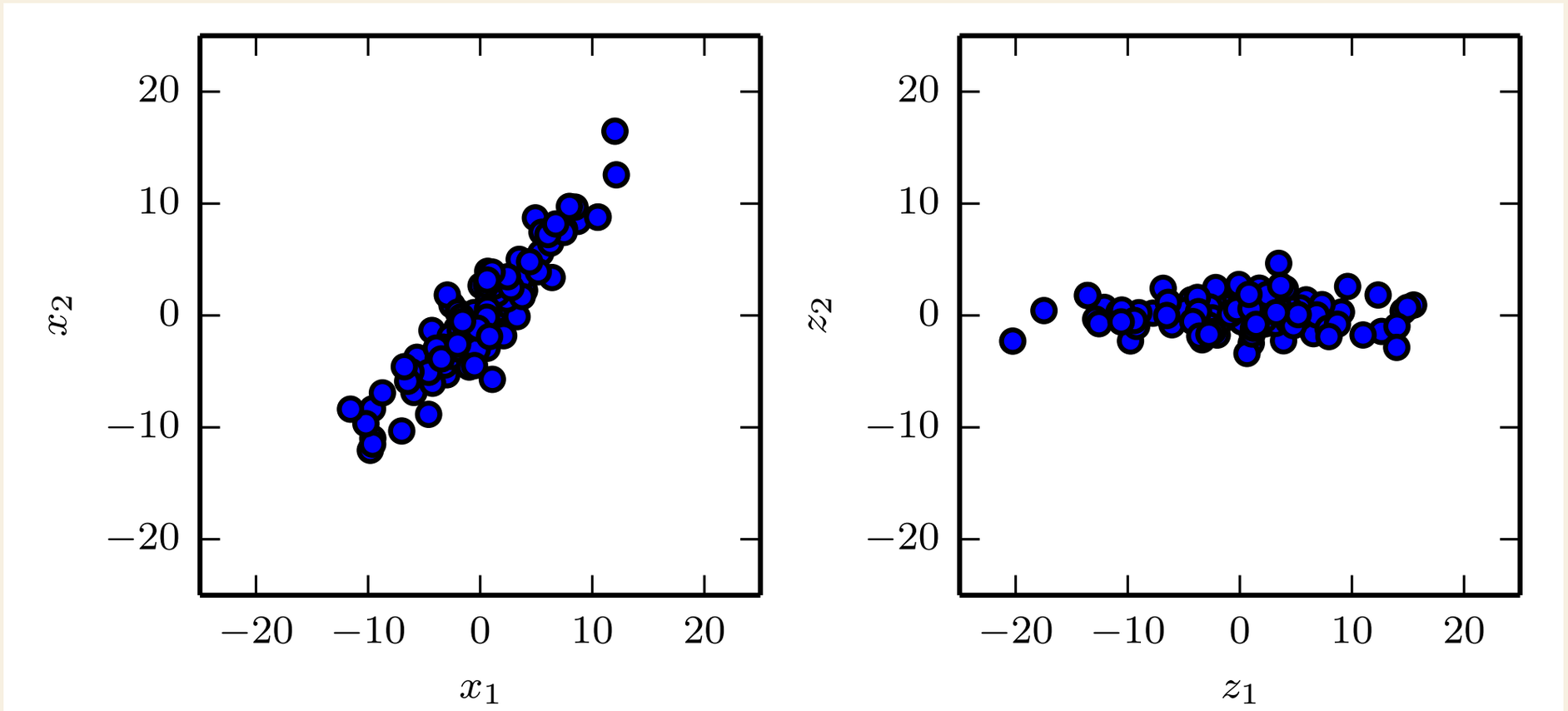
- **Eckhart-Young Thm:**

$$\|X - X_k\|_2^2 = \sigma_{k+1}^2$$

$$\|X - X_k\|_F^2 = \sum_{k+1}^{n} \sigma_i^2$$

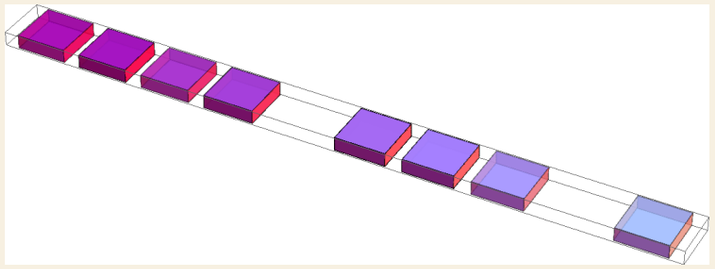
rank ***k*** approximation of ***X***.

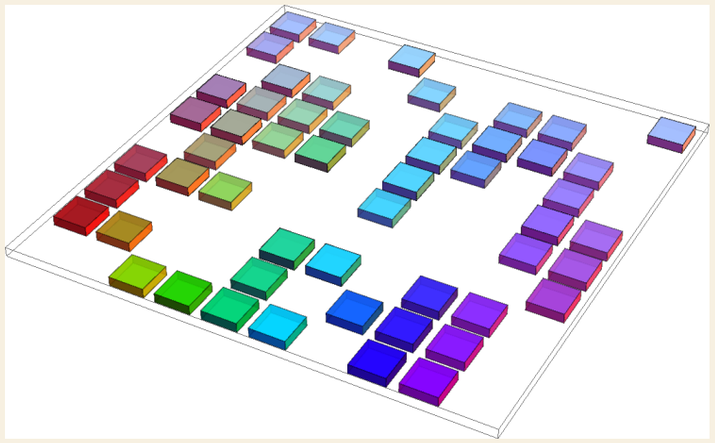
Principal Components Analysis

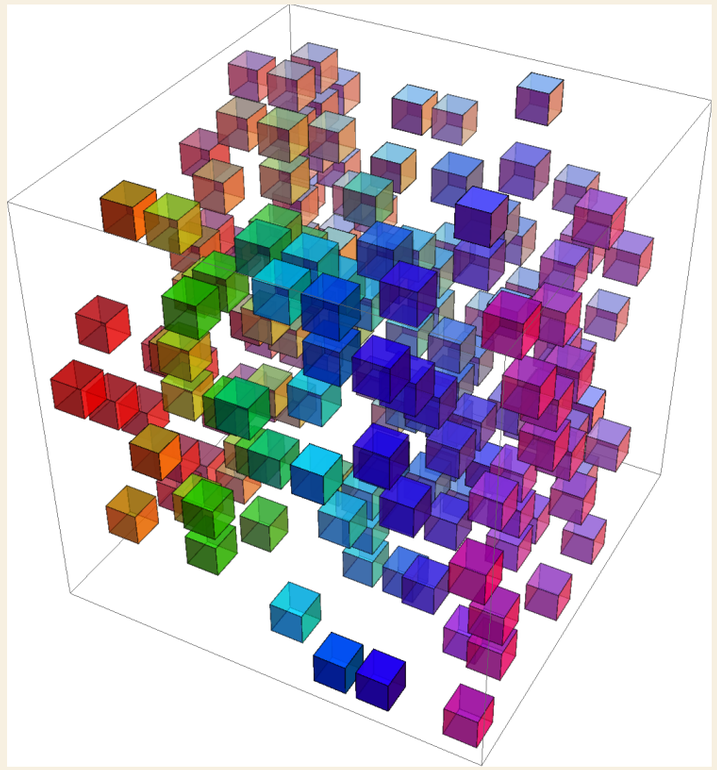


N.B. PCA is not linear regression.

Curse of Dimensionality







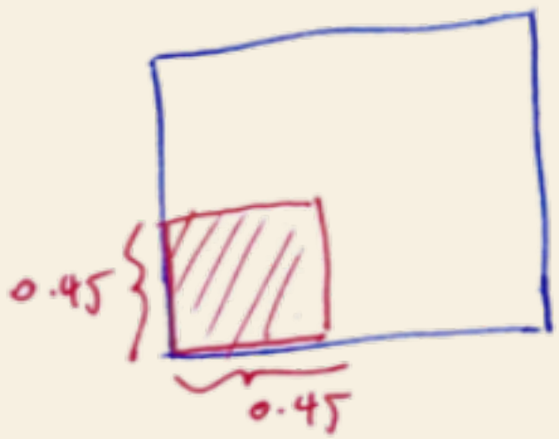
- For v-many values per each of the d many dimensions we'll need $O(v^d)$ examples.
- **Challenge:** the number of possible configurations of $\boldsymbol{x}$ is much larger than the number of training examples.

Curse of Dimensionality

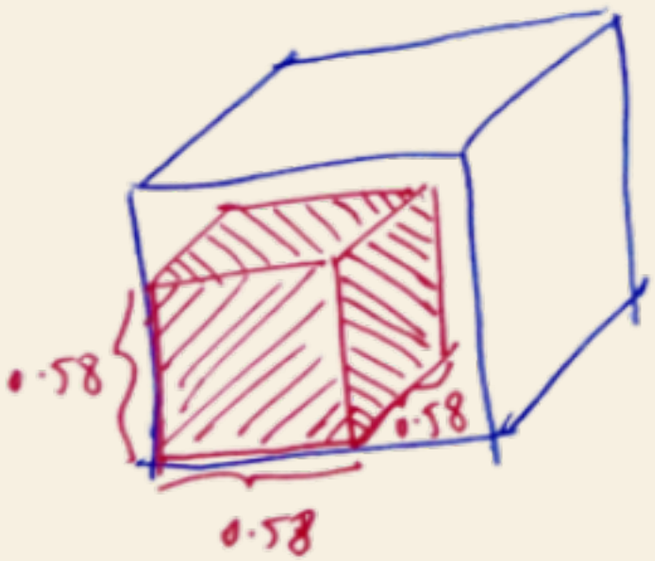
eg. Covering the 20% of data in n-dim Feature Space

n=1

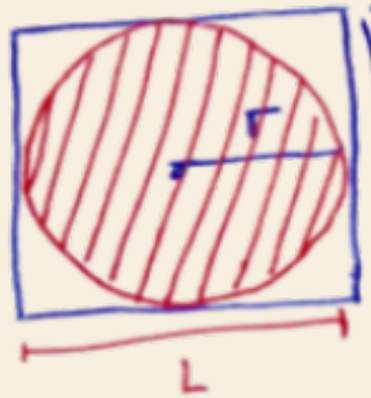
n=2



n=3



Volume of the Central Region of the Hypercube



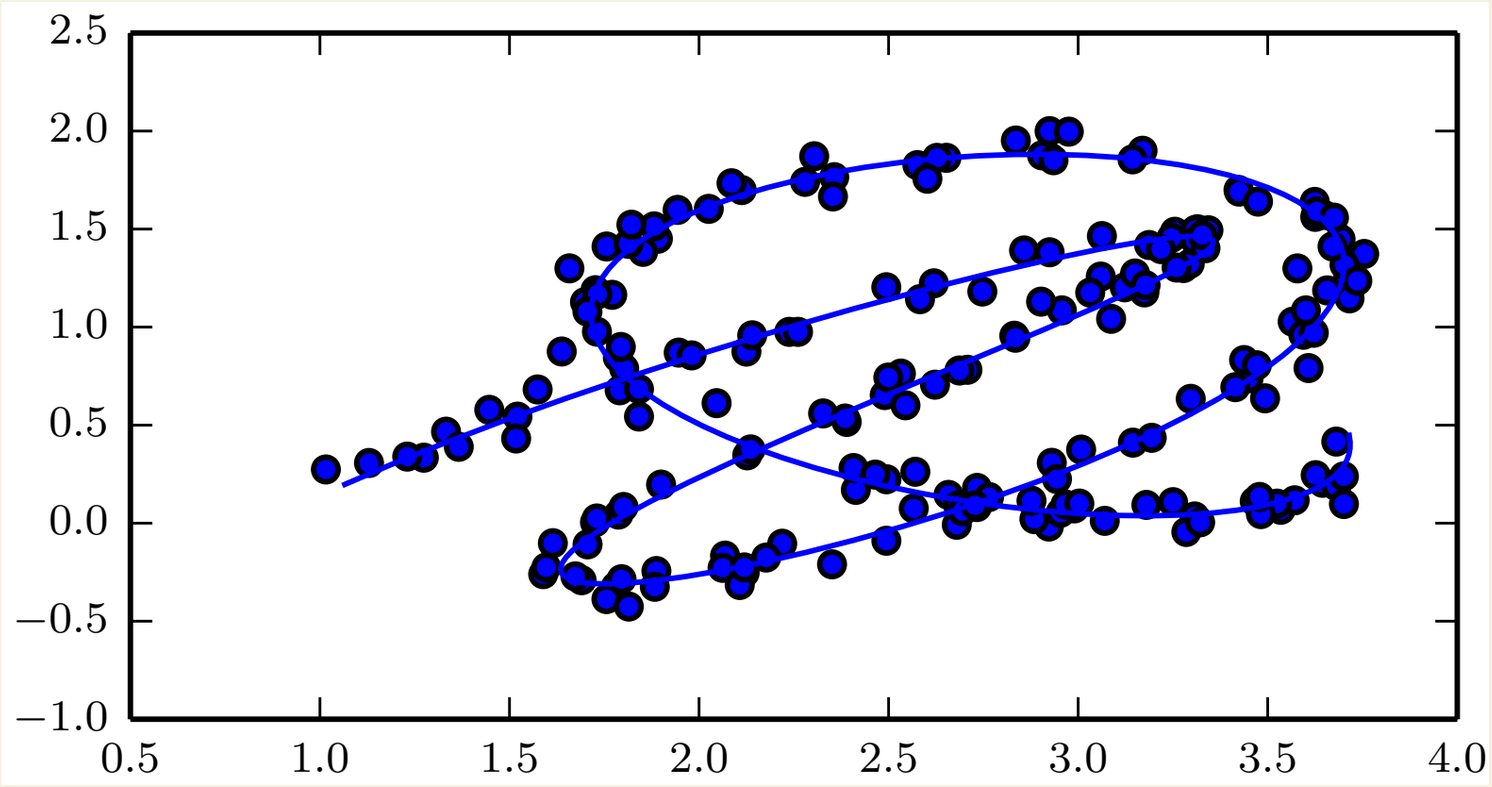
$$V_{hypersphere} = \frac{\pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} \left(\frac{L}{2}\right)^d$$

$\Gamma(n) = (n-1)!$ for n integer.

$$V_{hypercube} = L^d$$

$$\frac{V_{hypersphere}}{V_{hypercube}} = \frac{\pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} 0.5^d$$

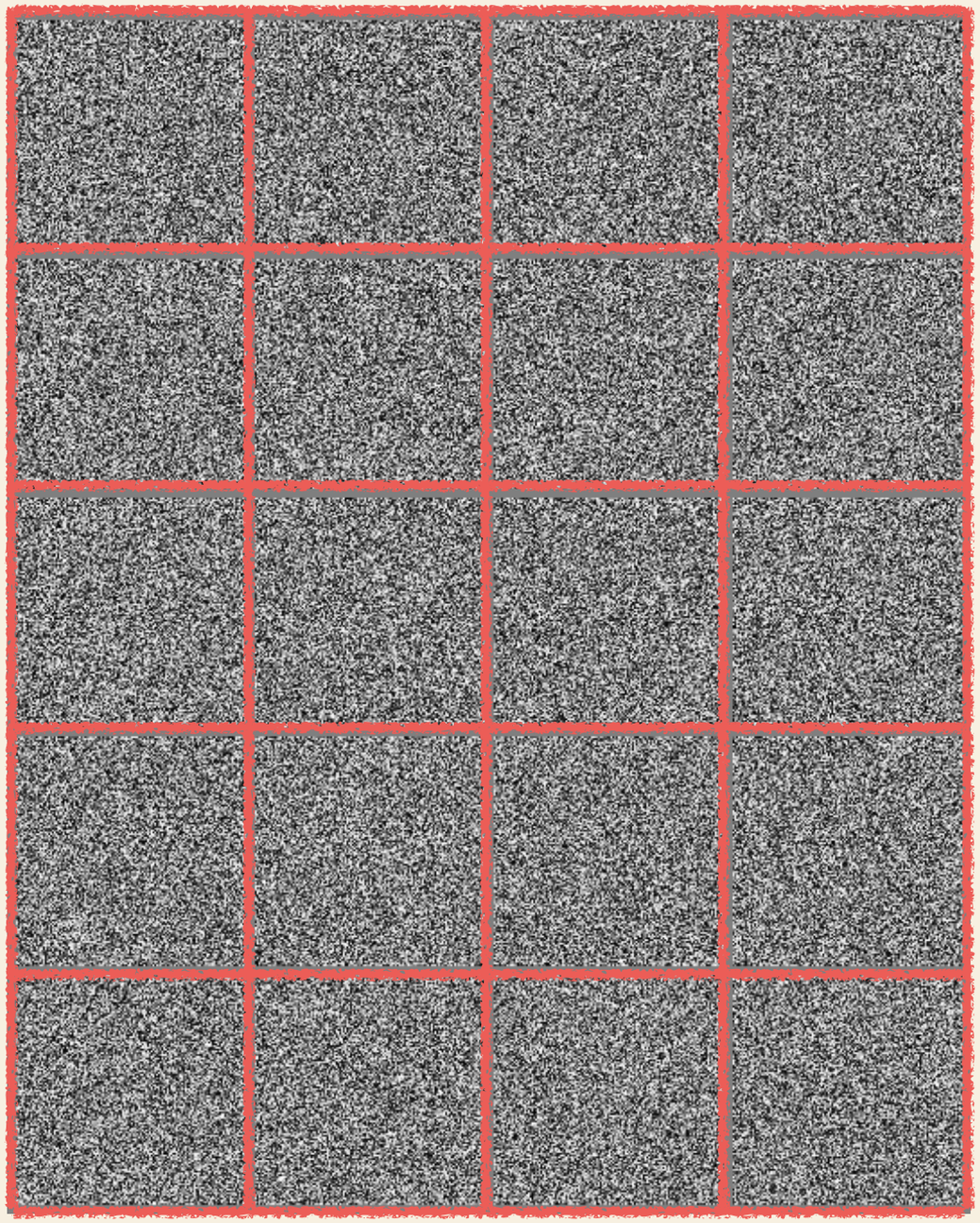
Manifold Learning



- **Manifold:** A ***connected*** set of points that can be approximated by considering a small number of dimensions, embedded in a higher dimensional space.
- **Manifold Hypothesis** is that the **probability distribution** over *images*, *text strings*, and *sounds etc* is highly **concentrated**. (if you pick letters randomly it's not likely to obtain a meaningful sentence).

Manifold Learning

- Sampling images uniformly at random gives rise to noisy images. The images encountered in AI applications occupy a *tiny* **manifold** of the image space.



Manifold Learning