

# **STA 414/2104: Machine Learning**

Russ Salakhutdinov

Department of Computer Science

Department of Statistics

rsalakhu@cs.toronto.edu

<http://www.cs.toronto.edu/~rsalakhu/>

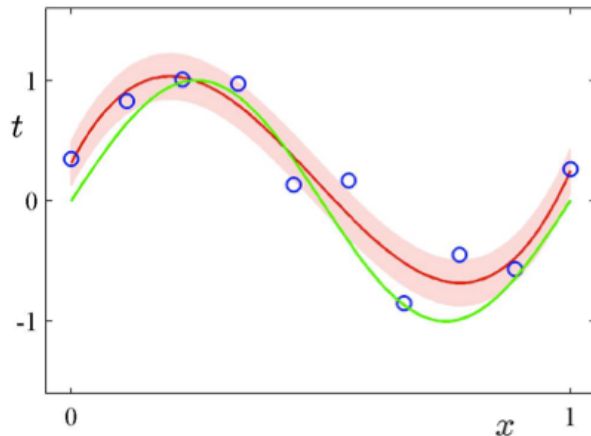
## **Lecture 3**

# Parametric Distributions

- We want model the probability distribution  $p(\mathbf{x}|\boldsymbol{\theta})$  of a random variable  $\mathbf{x}$  given a finite set of observations:  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$

Need to determine  $\boldsymbol{\theta}$  given  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$

- We will also assume that the data points are i.i.d
- We will focus on the maximum likelihood estimation  $\boldsymbol{\theta}^*$



- Remember curve fitting example.

$$p(t|\mathbf{x}, \mathbf{w}_{ML}, \beta_{ML}) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}_{ML}), \beta_{ML}^{-1}).$$

# Linear Basis Function Models

- Remember, the simplest linear model for regression:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1x_1 + w_2x_2 + \dots + w_dx_d = w_0 + \sum_{j=1}^d w_jx_j,$$

where  $\mathbf{x} = (x_1, x_2, \dots, x_d)^T$  a d-dimensional input vector (covariates).

**Key property:** linear function of the parameters  $w_0, w_1, \dots, w_d$ .

- However, it is also a linear function of input variables.

Instead consider:

$$y(\mathbf{x}, \mathbf{w}) = w_0\phi_0(\mathbf{x}) + w_1\phi_1(\mathbf{x}) + \dots + w_{M-1}\phi_{M-1}(\mathbf{x}) = \sum_{j=0}^{M-1} w_j\phi_j(\mathbf{x}),$$

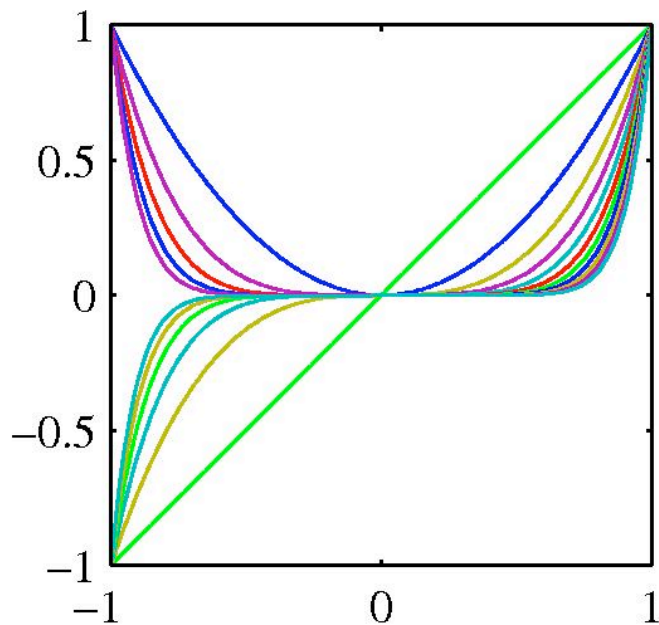
where  $\phi_j(\mathbf{x})$  are known as basis functions.

- Typically  $\phi_0(\mathbf{x}) = 1$  so that  $w_0$  acts as a bias (or intercept).
- In the simplest case, we use linear bases functions:  $\phi_j(\mathbf{x}) = x_j$ .
- Using nonlinear basis allows the functions  $y(\mathbf{x}, \mathbf{w})$  to be nonlinear functions of the input space.

# Linear Basis Function Models

Polynomial basis functions:

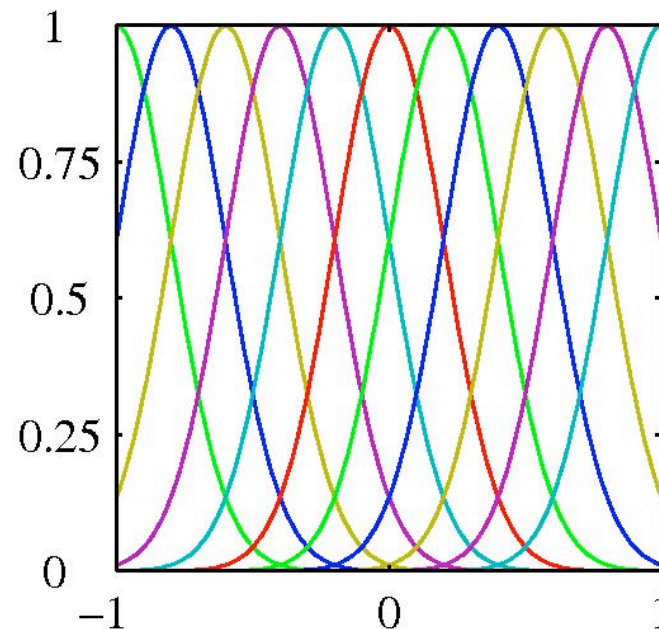
$$\phi_j(x) = x^j.$$



Basis functions are global: small changes in  $\mathbf{x}$  affect all basis functions.

Gaussian basis functions:

$$\phi_j(x) = \exp\left(-\frac{(x - \mu_j)^2}{2s^2}\right).$$

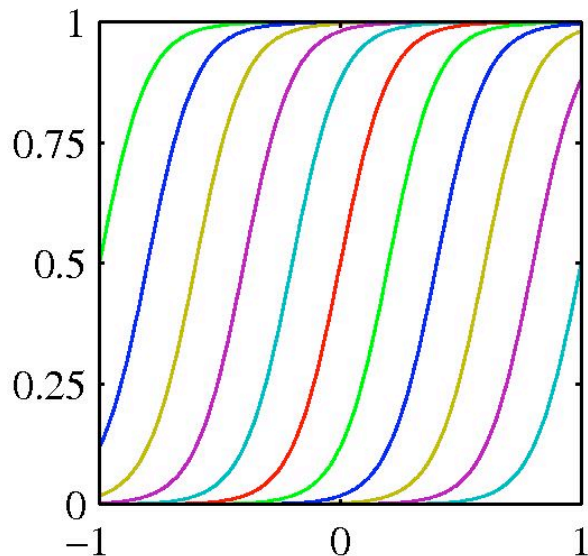


Basis functions are local: small changes in  $\mathbf{x}$  only affect nearby basis functions.  
 $\mu_j$  and  $s$  control location and scale (width).

# Linear Basis Function Models

# Sigmoidal basis functions

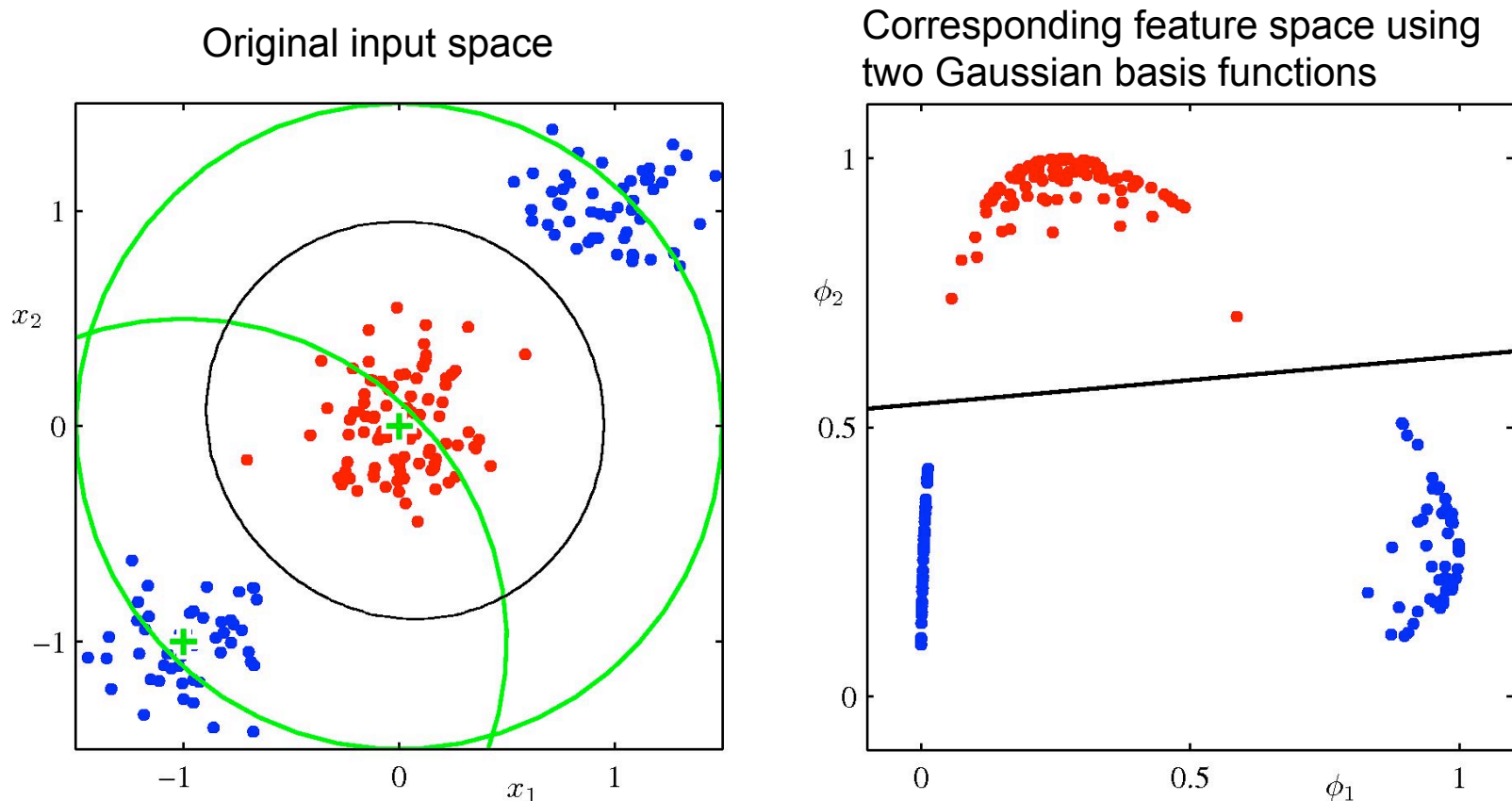
$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right), \text{ where } \sigma(a) = \frac{1}{1 + \exp(-a)}.$$



Basis functions are local: small changes in  $\mathbf{x}$  only affect nearby basis functions.  $\mu_j$  and  $s$  control location and scale (slope).

- Decision boundaries will be linear in the feature space  $\phi$ , but would correspond to nonlinear boundaries in the original input space  $x$ .
- Classes that are linearly separable in the feature space  $\phi(x)$  need not be linearly separable in the original input space.

# Linear Basis Function Models



- We define two Gaussian basis functions with centers shown by the green crosses, and with contours shown by the green circles.
- Linear decision boundary (right) is obtained by using logistic regression, and corresponds to the nonlinear decision boundary in the input space (left, black curve).

# Maximum Likelihood

- As before, assume observations arise from a deterministic function with an additive Gaussian noise:

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon,$$

which we can write as:

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}).$$

- Given observed inputs  $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ , and corresponding target values  $\mathbf{t} = [t_1, t_2, \dots, t_N]^T$  under i.i.d assumption, we can write down the likelihood function:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^N \mathcal{N}(t_n|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta),$$

where  $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \dots, \phi_{M-1}(\mathbf{x}))^T$ .

# Maximum Likelihood

Taking the logarithm, we obtain:

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) &= \sum_{i=1}^N \ln \mathcal{N}(t_n | \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta) \\ &= -\frac{\beta}{2} \underbrace{\sum_{n=1}^N (t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n))^2}_{\text{sum-of-squares error function}} + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi).\end{aligned}$$

Differentiating and setting to zero yields:

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{w}, \beta) = \beta \sum_{n=1}^N \{t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)\} \boldsymbol{\phi}(\mathbf{x}_n)^T = \mathbf{0}.$$



# Maximum Likelihood

Differentiating and setting to zero yields:

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{w}, \beta) = \beta \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\} \phi(\mathbf{x}_n)^T = \mathbf{0}.$$

Solving for  $\mathbf{w}$ , we get:

$$\mathbf{w}_{\text{ML}} = \left( \Phi^T \Phi \right)^{-1} \Phi^T \mathbf{t}$$

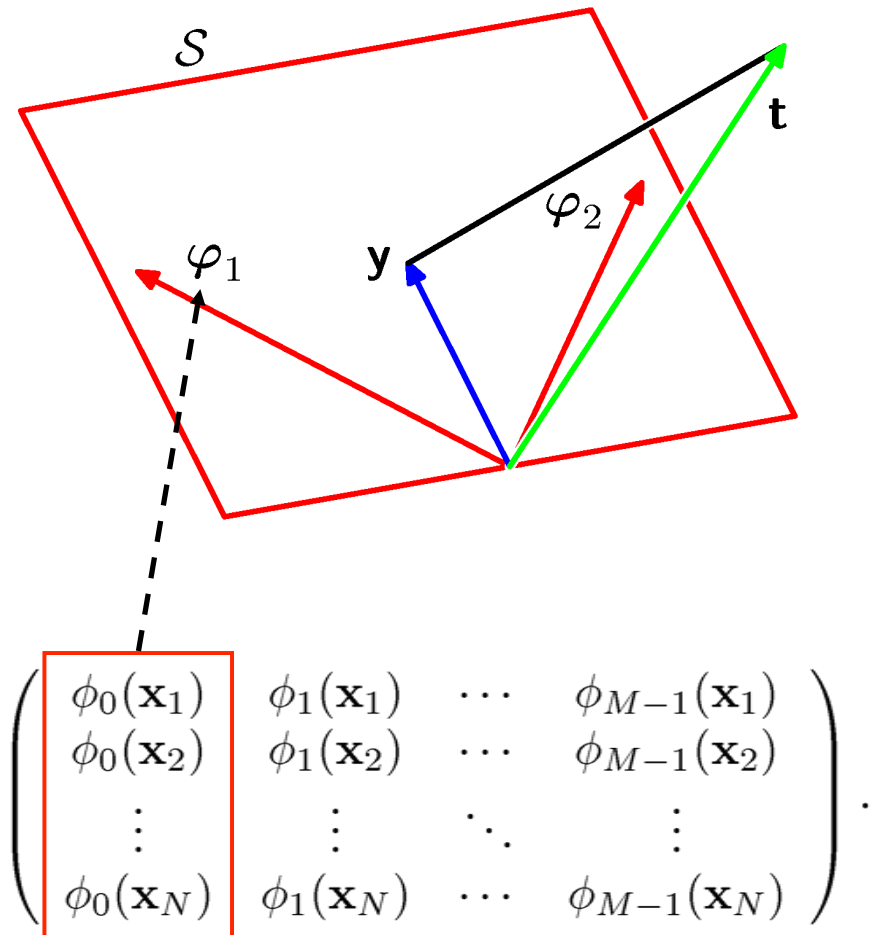
The Moore-Penrose pseudo-inverse,  $\Phi^\dagger$ .

where  $\Phi$  is known as the **design matrix**:

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}.$$

# Geometry of Least Squares

- Consider an N-dimensional space, so that  $\mathbf{t} = [t_1, t_2, \dots, t_N]^T$  is a vector in that space.
- Each basis function  $\phi_j(\mathbf{x}_n)$ , evaluated at the N data points, can be represented as a vector in the same space.
- If M is less than N, then the M basis function  $\phi_j(\mathbf{x}_n)$ , will span a linear subspace S of dimensionality M.
- Define:  $\mathbf{y} = \Phi \mathbf{w}_{\text{ML}}$ .
- The sum-of-squares error is equal to the squared Euclidean distance between  $\mathbf{y}$  and  $\mathbf{t}$  (up to a factor of 1/2).



The solution corresponds to the orthogonal projection of  $\mathbf{t}$  onto the subspace S.

# Sequential Learning

- The training data examples are presented one at a time, and the model parameter are updated after each such presentation (online learning):

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla E_n$$

Diagram illustrating the weight update equation:

- $\mathbf{w}^{(t+1)}$ : weights after seeing training case  $t+1$  (indicated by a blue arrow)
- $\mathbf{w}^{(t)}$ : weights before seeing training case  $t+1$  (indicated by a blue arrow)
- $\eta$ : learning rate (indicated by a blue arrow)
- $\nabla E_n$ : vector of derivatives of the squared error w.r.t. the weights on the training case presented at time  $t$ . (indicated by a red arrow)

- For the case of sum-of-squares error function, we obtain:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \left( t_n - \mathbf{w}^{(t)T} \phi(\mathbf{x}_n) \right) \phi(\mathbf{x}_n).$$

- **Stochastic gradient descent**: if the training examples are picked at random (dominant technique when learning with very large datasets).
- Care must be taken when choosing learning rate to ensure convergence.

# Regularized Least Squares

- Let us consider the following error function:

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Data term + Regularization term

$\lambda$  is called the regularization coefficient.

- Using sum-of-squares error function with a quadratic penalization term, we obtain:

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$

which is minimized by setting:

$$\mathbf{w} = \left( \lambda \mathbf{I} + \Phi^T \Phi \right)^{-1} \Phi^T \mathbf{t}.$$

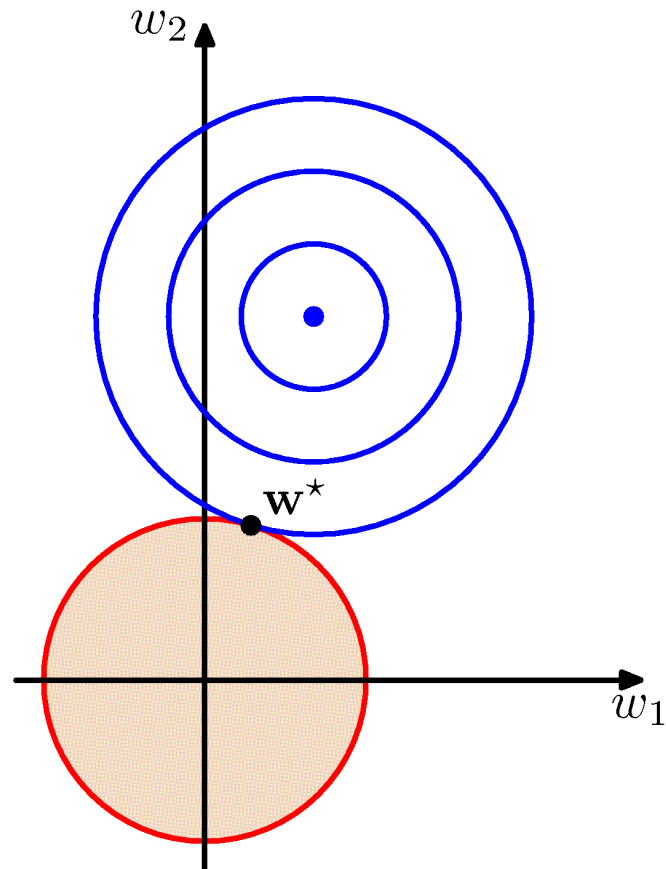
Ridge regression



The solution adds a positive constant to the diagonal of  $\Phi^T \Phi$ . This makes the problem nonsingular, even if  $\Phi^T \Phi$  is not of full rank (e.g. when the number of training examples is less than the number of basis functions).

# Effect of Regularization

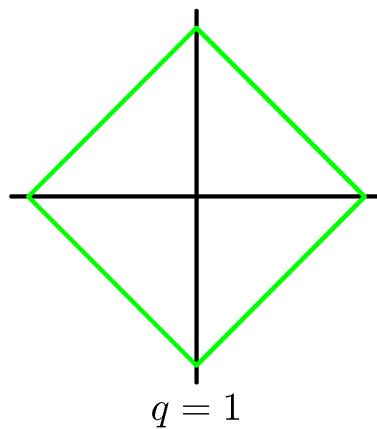
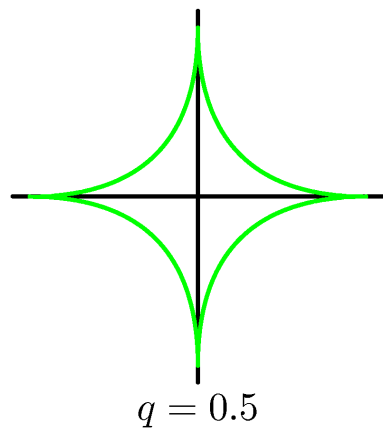
- The overall error function is the sum of two parabolic bowls.
- The combined minimum lies on the line between the minimum of the squared error and the origin.
- The regularizer shrinks model parameters to zero.



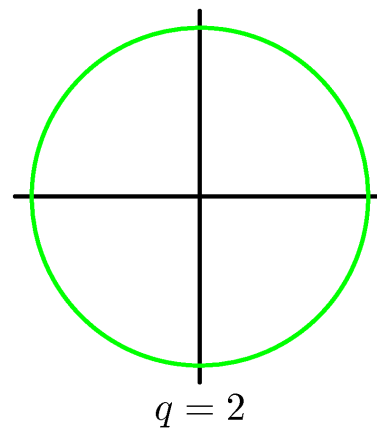
# Other Regularizers

Using a more general regularizer, we get:

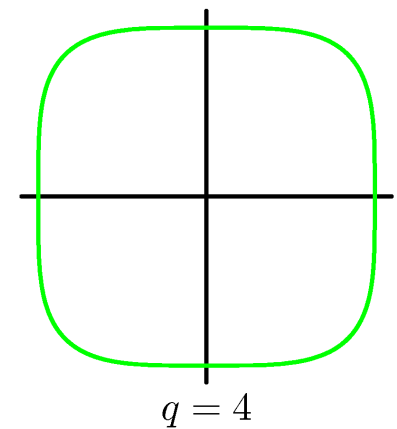
$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q$$



Lasso



Quadratic



# The Lasso

- Penalize the absolute value of the weights:

$$\mathbf{w}^{lasso} = \underset{\mathbf{w}}{\operatorname{argmin}} \left[ \frac{1}{2} \sum_{n=1}^N (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 + \frac{\lambda}{2} \sum_{j=1}^{M-1} |w_j| \right].$$

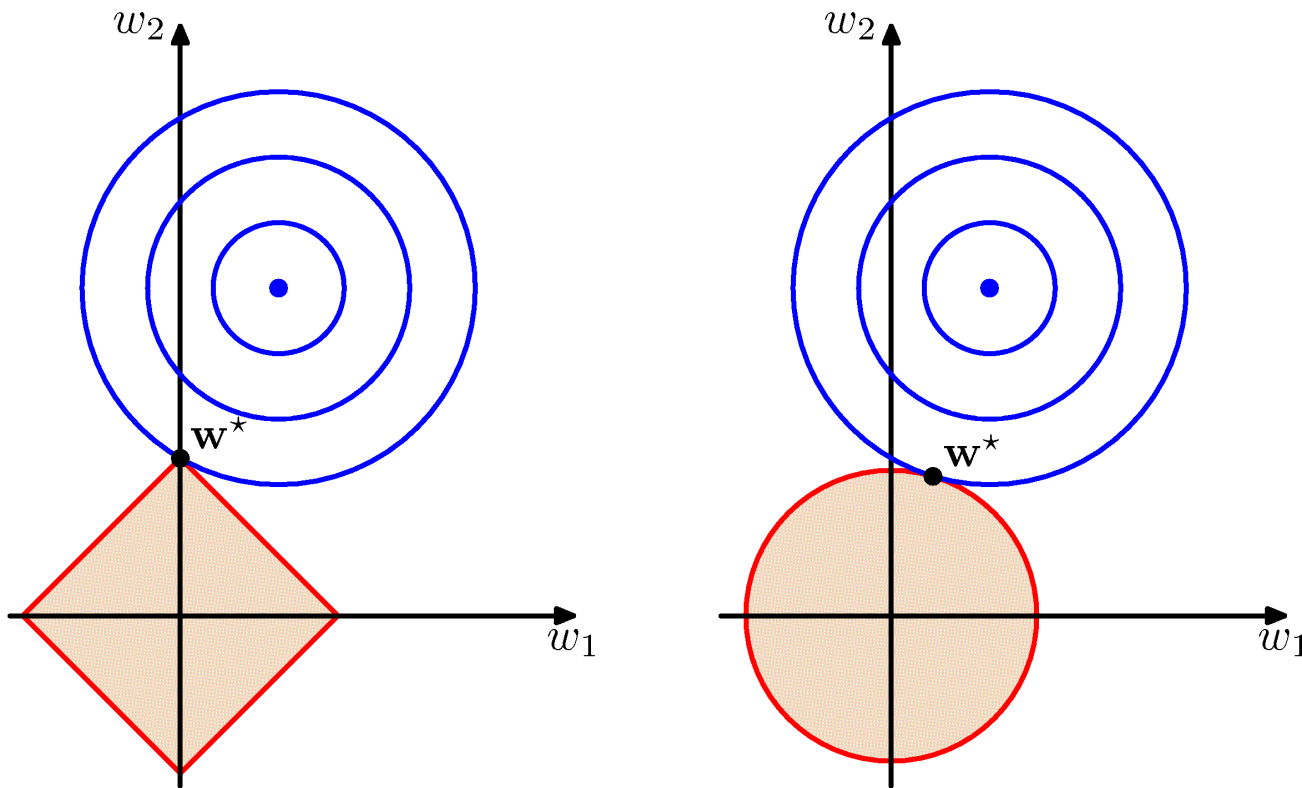
- For sufficiently large  $\lambda$ , some of the coefficients will be driven to exactly zero, leading to a sparse model.
- The above formulation is equivalent to:

$$\mathbf{w}^{lasso} = \underset{\mathbf{w}}{\operatorname{argmin}} \underbrace{\frac{1}{2} \sum_{n=1}^N (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2}_{\text{unregularized sum-of-squares error}}, \text{ subject to } \sum_{j=1}^{M-1} |w_j| \leq \tau.$$

- The two approaches are related using Lagrange multipliers.
- The Lasso solution is a quadratic programming problem: can be solved efficiently.

# Lasso vs. Quadratic Penalty

Lasso tends to generate sparser solutions compared to a quadratic regularizer (sometimes called  $L_1$  and  $L_2$  regularizers).





# Statistical Decision Theory

- We now develop a small amount of theory that provides a framework for developing many of the models we consider.
- Suppose we have a real-valued input vector  $\mathbf{x}$  and a corresponding target (output) value  $t$  with joint probability distribution:  $p(\mathbf{x}, t)$ .
- Our goal is predict target  $t$  given a new value for  $\mathbf{x}$ :
  - for regression:  $t$  is a real-valued continuous target.
  - for classification:  $t$  a categorical variable representing class labels.

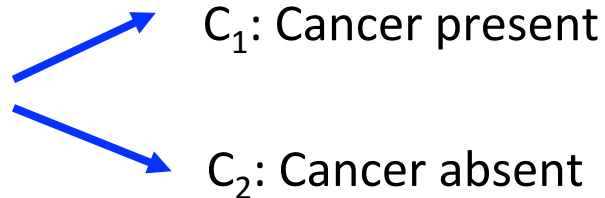
The joint probability distribution  $p(\mathbf{x}, t)$  provides a complete summary of uncertainties associated with these random variables.

Determining  $p(\mathbf{x}, t)$  from training data is known as the **inference problem**.

# Example: Classification

**Medical diagnosis:** Based on the X-ray image, we would like determine whether the patient has cancer or not.

- The input vector  $\mathbf{x}$  is the set of pixel intensities, and the output variable  $t$  will represent the presence of cancer, class  $C_1$ , or absence of cancer, class  $C_2$ .



$\mathbf{x}$  -- set of pixel intensities

- Choose  $t$  to be binary:  $t=0$  correspond to class  $C_1$ , and  $t=1$  corresponds to  $C_2$ .

**Inference Problem:** Determine the joint distribution  $p(\mathbf{x}, \mathcal{C}_k)$  or equivalently  $p(\mathbf{x}, t)$ . However, in the end, we must **make a decision** of whether to give treatment to the patient or not.

# Example: Classification

**Informally:** Given a new X-ray image, our goal is to decide which of the two classes that image should be assigned to.

- We could compute conditional probabilities of the two classes, given the input image:

Diagram illustrating the components of Bayes' Rule:

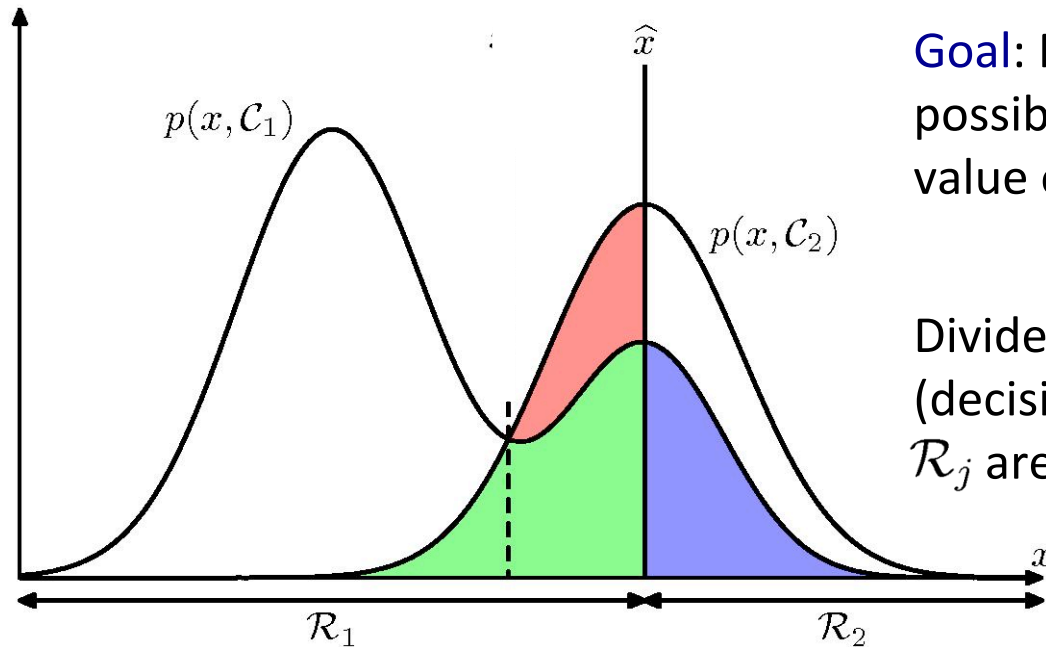
- posterior probability of  $C_k$  given observed data. (points to  $p(C_k|\mathbf{x})$ )
- probability of observed data given  $C_k$  (points to  $p(\mathbf{x}|C_k)$ )
- prior probability for class  $C_k$  (points to  $p(C_k)$ )

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}, C_k)}{\sum_{k=1}^K p(\mathbf{x}, C_k)} = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})}$$

Bayes' Rule (points to the entire equation)

- If our goal to minimize the probability of assigning  $\mathbf{x}$  to the wrong class, then we should choose the class having the highest posterior probability.

# Minimizing Misclassification Rate



**Goal:** Make as few misclassifications as possible. We need a rule that assigns each value of  $\mathbf{x}$  to one of the available classes.

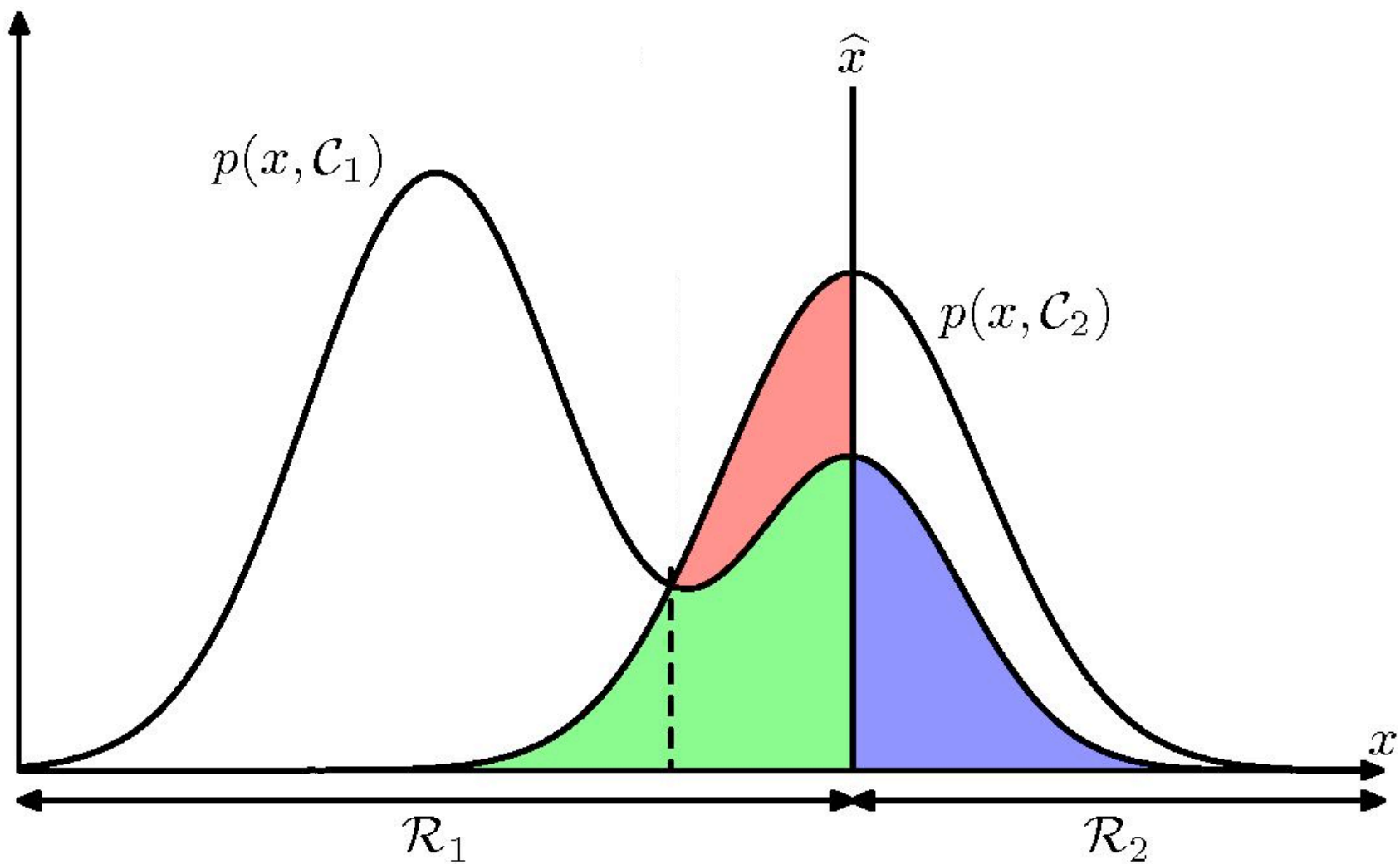
Divide the input space into regions  $\mathcal{R}_j$  (decision regions), such that all points in  $\mathcal{R}_j$  are assigned to class  $\mathcal{C}_j$ .

red+green regions: input belongs to class  $\mathcal{C}_2$ , but is assigned to  $\mathcal{C}_1$

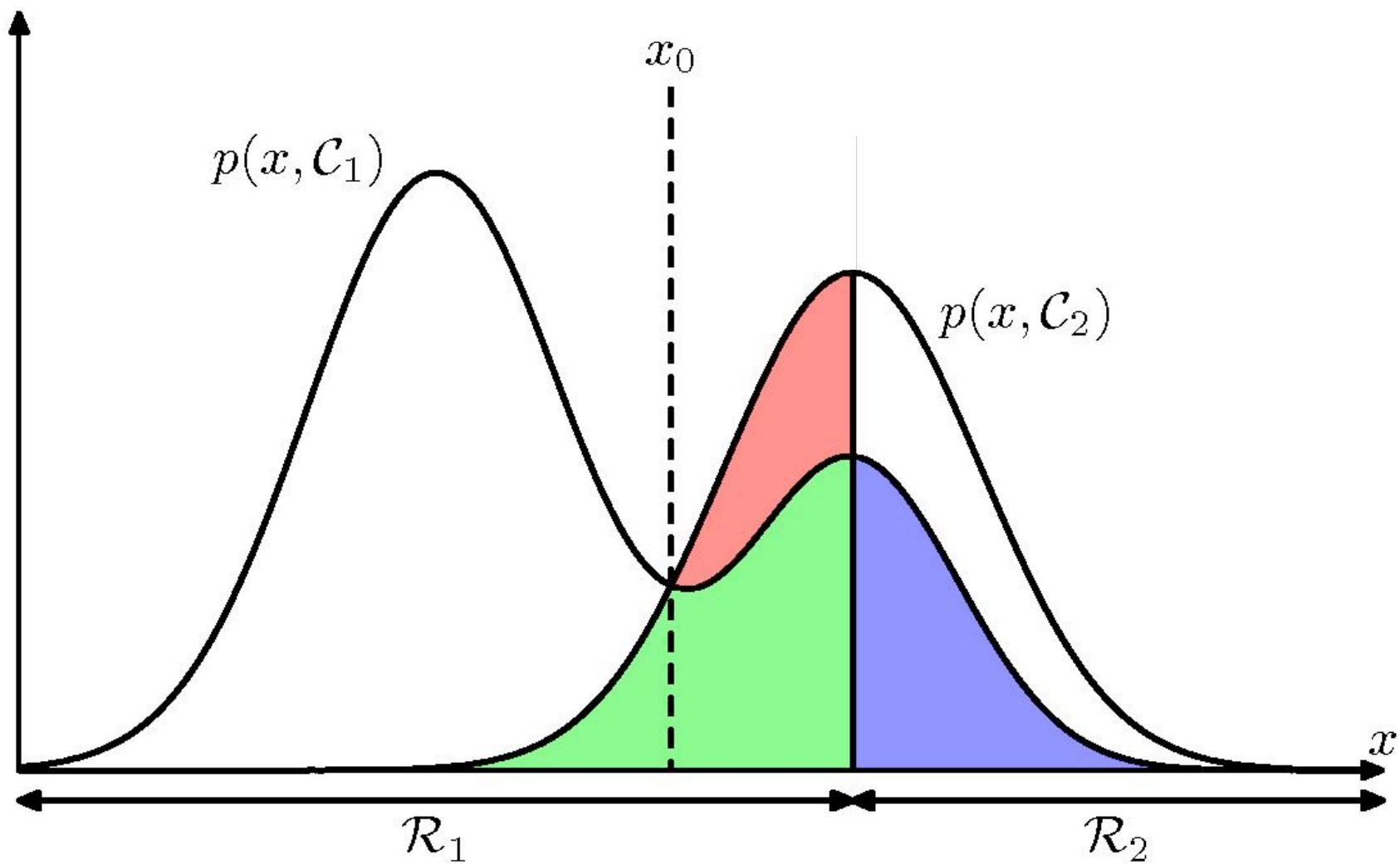
blue region: input belongs to class  $\mathcal{C}_1$ , but is assigned to  $\mathcal{C}_2$

$$\begin{aligned} p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\ &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}. \end{aligned}$$

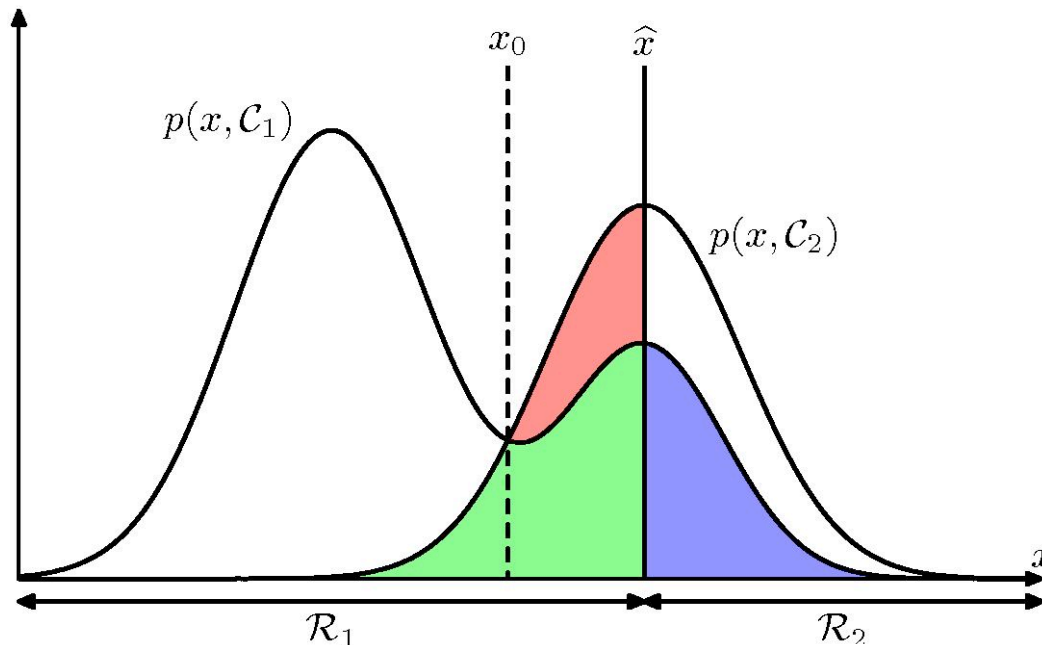
# Minimizing Misclassification Rate



# Minimizing Misclassification Rate



# Minimizing Misclassification Rate



$$p(\text{mistake}) = p(\mathbf{x} \in \mathcal{R}_1, C_2) + p(\mathbf{x} \in \mathcal{R}_2, C_1) = \int_{\mathcal{R}_1} p(\mathbf{x}, C_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, C_1) d\mathbf{x}$$

if  $p(\mathbf{x}, C_1) > p(\mathbf{x}, C_2)$  then we should assign  $\mathbf{x}$  to class  $C_1$ .

Using  $p(\mathbf{x}, C_k) = p(C_k|\mathbf{x})p(\mathbf{x})$  : To minimize the probability of making mistake, we assign each  $\mathbf{x}$  to the class for which the posterior probability  $p(C_k|\mathbf{x})$  is largest.

# Expected Loss

- **Loss Function**: overall measure of loss incurred by taking any of the available decisions.
- Suppose that for  $\mathbf{x}$ , the true class is  $C_k$ , but we assign  $\mathbf{x}$  to class  $j$   
→ incur loss of  $L_{kj}$  ( $k,j$  element of a loss matrix).

Consider medical diagnosis example: example of a loss matrix:

		Decision	
		cancer	normal
Truth	cancer	0	1000
	normal	1	0

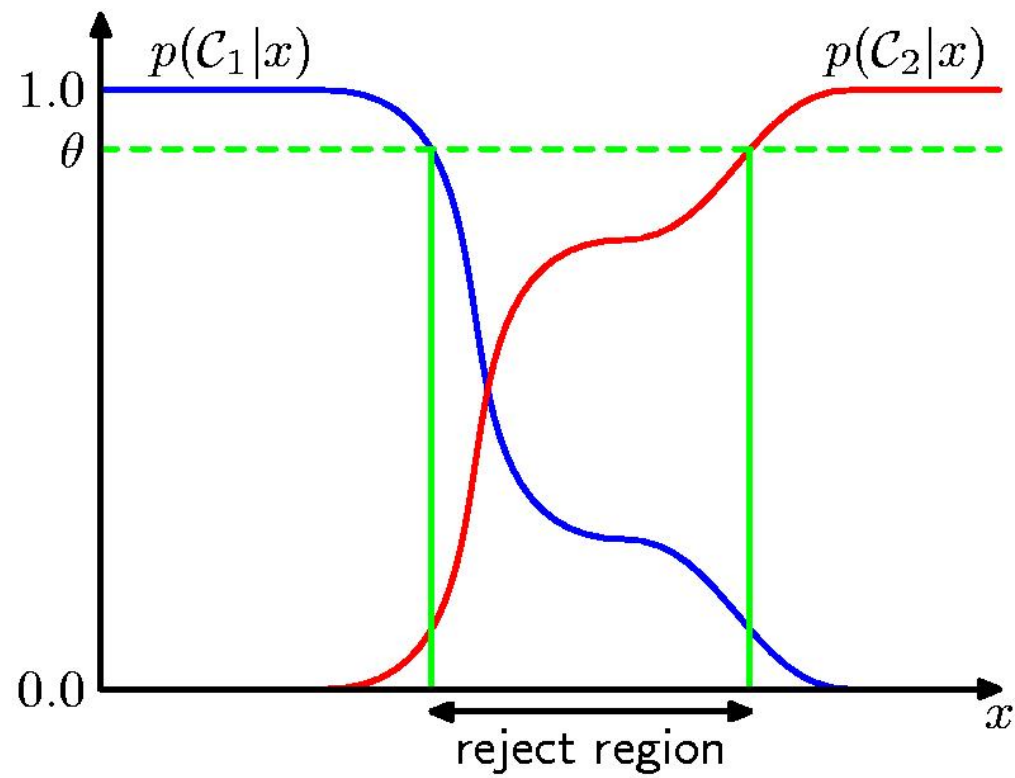
Expected Loss:

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, C_k) d\mathbf{x}$$

Goal is to choose regions  $\mathcal{R}_j$  as to minimize expected loss.



# Reject Option



# Regression

Let  $\mathbf{x} \in \mathbb{R}^d$  denote a real-valued input vector, and  $t \in \mathbb{R}$  denote a real-valued random target (output) variable with joint the distribution  $p(\mathbf{x}, t)$ .

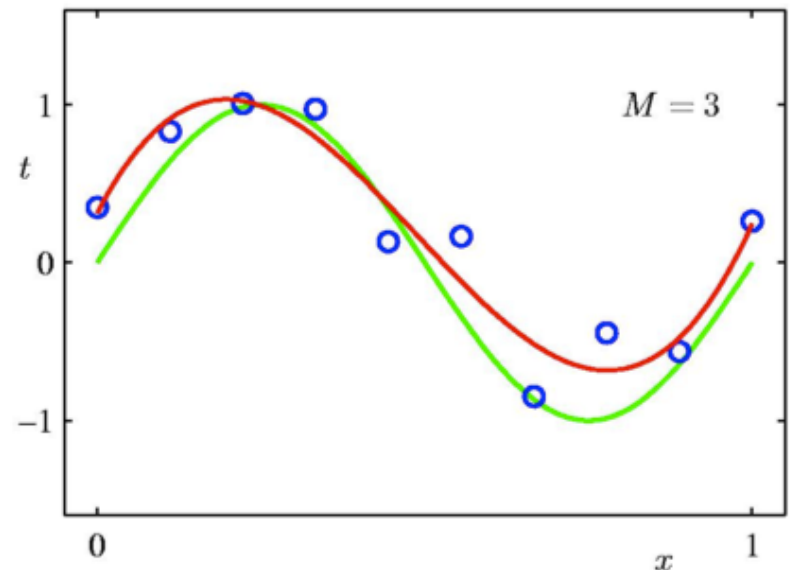
- The decision step consists of finding an estimate  $y(\mathbf{x})$  of  $t$  for each input  $\mathbf{x}$ .
- Similar to classification case, to quantify what it means to do well or poorly on a task, we need to define a loss (error) function:  $L(t, y(\mathbf{x}))$ .

- The average, or expected, loss is given by:

$$\mathbb{E}[L] = \int \int L(t, y(\mathbf{x})) p(\mathbf{x}, t) d\mathbf{x} dt.$$

- If we use squared loss, we obtain:

$$\mathbb{E}[L] = \int \int (t - y(\mathbf{x}))^2 p(\mathbf{x}, t) d\mathbf{x} dt.$$



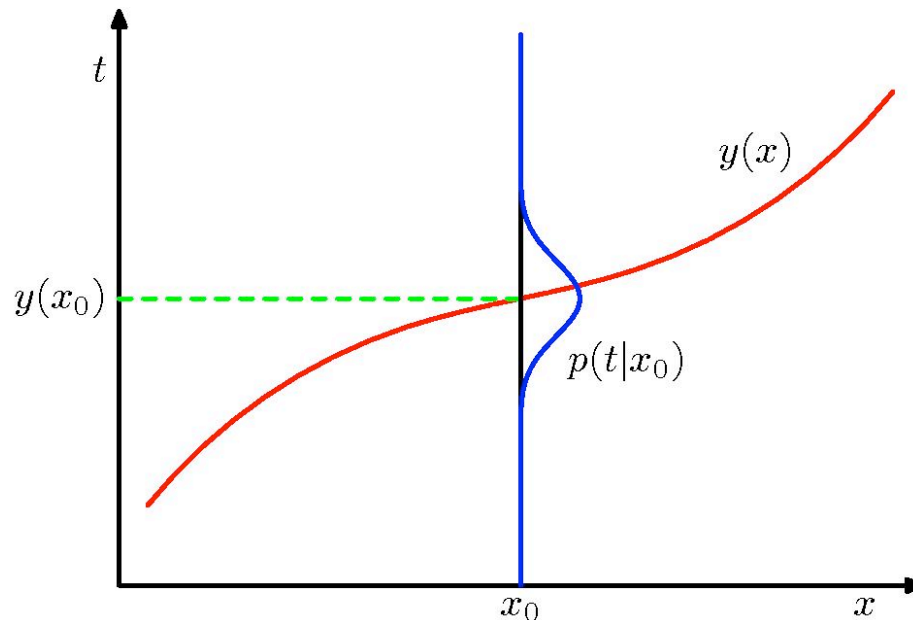
# Squared Loss Function

- If we use squared loss, we obtain:

$$\mathbb{E}[L] = \int \int (t - y(\mathbf{x}))^2 p(\mathbf{x}, t) d\mathbf{x} dt.$$

- Our goal is to choose  $y(x)$  so as to minimize the expected squared loss.
- The optimal solution (if we assume a completely flexible function) is the conditional average:

$$y(\mathbf{x}) = \int t p(t|\mathbf{x}) dt = \mathbb{E}[t|\mathbf{x}].$$



The regression function  $y(\mathbf{x})$  that minimizes the expected squared loss is given by the mean of the conditional distribution  $p(t|\mathbf{x})$ .

# Squared Loss Function

- If we use squared loss, we obtain:

$$\begin{aligned}(y(\mathbf{x}) - t)^2 &= (y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}] + \mathbb{E}[t|\mathbf{x}] - t)^2 \\ &= (y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}])^2 + 2(y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}])(\mathbb{E}[t|\mathbf{x}] - t) + (\mathbb{E}[t|\mathbf{x}] - t)^2.\end{aligned}$$

- Plugging into expected loss:

$$\mathbb{E}[L] = \underbrace{\int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x}) d\mathbf{x}}_{\text{expected loss is minimized when } y(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}]} + \underbrace{\int \text{var}[t|\mathbf{x}] p(\mathbf{x}) d\mathbf{x}}_{\text{intrinsic variability of the target values.}}$$

Because it is independent noise, it represents an irreducible minimum value of expected loss.

# Other Loss Function

- Simple generalization of the squared loss, called the *Minkowski* loss:

$$\mathbb{E}[L] = \int \int (t - y(\mathbf{x}))^q p(\mathbf{x}, t) d\mathbf{x} dt.$$

- The minimum of  $\mathbb{E}[L]$  is given by:
  - the conditional mean for  $q=2$ ,
  - the conditional median when  $q=1$ , and
  - the conditional mode for  $q \rightarrow 0$ .

# Bias-Variance Decomposition

- Introducing a regularization term can help us control overfitting. But how can we determine a suitable value of the regularization coefficient?
- Let us examine the expected squared loss function. Remember:

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \underbrace{\int \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt}_{\text{intrinsic variability of the target values: The minimum achievable value of expected loss}}$$

for which the optimal prediction is given by the conditional expectation:

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x}) dt.$$

intrinsic variability of the target values: The minimum achievable value of expected loss

- If we model  $h(\mathbf{x})$  using a parametric function  $y(\mathbf{x}, \mathbf{w})$ , then from a Bayesian perspective, the uncertainty in our model is expressed through the posterior distribution over parameters  $\mathbf{w}$ .
- We first look at the frequentist perspective.

# Bias-Variance Decomposition

- From a frequentist perspective: we make a point estimate of  $\mathbf{w}^*$  based on the dataset  $D$ .
- We next interpret the uncertainty of this estimate through the following thought experiment:
  - Suppose we had a large number of datasets, each of size  $N$ , where each dataset is drawn independently from  $p(\mathbf{x}, t)$ .
  - For each dataset  $D$ , we can obtain a prediction function  $y(\mathbf{x}; \mathcal{D})$ .
  - Different datasets will give different prediction functions.
  - The performance of a particular learning algorithm is then assessed by taking the average over the ensemble of these datasets.

- Let us consider the expression:

$$\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2.$$

- Note that this quantity depends on a particular dataset  $D$ .

# Bias-Variance Decomposition

- Consider:

$$\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2.$$

- Adding and subtracting the term  $\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]$ , we obtain

$$\begin{aligned} & \{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2 \\ &= \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] + \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\ &= \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\ &\quad + 2\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}. \end{aligned}$$


- Taking the expectation over  $\mathcal{D}$ , the last term vanishes, so we get:

$$\begin{aligned} & \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2] \\ &= \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2]}_{\text{variance}}. \end{aligned}$$



# Bias-Variance Trade-off

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



Average predictions over all datasets differ from the optimal regression function.

Solutions for individual datasets vary around their averages -- how sensitive is the function to the particular choice of the dataset.

Intrinsic variability of the target values.

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

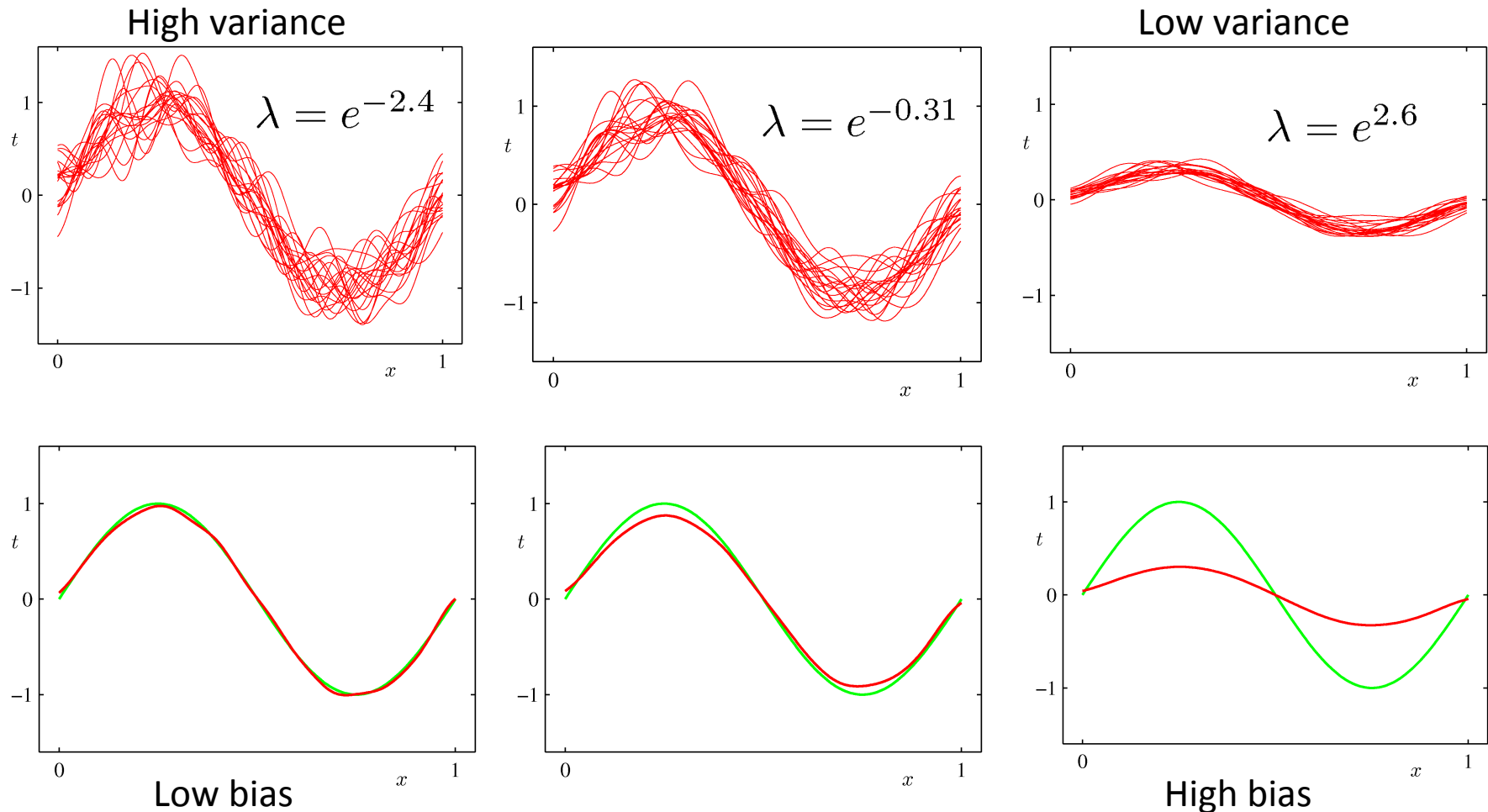
$$\text{variance} = \int \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2] p(\mathbf{x}) d\mathbf{x}$$

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

- Trade-off between bias and variance: With very flexible models (high complexity) we have low bias and high variance; With relatively rigid models (low complexity) we have high bias and low variance.
- The model with the optimal predictive capabilities has to balance between bias and variance.

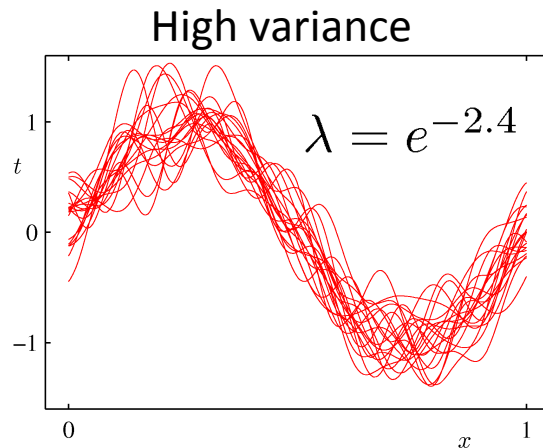
# Bias-Variance Trade-off

- Consider the sinusoidal dataset. We generate 100 datasets, each containing  $N=25$  points, drawn independently from  $h(x) = \sin 2\pi x$ .



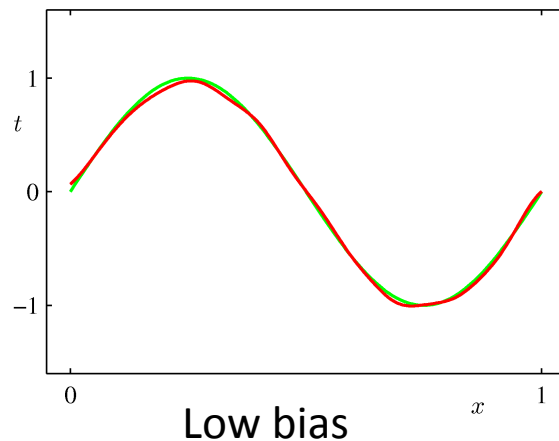
# Bias-Variance Trade-off

- Consider the sinusoidal dataset. We generate 100 datasets, each containing  $N=25$  points, drawn independently from  $h(x) = \sin 2\pi x$ .



- Note that averaging many solutions to the complex model with  $M=25$  data points represents a very good fit to the regression function

- Averaging may be a beneficial procedure.



- Let us examine the bias-variance trade-off quantitatively.

# Bias-Variance Trade-off

- Consider the sinusoidal dataset. We generate 100 datasets, each containing  $N=25$  points, drawn independently from  $h(x) = \sin 2\pi x$ .
- The average prediction is estimated as:

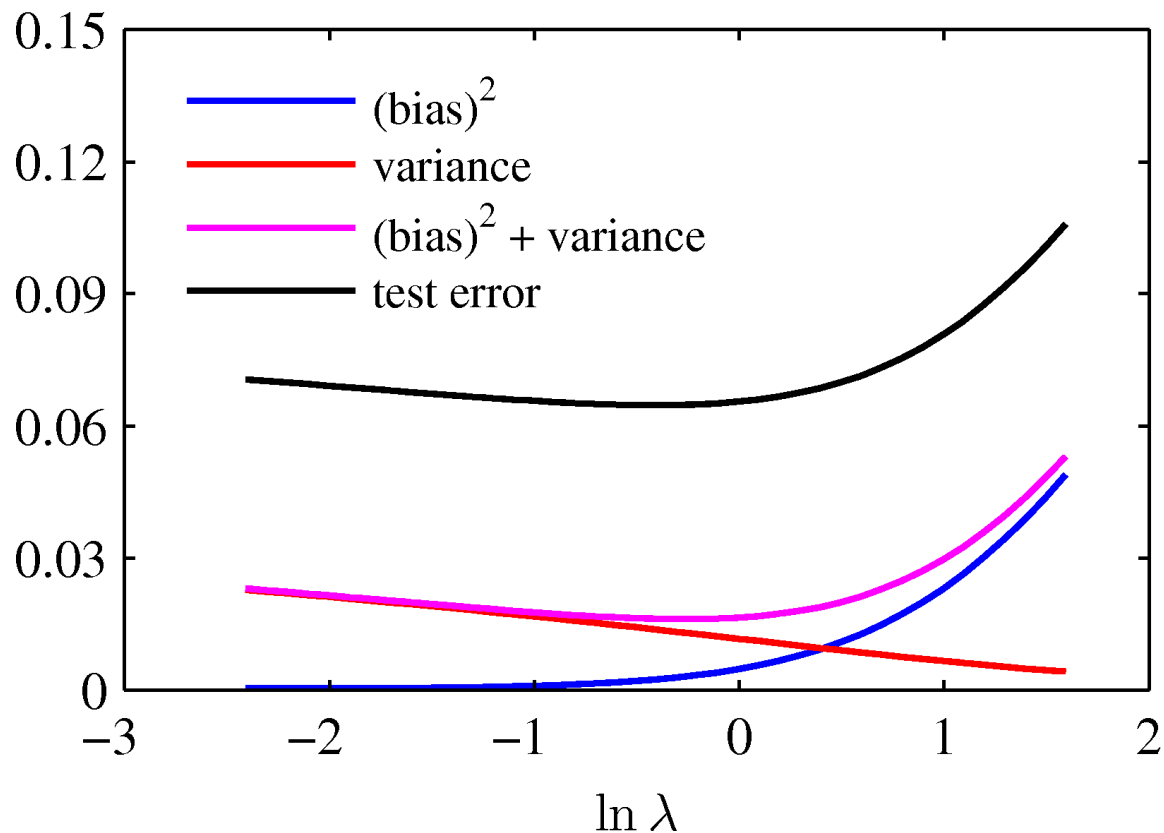
$$\bar{y} = \frac{1}{L} \sum_{l=1}^L y^{(l)}(x).$$
$$\begin{aligned} (\text{bias})^2 &= \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} \\ \text{variance} &= \int \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2] p(\mathbf{x}) d\mathbf{x} \end{aligned}$$

- And the integrated squared bias and variance are given by:

$$\begin{aligned} (\text{bias})^2 &= \frac{1}{N} \sum_{n=1}^N [\bar{y}(x_n) - h(x_n)]^2 \\ \text{variance} &= \frac{1}{N} \sum_{n=1}^N \frac{1}{L} \sum_{l=1}^L [y^{(l)}(x_n) - \bar{y}(x_n)]^2 \end{aligned}$$

where the integral over  $x$  weighted by the distribution  $p(x)$  is approximated by the finite sum over data points drawn from that distribution.

# Bias-Variance Trade-off



From these plots note that over-regularized model (large  $\lambda$ ) has high bias, and under-regularized model (low  $\lambda$ ) has high variance.

# Beating the Bias-Variance Trade-off

- We can reduce the variance by averaging over many models trained on different datasets:
  - In practice, we only have a single observed dataset. If we had many independent training sets, we would be better off combining them into one large training dataset. With more data, we have less variance.
- Given a standard training set  $D$  of size  $N$ , we could generate new training sets,  $N$ , by sampling examples from  $D$  uniformly and with replacement.
  - This is called bagging and it works quite well in practice.
- Given enough computation, we would be better off resorting to the Bayesian framework (which we will discuss next):
  - Combine the predictions of many models using the posterior probability of each parameter vector as the combination weight.