MACHINE LEARNING FOR SIGNAL PROCESSING

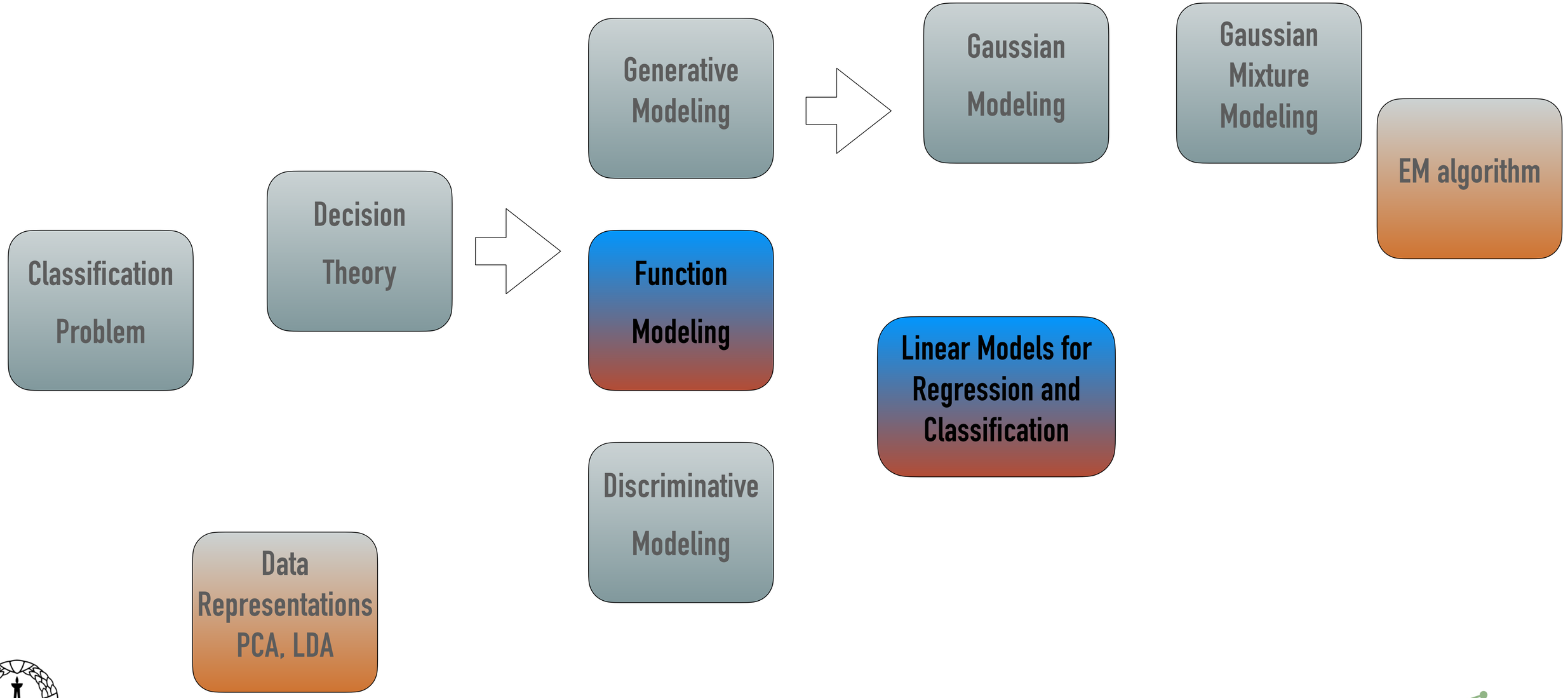
3-2-2025

Sriram Ganapathy
LEAP lab, Electrical Engineering, Indian Institute of Science,
sriramg@iisc.ac.in

Viveka Salinamakki, Varada R.
LEAP lab, Electrical Engineering, Indian Institute of Science

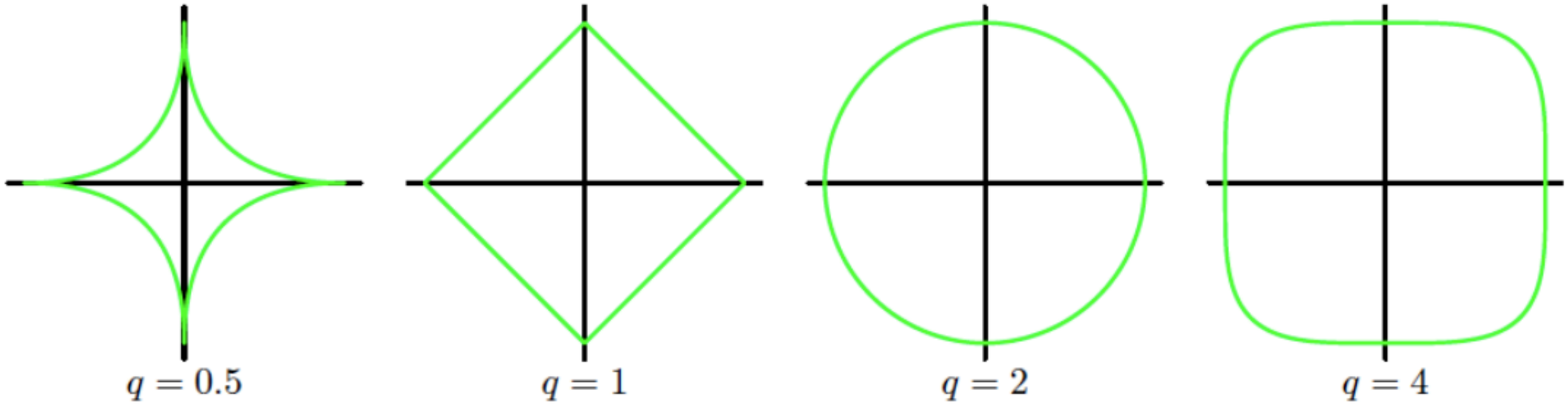
http://leap.ee.iisc.ac.in/sriram/teaching/MLSP25/

STORY SO FAR

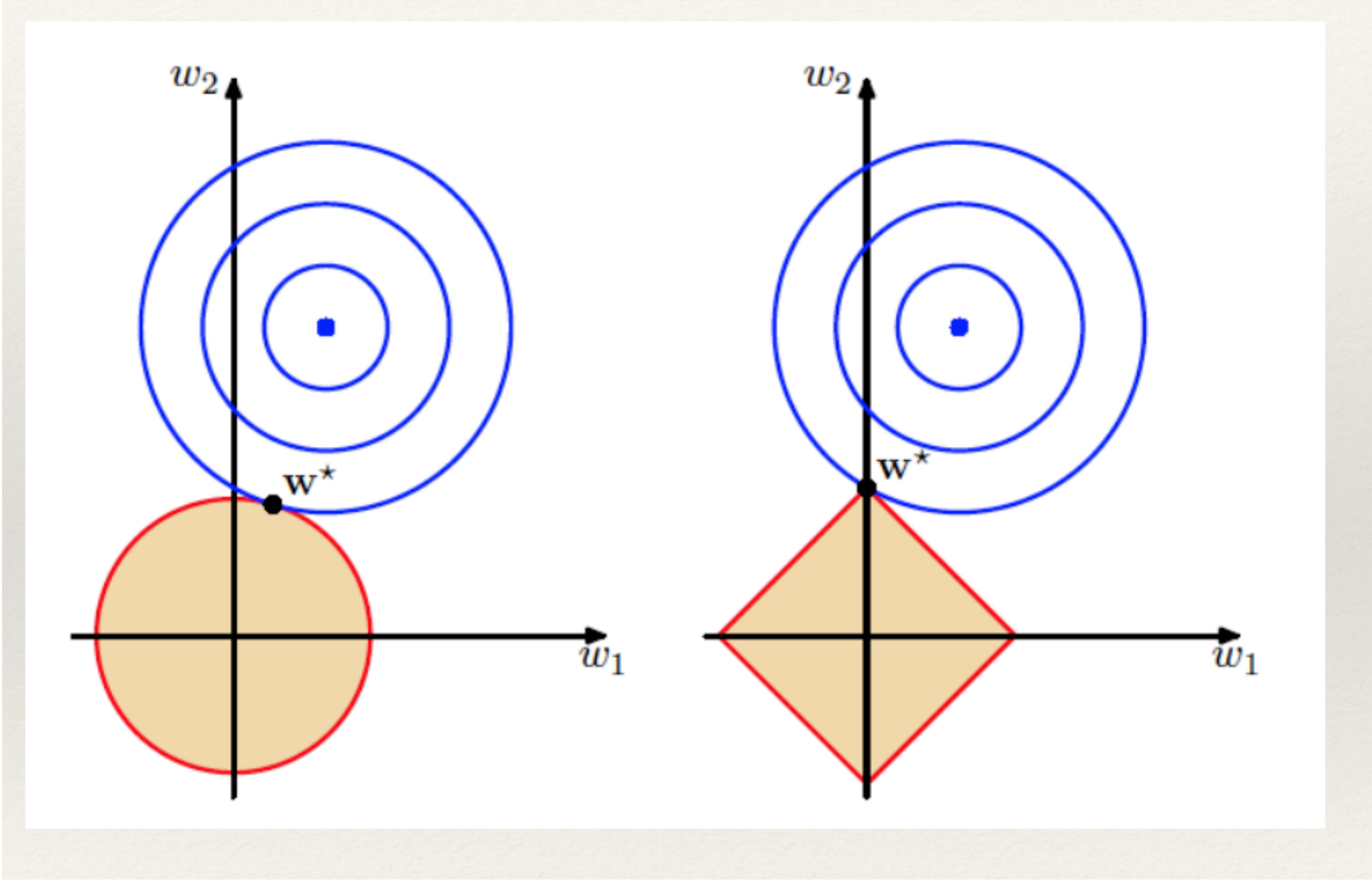

Classification Problem
Decision Theory
Generative Modeling
Gaussian Modeling
Gaussian Mixture Modeling
EM algorithm
Function Modeling
Linear Models for Regression and Classification
Discriminative Modeling
Data Representations PCA, LDA

REGULARIZED LINEAR REGRESSION

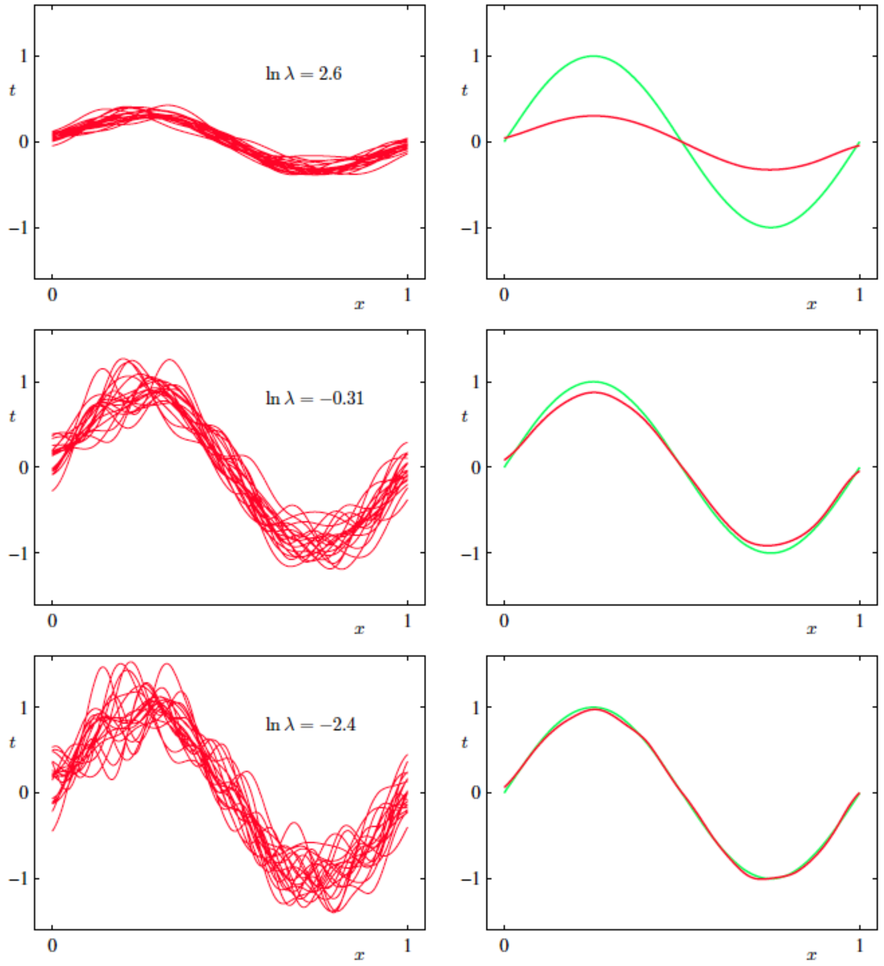
- Optimize a modified cost function $E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$



REGULARIZED LINEAR REGRESSION



Choice of Regularization Parameter



Bias Variance Tradeoff

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

where

$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

$$\text{variance} = \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

$$\text{noise} = \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x},t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t$$

Choice of Regularization Parameter

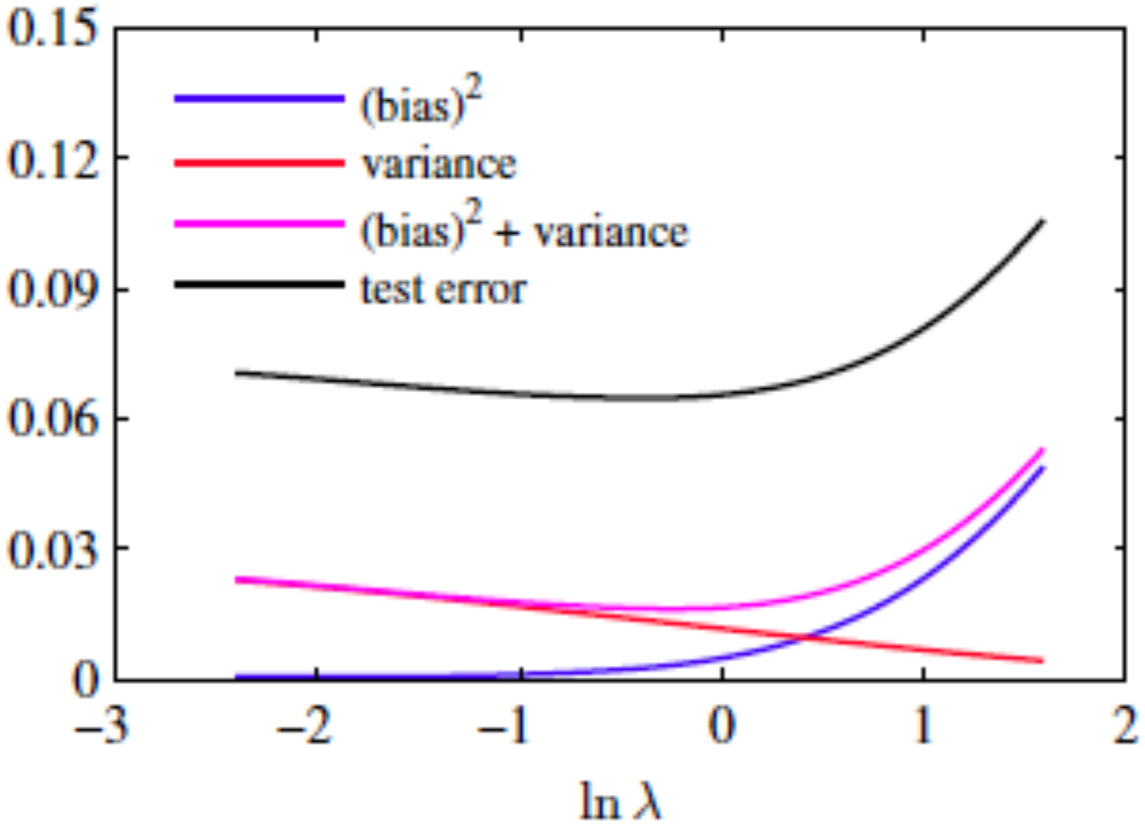
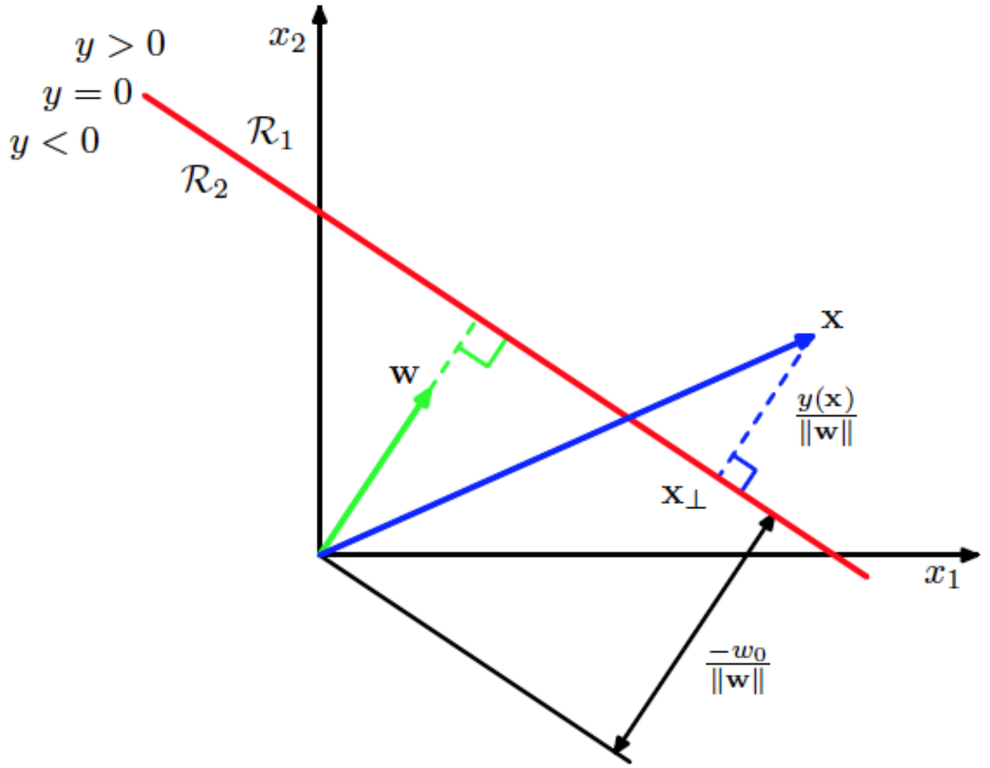


Figure 3.5 Illustration of the dependence of bias and variance on model complexity, governed by a regularization parameter λ, using the sinusoidal data set from Chapter 1. There are $L = 100$ data sets, each having $N = 25$ data points, and there are 24 Gaussian basis functions in the model so that the total number of parameters is $M = 25$ including the bias parameter. The left column shows the result of fitting the model to the data sets for various values of $\ln \lambda$ (for clarity, only 20 of the 100 fits are shown). The right column shows the corresponding average of the 100 fits (red) along with the sinusoidal function from which the data sets were generated (green).

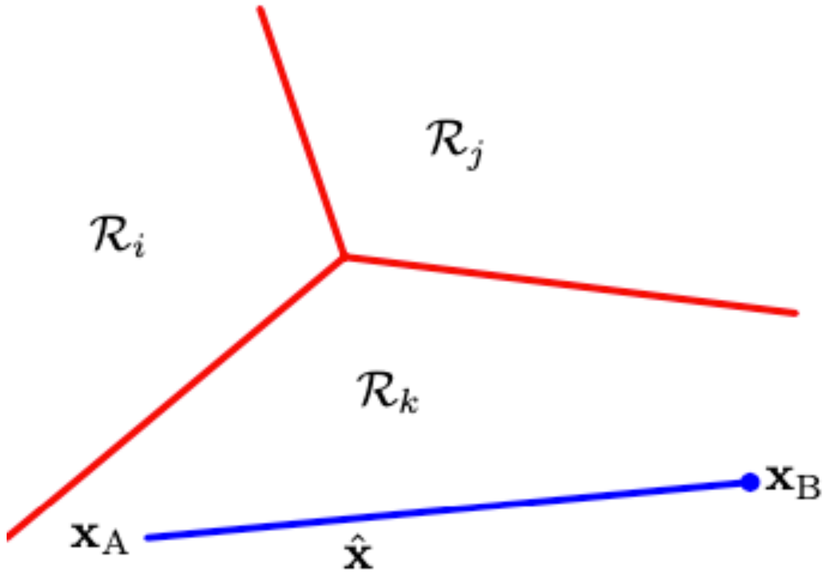
Linear Models for Classification

- Optimize a modified cost function

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$



LINEAR MODELS FOR CLASSIFICATION



Least Squares for Classification

- K-class classification problem

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$$

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}}$$

  - With 1-of-K hot encoding, and least squares regression

$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\left\{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^{\mathrm{T}}(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})\right\}$$

LOGISTIC REGRESSION

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

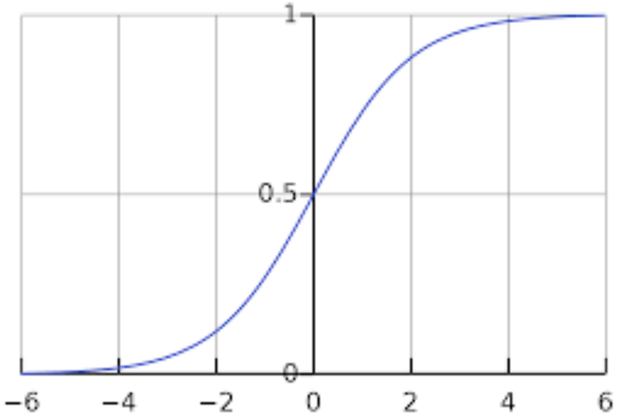
$$= \frac{1}{1+\exp(-a)} = \sigma(a)$$

where we have defined

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

and $\sigma(a)$ is the *logistic sigmoid* function defined by

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$



Logistic Regression

- 2- class logistic regression

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma\left(\mathbf{w}^{\mathrm{T}}\phi\right)$$

  - Maximum likelihood solution

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\phi_n$$

  - K-class logistic regression

$$p(\mathcal{C}_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

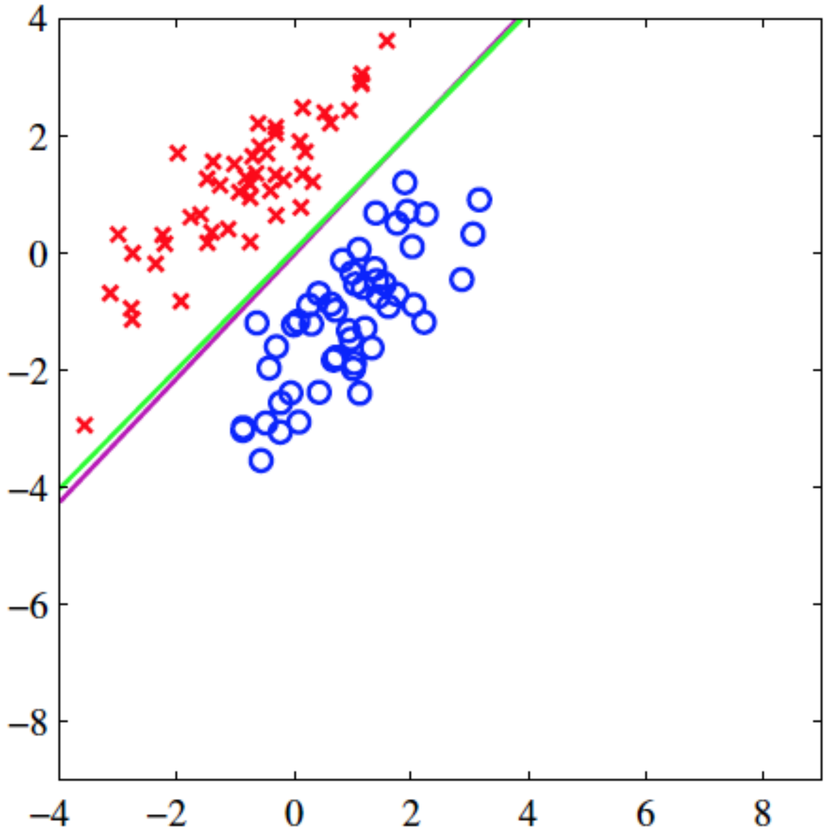
$$a_k = \mathbf{w}_k^{\mathrm{T}}\phi.$$

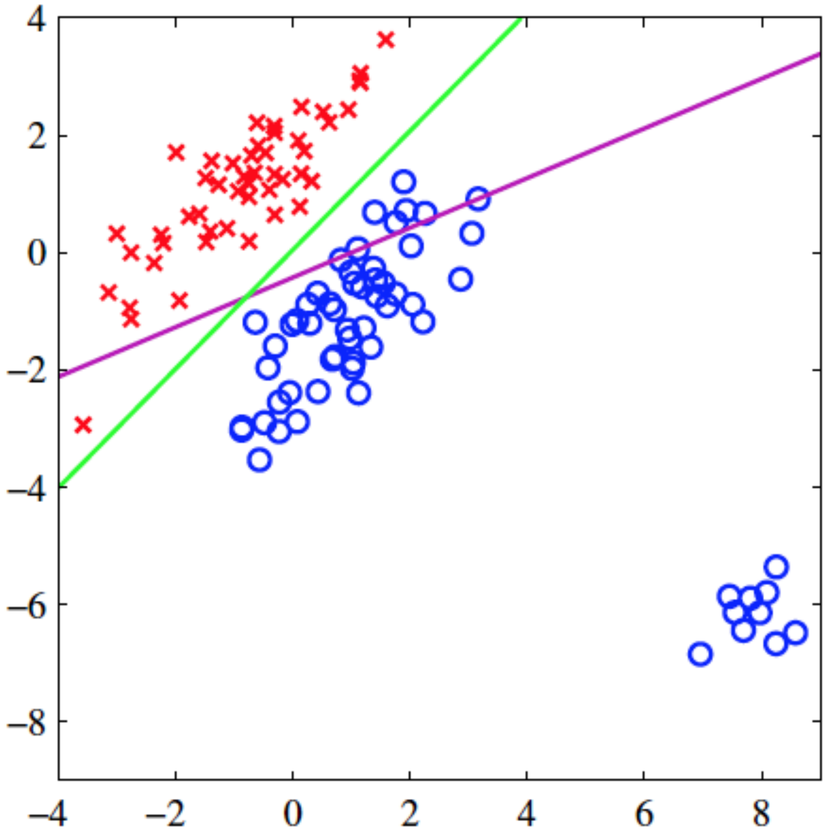
  - Maximum likelihood solution

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\,\phi_n$$

Least Squares versus Logistic Regression

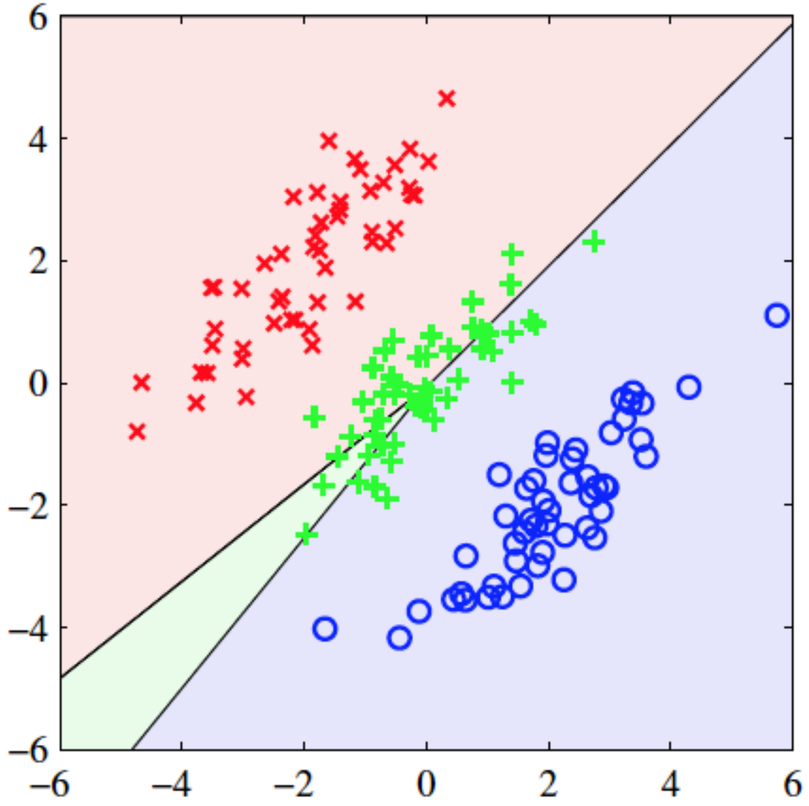
- 2- class logistic regression

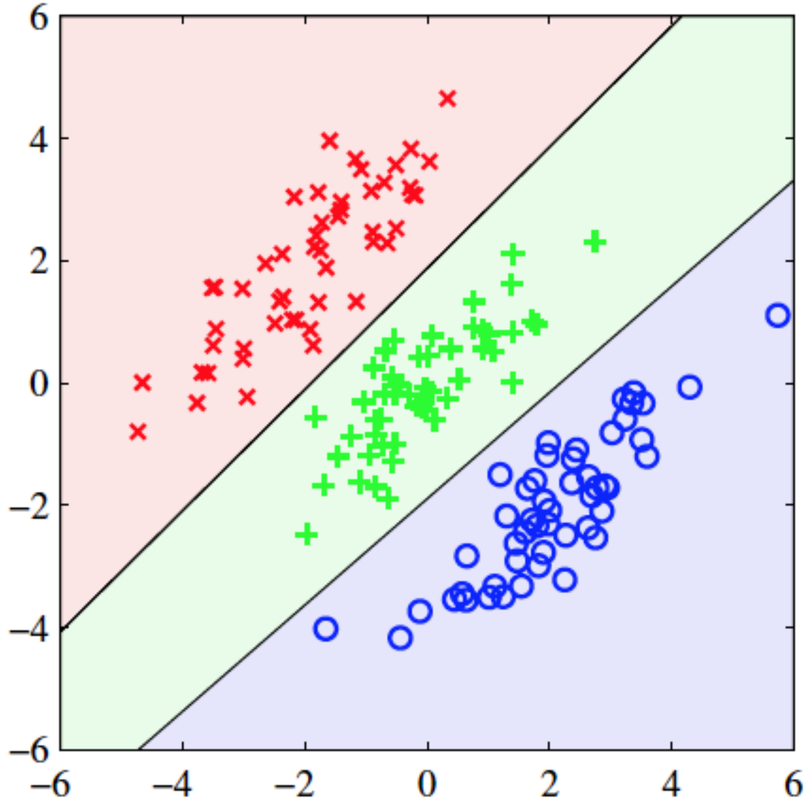




Least Squares versus Logistic Regression

- 2- class logistic regression

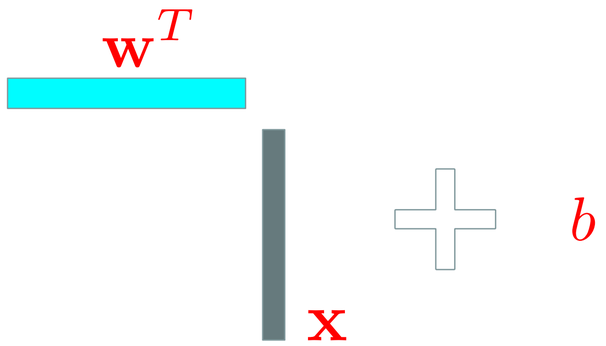




LETS START OFF WITH SIMPLEST MODEL

- Define the model as an affine transform

$$\mathbf{w}^T\mathbf{x} + b$$



$$w_1x_1 + w_2x_2 + w_3x_3 + ...w_Dx_D + b$$

- Convert the values to probabilities

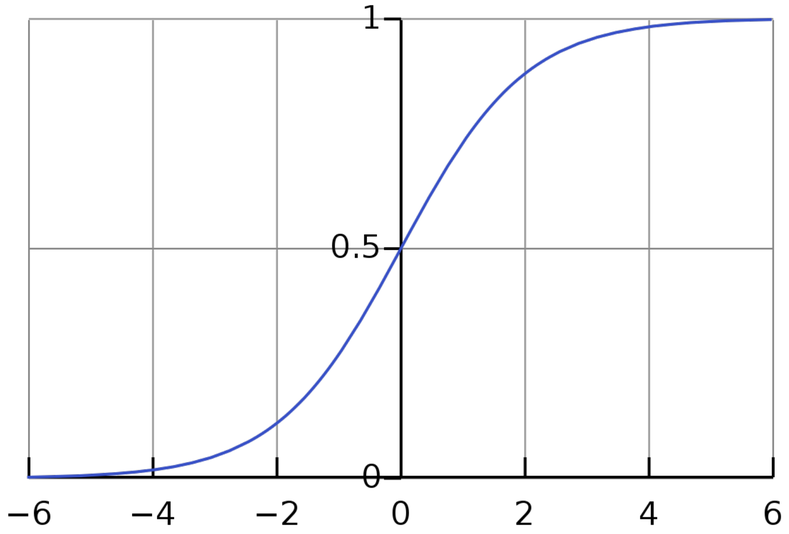
$$\mathbf{w}^T\mathbf{x} + b \geq 0 \rightarrow x \in C_1 \qquad p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$$

$$\mathbf{w}^T\mathbf{x} + b < 0 \rightarrow x \in C_2 \qquad p(C_2|\mathbf{x}) = 1 - p(C_1|\mathbf{x})$$

LETS START OFF WITH SIMPLEST MODEL

- Sigmoid function

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



- Convert the values to probabilities

$$\mathbf{w}^T\mathbf{x} + b \geq 0 \rightarrow x \in C_1 \qquad p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$$

$$\mathbf{w}^T\mathbf{x} + b < 0 \rightarrow x \in C_2 \qquad p(C_2|\mathbf{x}) = 1 - p(C_1|\mathbf{x})$$

HOW DO WE FIND THE MODEL PARAMETERS

- Given a set of training data

  - The targets $\{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_N\}$ $\quad t_n = 1 \quad for \quad \mathbf{x}_n \in \mathbf{C}_1$

    $\{t_1, t_2, ...t_N\}$ $\quad t_n = 0 \quad for \quad \mathbf{x}_n \in \mathbf{C}_2$

  - The model outputs $y_n = \sigma(\mathbf{w}^T \mathbf{x}_n + b)$
  - probability when $t_n = 1 \rightarrow y_n^{t_n}$
  - probability when $t_n = 0 \rightarrow (1 - y_n)^{(1-t_n)}$
  - Jointly $p(y_n) = y_n^{t_n}(1 - y_n)^{(1-t_n)}$

TOTAL PROBABILITY

- Each data sample is independent

$$p([y_1, y_2..., y_N]) = p(y_1)p(y_2)...p(y_N)$$

$$p([y_1, y_2..., y_N]) = p(y_1)p(y_2)...p(y_N)$$

$$= \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{(1-t_n)}$$

$$y_n = \sigma(\mathbf{w}^T \mathbf{x}_n + b)$$

- Log total probability $log\ p([y_1, y_2..., y_N]) = \sum_{n=1}^{N} t_n log\ y_n + (1 - t_n) log\ (1 - y_n)$

LEAP

HOW DO WE FIND THE MODEL PARAMETERS

- We want to maximize the total probability
  - Or equivalently minimize the negative log probability

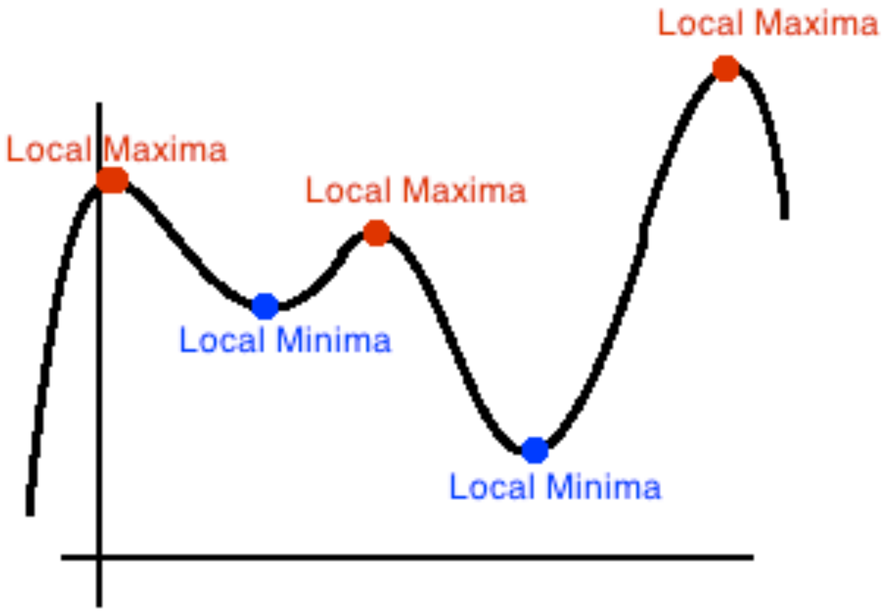
$$E(\mathbf{w}, b) = -\left[\sum_{n=1}^{N} t_n log\ y_n + (1 - t_n) log\ (1 - y_n)\right]$$

$$y_n = \sigma(\mathbf{w}^T \mathbf{x}_n + b)$$

- The parameters of the model need to solved based on the minimizing the error
  - This error function is non convex in the parameters
  - Need to perform iterative update —> Gradient descent

NON-CONVEX OPTIMIZATION

- Local maxima and minima



GRADIENT DESCENT ALGORITHM

- Start with random weights

$$iter \quad t = 0 \rightarrow \mathbf{w}^0, b^0$$

- Compute gradients and update the model iteratively

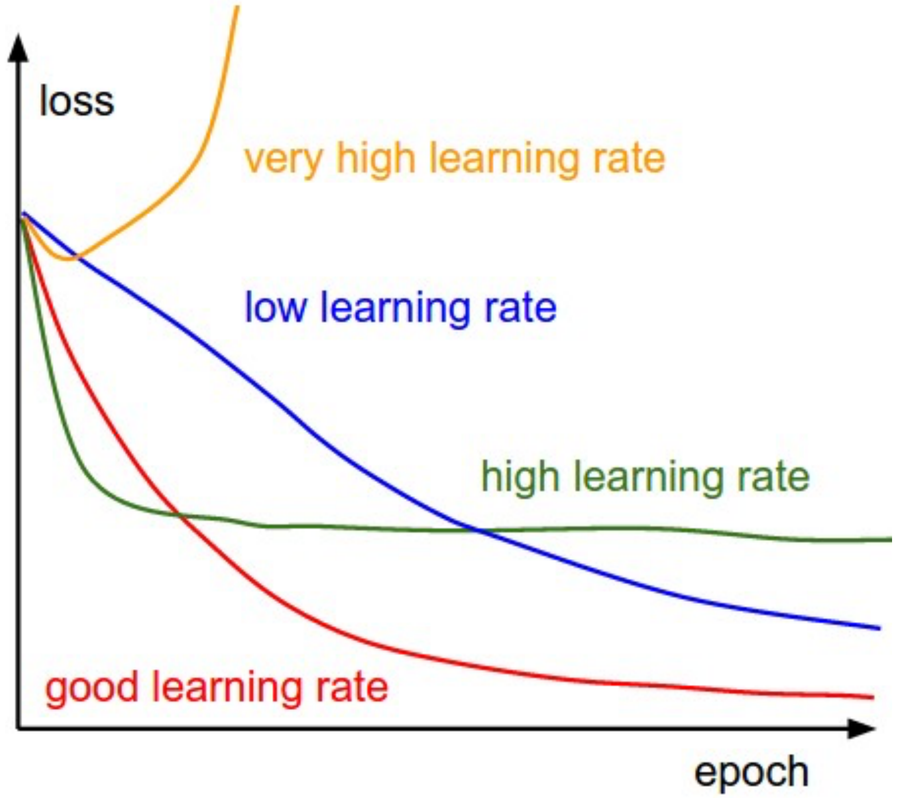
$$\mathbf{w}^t = \mathbf{w}^{t-1} - \eta \frac{\partial E}{\partial \mathbf{w}}\bigg|_{\mathbf{w}, b=\mathbf{w}^{t-1}, b^{t-1}}$$

$$b^t = b^{t-1} - \eta \frac{\partial E}{\partial b}\bigg|_{\mathbf{w}, b=\mathbf{w}^{t-1}, b^{t-1}}$$

- Check for convergence —> If not, $t = t + 1$ and go to the step above.

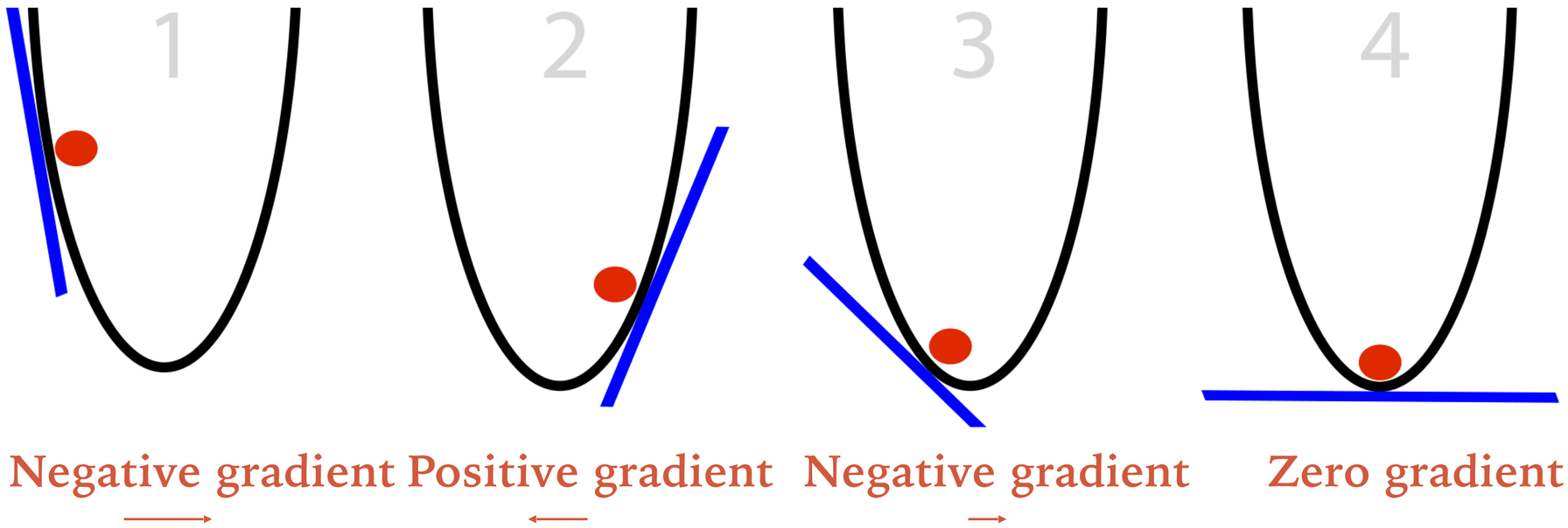
LEARNING RATE

- Solving a non-convex optimization.
- Iterative solution.
- Depends on the initialization.
- Convergence to a local optima.

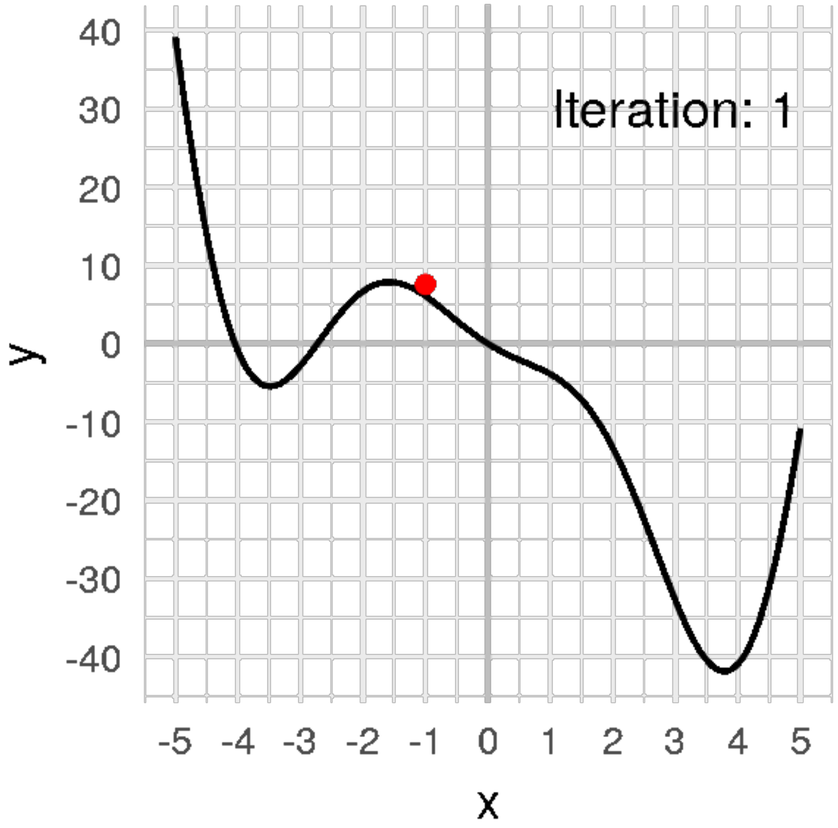


ERROR FUNCTION REVISITED

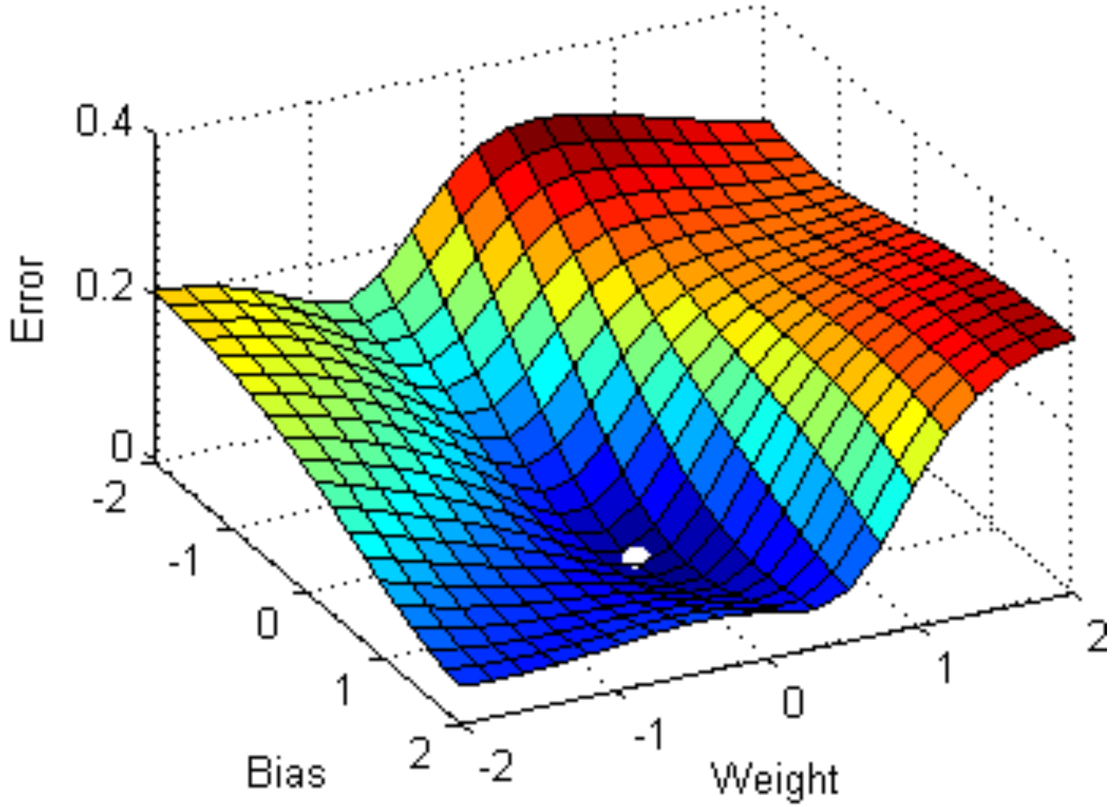
- Iterative solution is also possible.
  - ✓ Move the parameters in the negative direction of the gradient



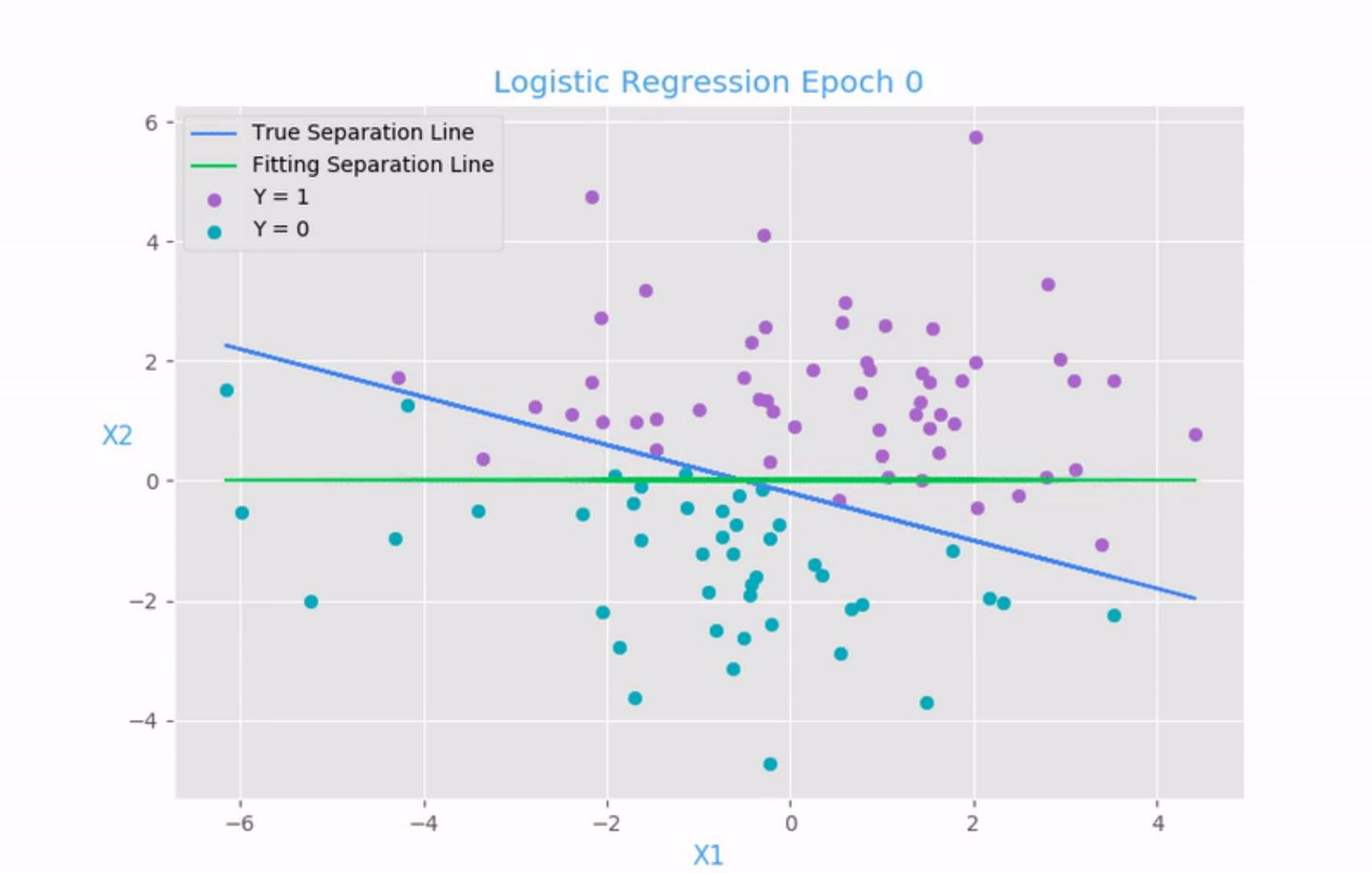
GRADIENT DESCENT ALGORITHM



GRADIENT DESCENT ALGORITHM



BACK TO OUR PROBLEM



MULTI-CLASS LOGISTIC REGRESSION

- Outputs are

$$\mathbf{y} = softmax(\mathbf{W}^T\mathbf{x} + \mathbf{b})$$

- Targets are one hot encoded vectors

$$\mathbf{t} \quad \begin{bmatrix} 0 \\ 0 \\ . \\ 0 \\ 1 \\ 0 \\ . \\ 0 \end{bmatrix}$$

- Error function

$$E(\mathbf{W}, \mathbf{b}) = -\sum_n \sum_k t_{nk} \; log \; y_{nk}$$

  - cross entropy function

- as before, learnt using gradient descent

STOCHASTIC GRADIENT DESCENT

- The total error is a sum of errors in all the data samples

$$E(\mathbf{w}, b) = -\left[\sum_{n=1}^{N} t_n log\ y_n + (1 - t_n) log\ (1 - y_n)\right]$$

  - May be too slow - requiring a large number of updates before convergence.

- Can we take a subset of the data and perform gradient descent and update the model on the subset ?

STOCHASTIC GRADIENT DESCENT

- Split the data into random chunks

- perform gradient updates on each mini batch
  - assumption - gradients computed on the mini-batch well approximate the full batch gradients

STOCHASTIC GRADIENT DESCENT

- Split the data into random chunks

- hyper-parameters
  - batch size

THANK YOU

Sriram Ganapathy and TA team
LEAP lab, C328, EE, IISc
sriramg@iisc.ac.in