STA 4273H:
Statistical Machine Learning

Russ Salakhutdinov

Department of Statistics
rsalakhu@utstat.toronto.edu
http://www.cs.toronto.edu/~rsalakhu/

Lecture 3

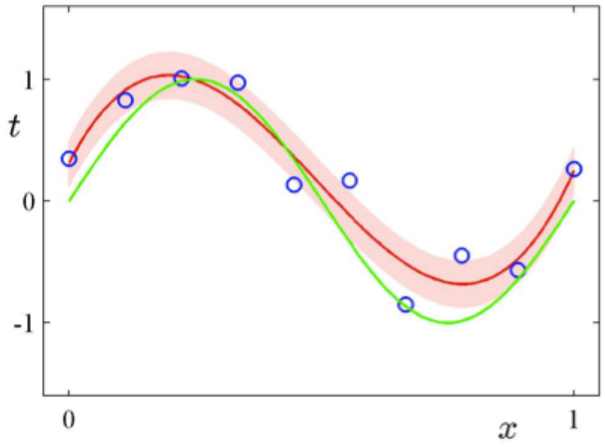
Parametric Distributions

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

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

• We will also assume that the data points are i.i.d

• We will focus on the maximum likelihood estimation $\boldsymbol{\theta}^\star$



• 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:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + ... + w_d x_d = w_0 + \sum_{j=1}^{d} w_j x_j,$$

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

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

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

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

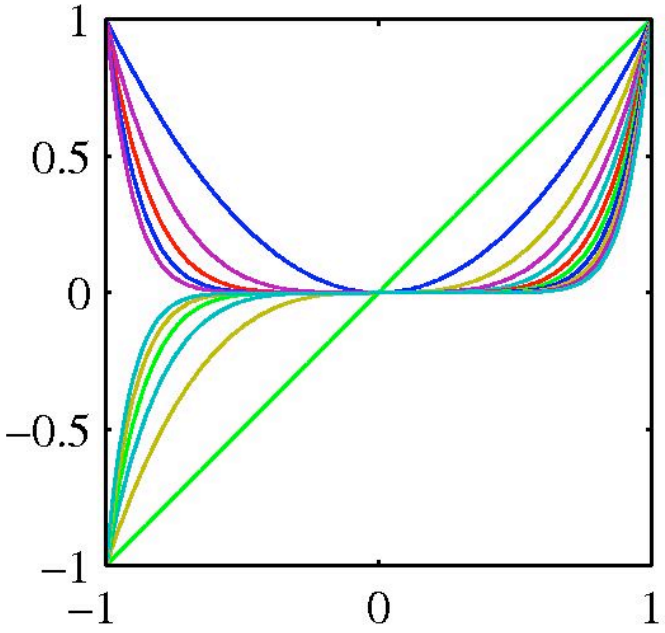
where $\phi_j(\mathbf{x})$ are known as basis functions.

- 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:

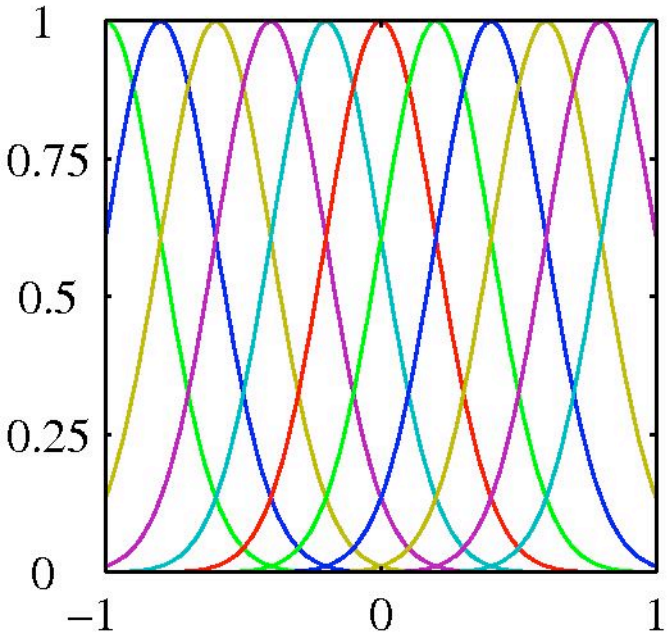
$$\phi_j(x) = x^j.$$



Basis functions are global: small changes in **x** affect all basis functions.

Gaussian basis functions:

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

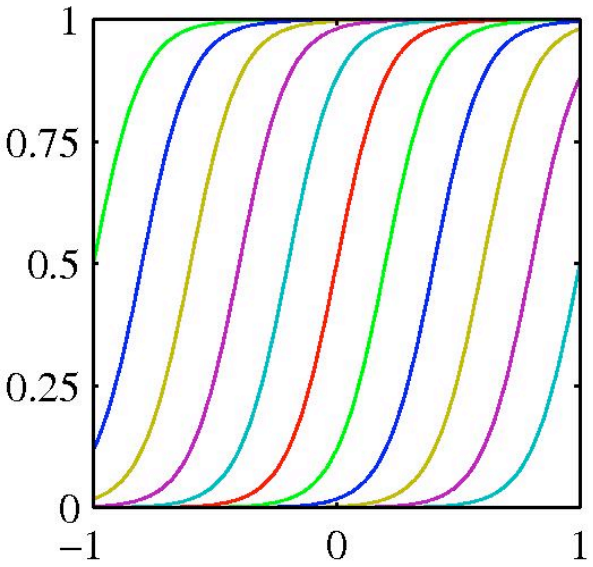


Basis functions are local: small changes in **x** only affect nearby basis functions.
μ_j and s control location and scale (width).

Linear Basis Function Models

Sigmoidal basis functions

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



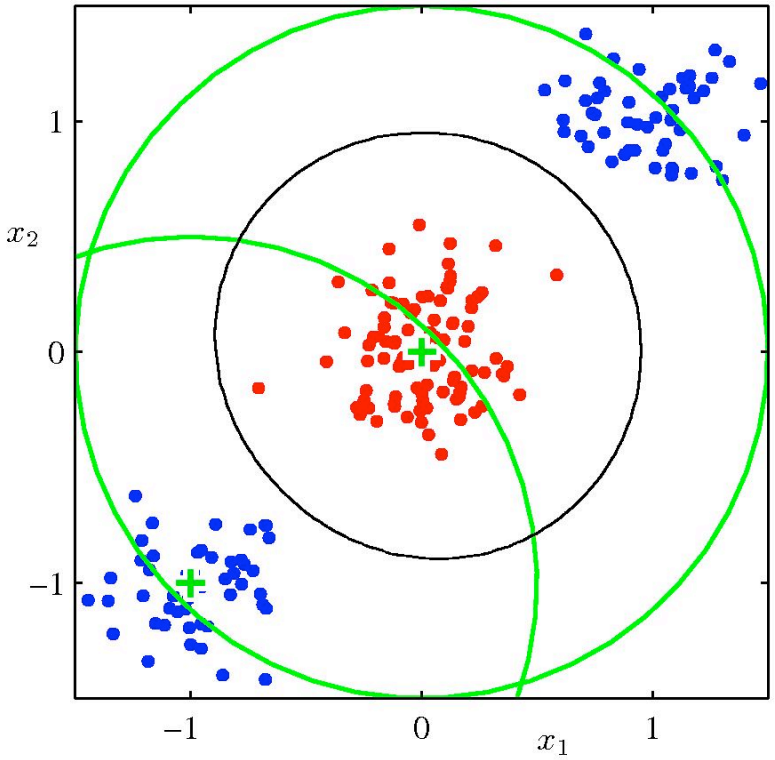
Basis functions are local: small changes in **x** only affect nearby basis functions. μ_j and s control location and scale (slope).

• Decision boundaries will be linear in the feature space ϕ, but would correspond to nonlinear boundaries in the original input space x.

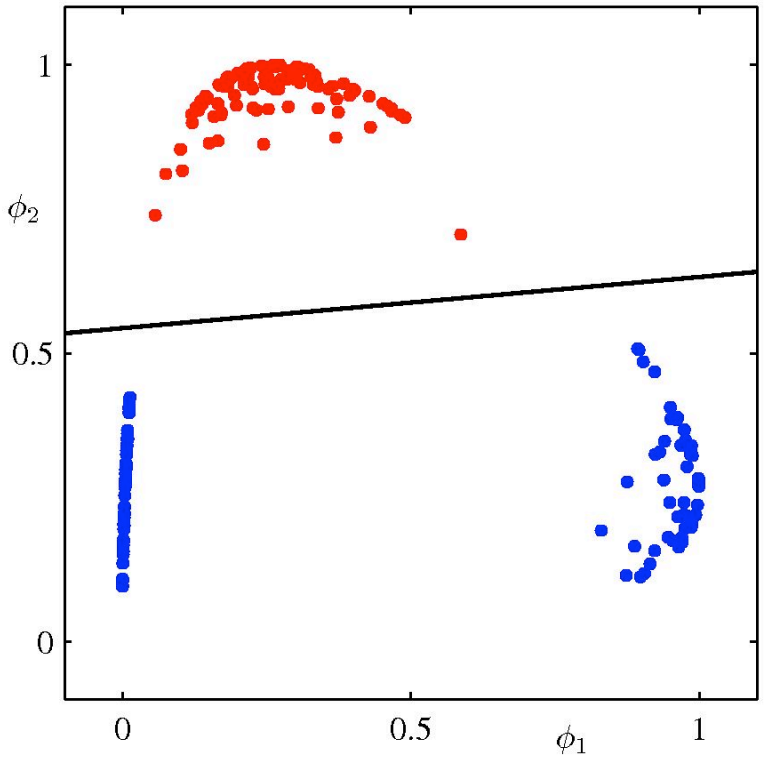
• Classes that are linearly separable in the feature space $\phi(\mathbf{x})$ need not be linearly separable in the original input space.

Linear Basis Function Models

Original input space



Corresponding feature space using two Gaussian basis functions



• We define two Gaussian basis functions with centers shown by green the crosses, and with contours shown by the green circles.

• Linear decision boundary (right) is obtained using logistic regression, and corresponds to nonlinear decision boundary in the input space (left, black curve).

Maximum Likelihood

• As before, assume observations from deterministic function with 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 $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, and corresponding target values $\mathbf{t} = [t_1, t_2, ..., t_N]^T$ 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 $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), ..., \phi_{M-1}(\mathbf{x}))^T$.

Maximum Likelihood

Taking the logarithm, we obtain:

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

$$= -\frac{\beta}{2} \underbrace{\sum_{n=1}^{N} \left(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right)^2}_{\text{sum-of-squares error function}} + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi).$$

Differentiating and setting to zero yields:

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

Maximum Likelihood

Differentiating and setting to zero yields:

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

Solving for **w**, we get:

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

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

where $\boldsymbol{\Phi}$ is known as the design matrix:

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

Geometry of Least Squares

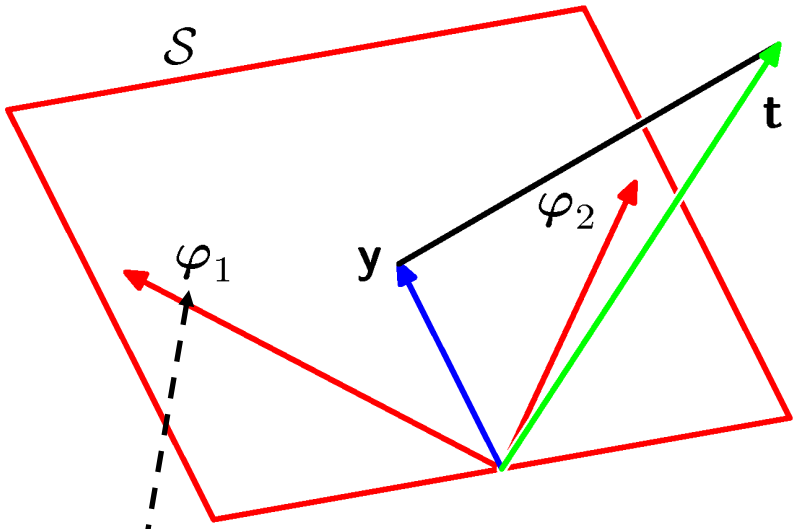
• Consider an N-dimensional space, so that $\mathbf{t} = [t_1, t_2, ..., t_N]^T$ is a vector in that space.

• 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: $\mathbf{y} = \mathbf{\Phi}\mathbf{w}_{\mathbf{ML}}$.

• The sum-of-squares error is equal to the squared Euclidean distance between **y** and **t** (up to a factor of 1/2).



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

The solution corresponds to the orthogonal projection of **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):

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

weights after seeing training case t+1

learning rate

vector of derivatives of the squared error w.r.t. the weights on the training case presented at time t.

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

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

• 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

λ is called the regularization coefficient.

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

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

which is minimized by setting:

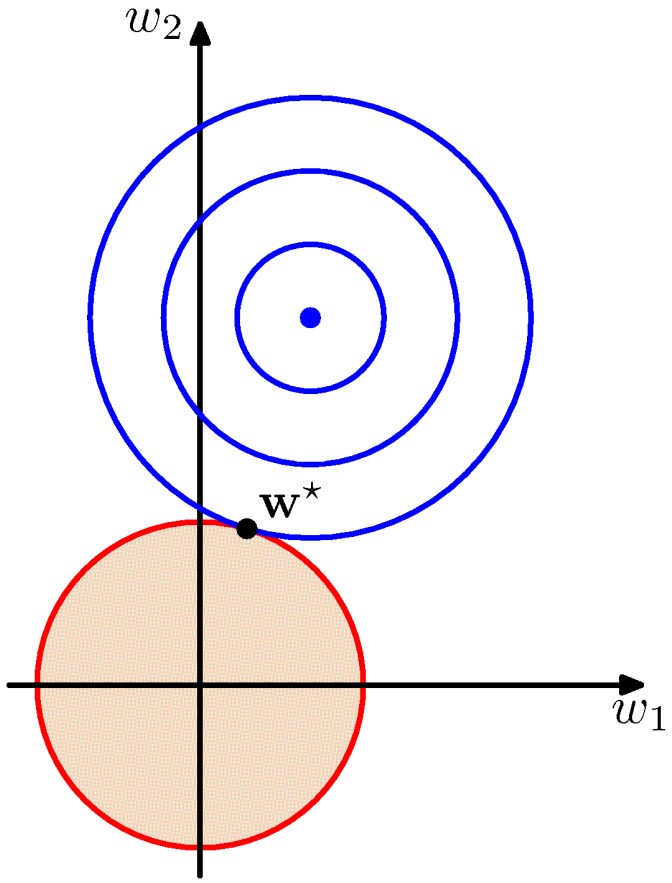
Ridge regression

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

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

Effect of Regularization

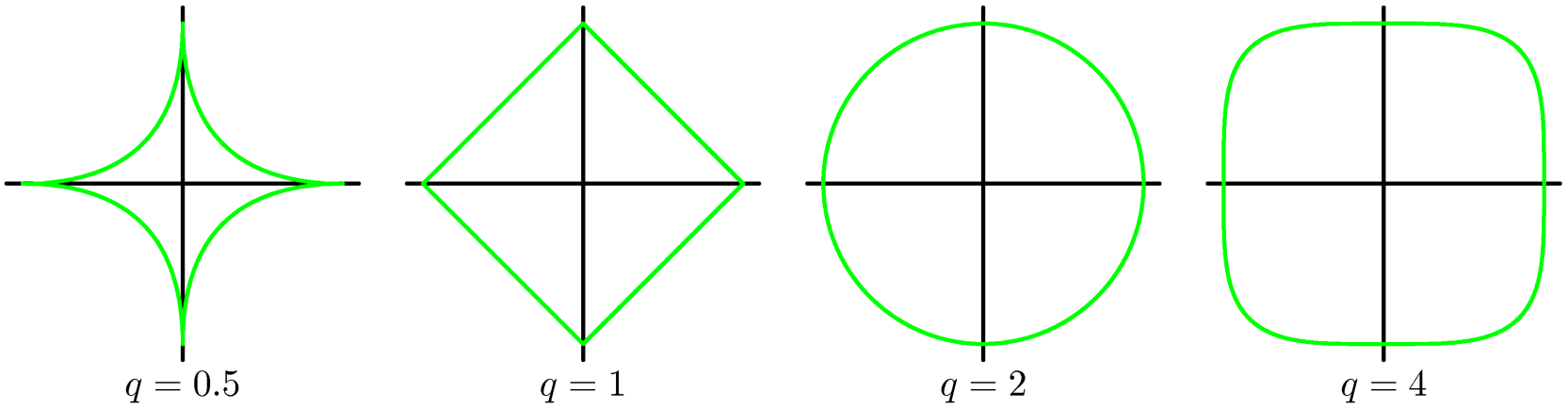
- The overall error function is the sum of two parabolic bowls.
- The combined minimum lies on the line between the minimum of the squared error and the origin.
- The regularizer shrinks model parameters to zero.



Other Regularizers

Using a more general regularizer, we get:

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



Lasso

Quadratic

The Lasso

- Penalize the absolute value of the weights:

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

- For sufficiently large λ, some of the coefficients will be driven to exactly zero, leading to a sparse model.

- The above formulation is equivalent to:

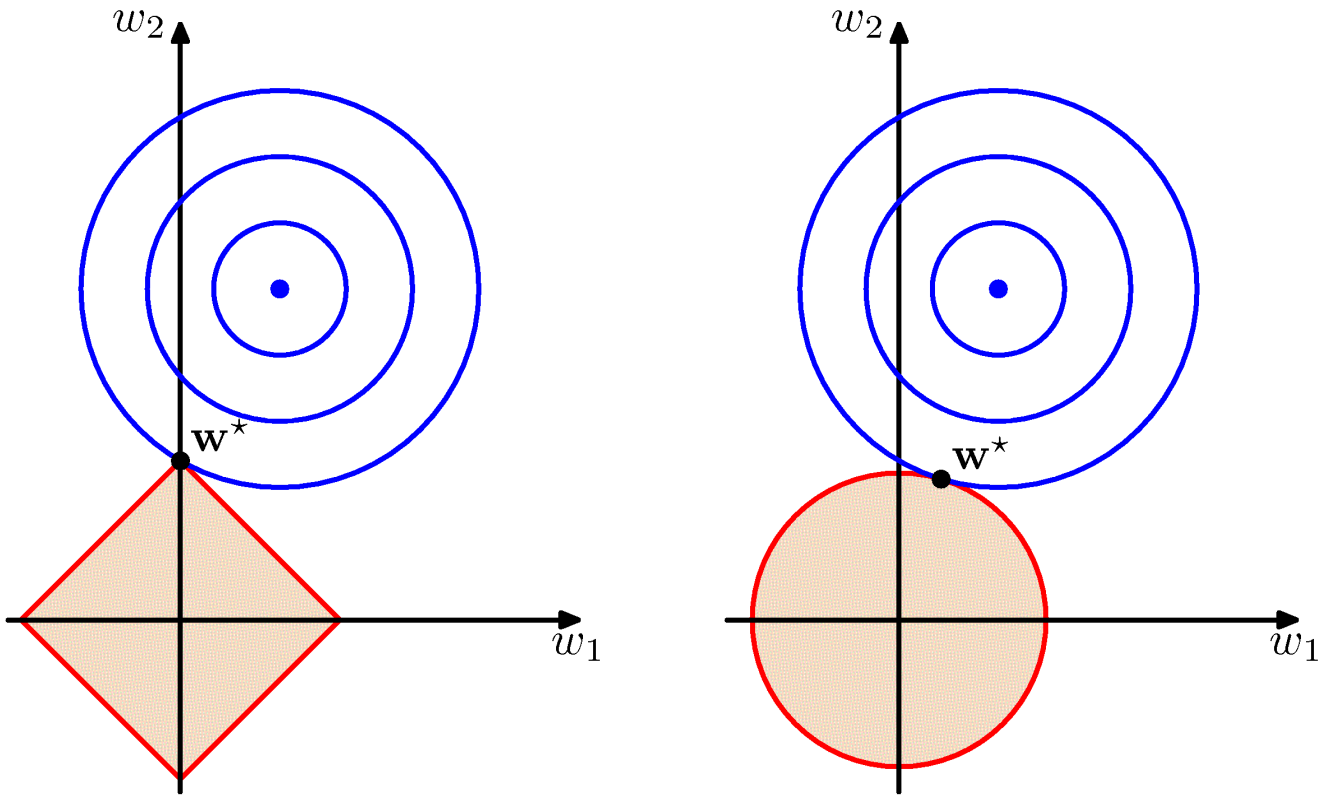
$$\mathbf{w}^{lasso} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2} \underbrace{\sum_{n=1}^{N} \left(t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) \right)^2}_{\text{unregularized sum-of-squares error}}, \text{ subject to } \sum_{j=1}^{M-1} |w_j| \leq \tau.$$

- The two approaches are related using Lagrange multiplies.

- 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 regualrizer (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 **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 **x**:
  - for regression: t is a real-valued continuous target.
  - for classification: t a categorical variable representing class labels.

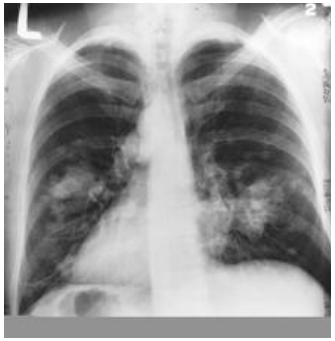
The joint probability distribution $p(\mathbf{x}, t)$ provides a complete summary of uncertainties associated with these random variables.

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 **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.



C_1: Cancer present

C_2: Cancer absent

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:

posterior probability of C_k given observed data.

probability of observed data given C_k

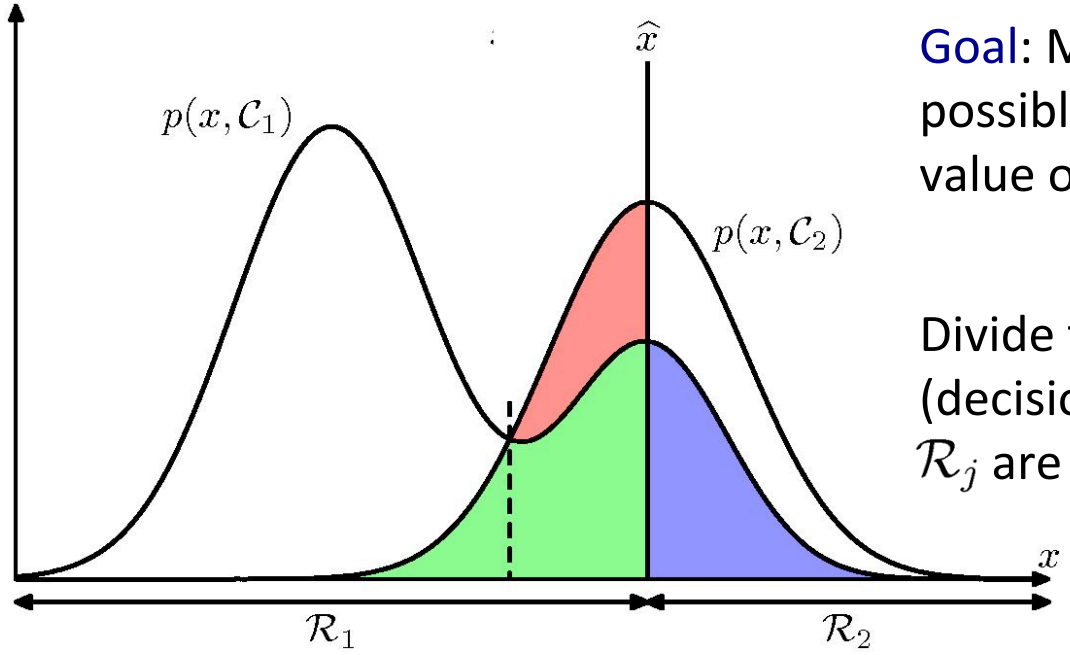
prior probability for class C_k

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

Bayes' Rule

• If our goal to minimize the probability of assigning **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 **x** to one of the available classes.

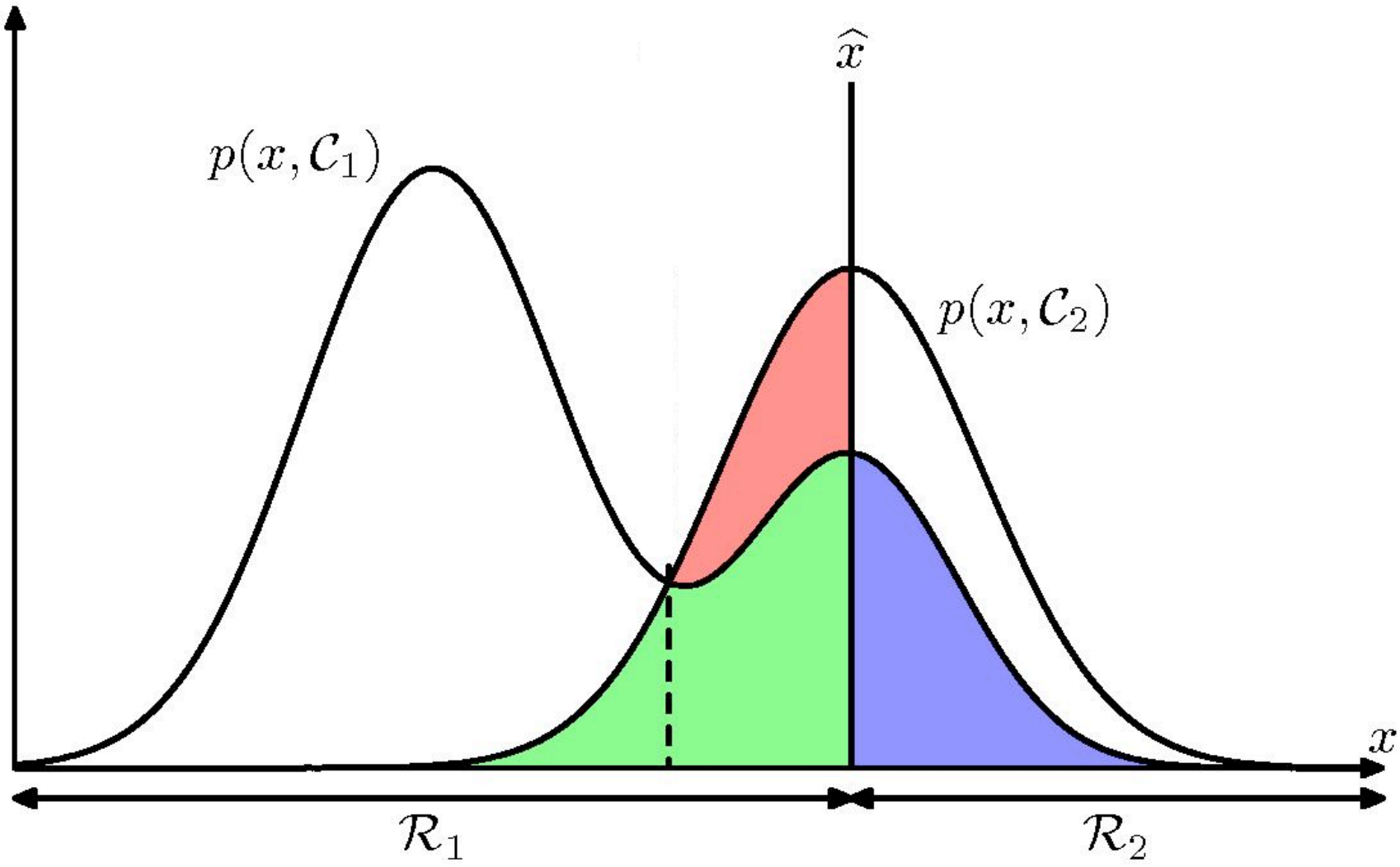
Divide the input space into regions $\mathcal{R}_j$ (decision regions), such that all points in $\mathcal{R}_j$ are assigned to class $\mathcal{C}_j$.

red+green regions: input belongs to class C_2, but is assigned to C_1

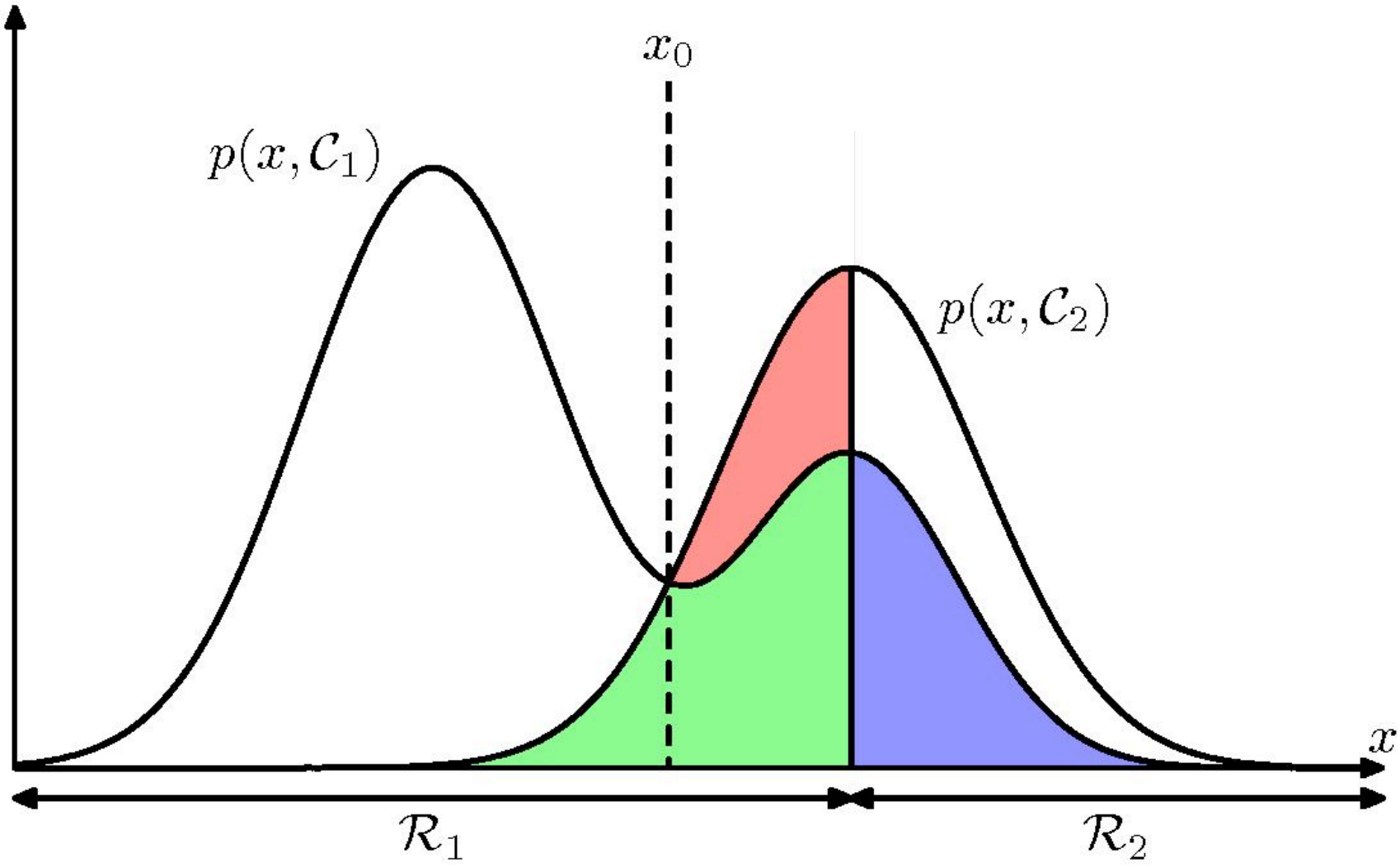
blue region: input belongs to class C_1, but is assigned to C_2

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

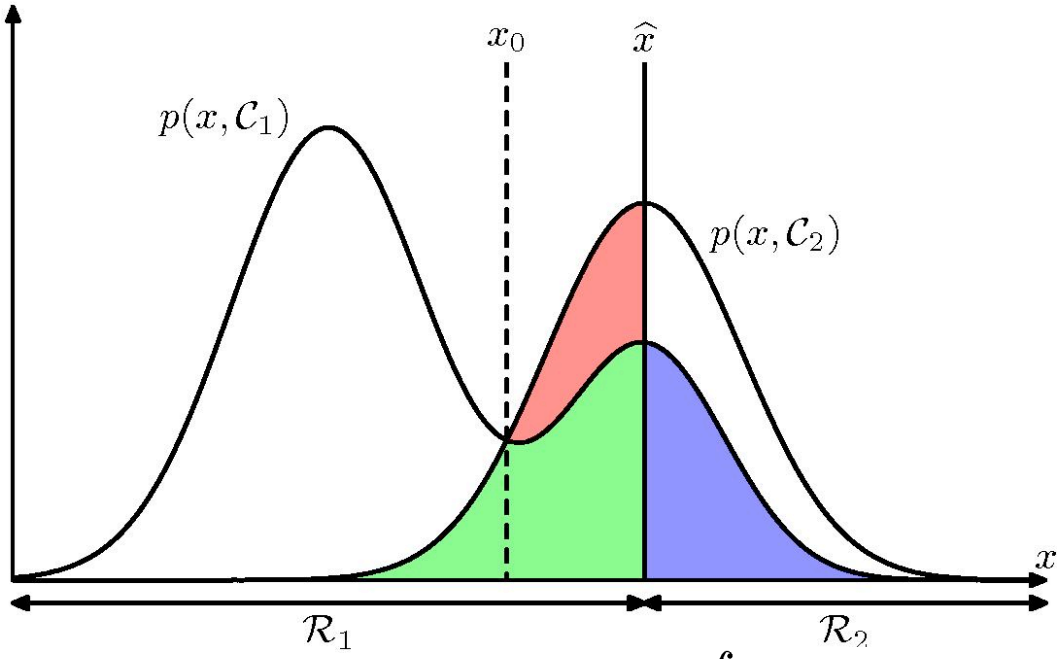
Minimizing Misclassification Rate



Minimizing Misclassification Rate



Minimizing Misclassification Rate



$$p(\text{mistake}) = p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) = \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2)\mathrm{d}\mathbf{x} + \int_{R_2} p(\mathbf{x}, \mathcal{C}_1)\mathrm{d}\mathbf{x}$$

if $p(\mathbf{x}, \mathcal{C}_1) > p(\mathbf{x}, \mathcal{C}_2)$ then we should assign $\mathbf{x}$ to class $\mathcal{C}_1$.

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

Expected Loss

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

Consider medical diagnosis example: example of a loss matrix:

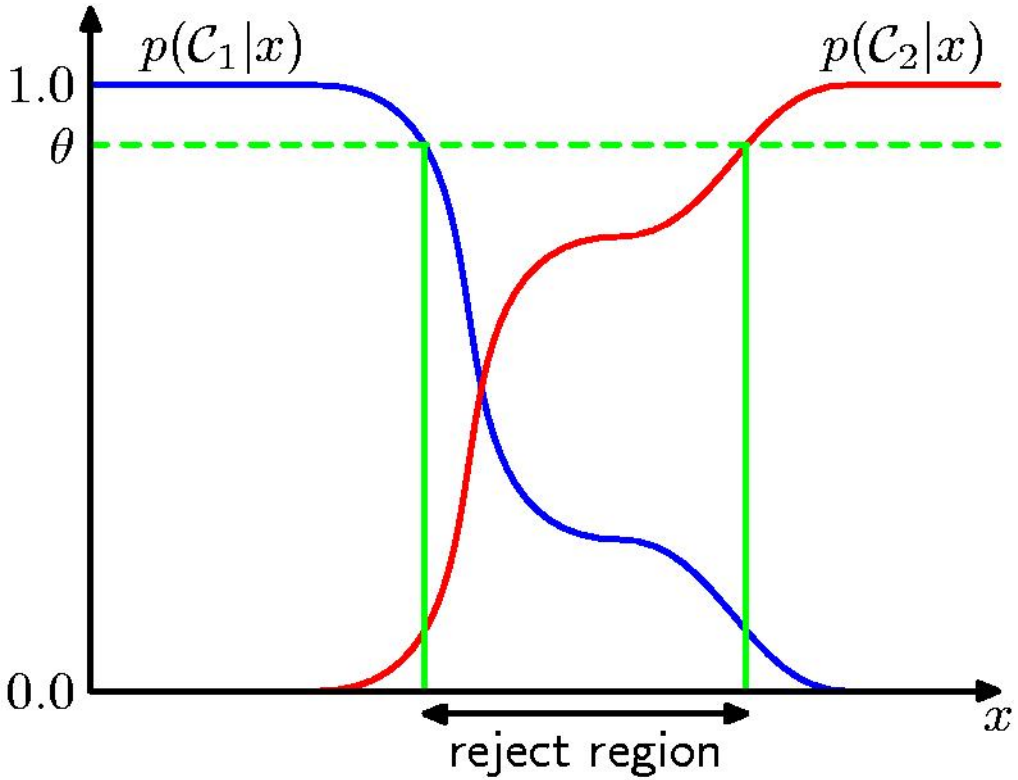
		Decision	
		cancer	normal
Truth	cancer	0	1000
	normal	1	0

Expected Loss:

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

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

Reject Option



Regression

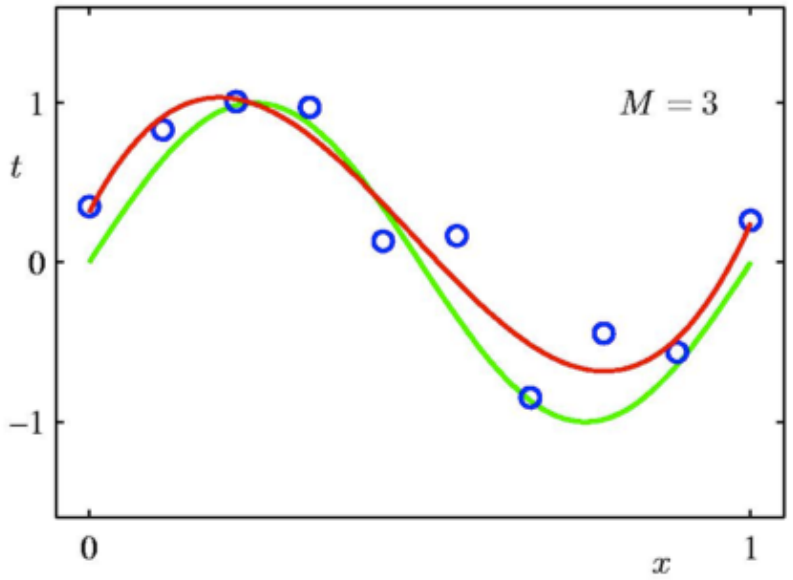
Let **x** $\in$ R^d denote a real-valued input vector, and t $\in$ R denote a real-valued random target (output) variable with joint distribution $p(\mathbf{x}, t)$.

- The decision step consists of finding an estimate y(**x**) of t for each input **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:

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

- If we use squared loss, we obtain:

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



Squared Loss Function

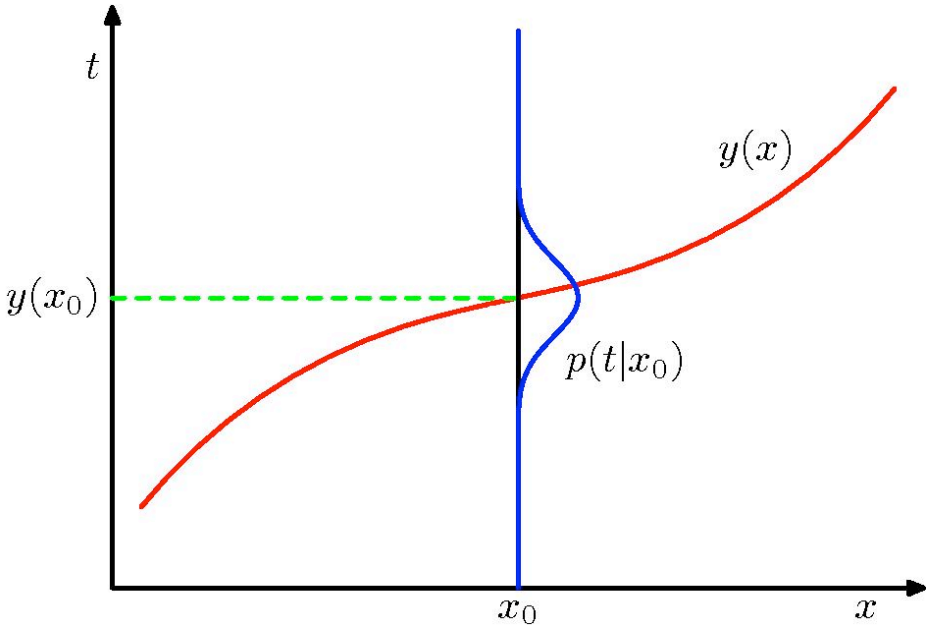
- If we use squared loss, we obtain:

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

- Our goal is to choose y(x) so as minimize expected squared loss.

- The optimal solution (if we assume a completely flexible function) is the conditional average:

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



The regression function y(**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:

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

- Plugging into expected loss:

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \int \mathrm{var}\,[t|\mathbf{x}]\, p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

expected loss is minimized when $y(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}]$.

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:

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

- 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 regularization term can control overfitting, but how can we determine a suitable value of the regularization coefficient.

• Let us examine expected squared loss function. Remember:

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t$$

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

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

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 uncertainly in our model is expressed through the posterior distribution over parameters **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 data set D.

• We next interpret the uncertainly 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 data sets will give different prediction functions.
- The performance of a particular learning algorithm is then assessed by taking the average over the ensemble of data sets.

• 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:

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

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.

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

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

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

- Trade-off between bias and variance with very flexible models (high complexity) having low bias and high variance, and relative rigid models (low complexity) having high bias and low variance.
- The model with the optimal predictive capabilities has to balance between bias and variance.

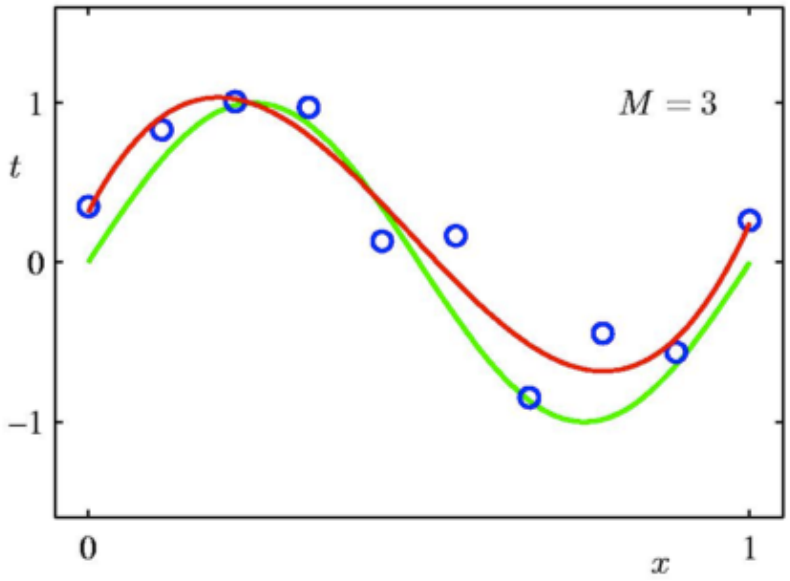
Bias-Variance Trade-off

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

• The datasets are indexed by l=1,…,L, where L=100.

• For each dataset, we fit a model with 24 Gaussian basis functions by minimizing the regularized error function:

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

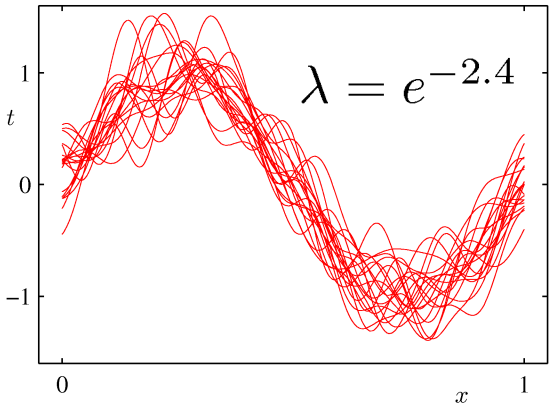


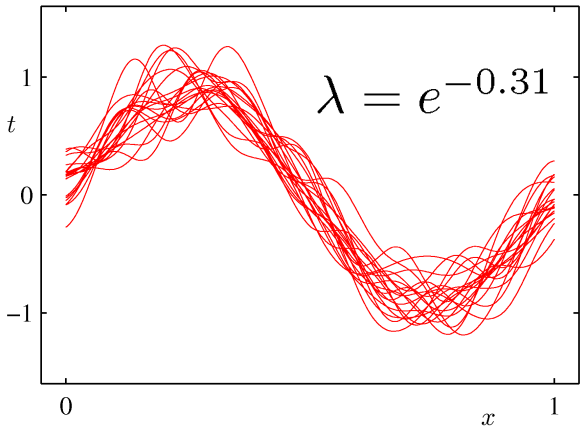
• Once the model is fit, we can make predictions $y^{(l)}(x)$ for each of the L datasets.

Bias-Variance Trade-off

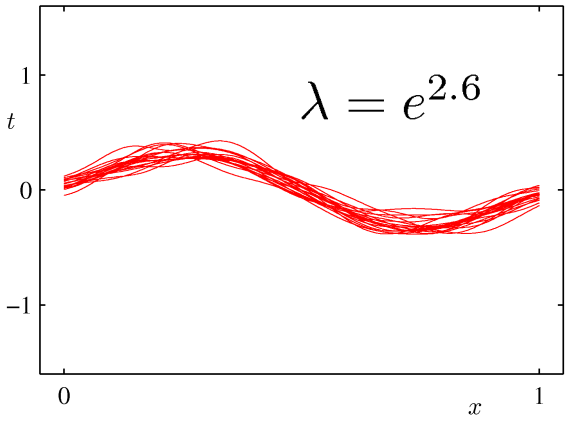
• Consider the sinusoidal dataset. We generate 100 datasets, each containing N=25 points, drawn independently from $h(x) = \sin 2\pi x$.

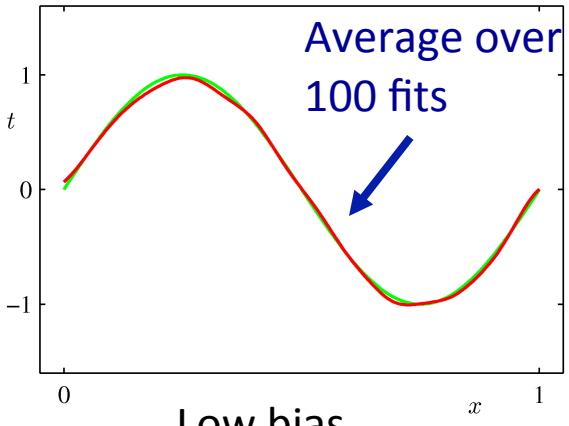
High variance



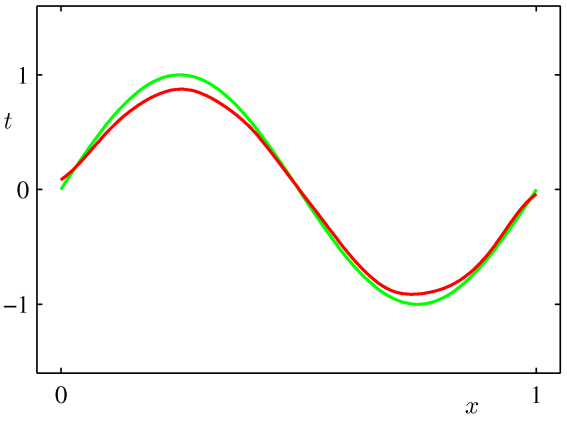


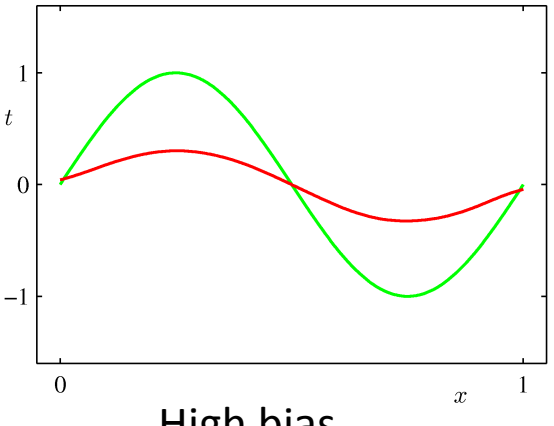
Low variance





Low bias



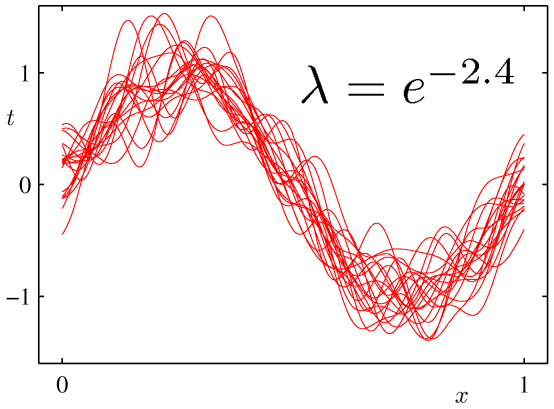


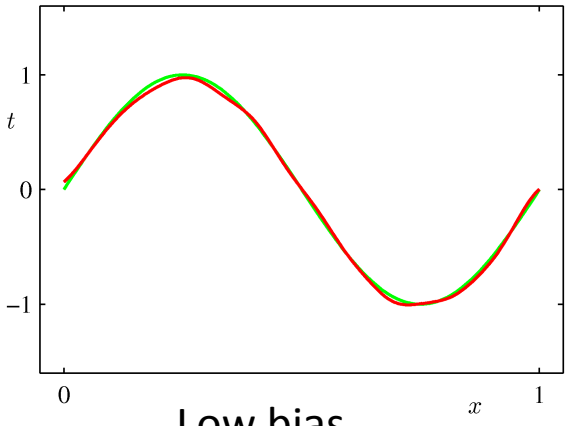
High bias

Bias-Variance Trade-off

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

High variance





Low bias

• 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:

$$\bar{y} = \frac{1}{L}\sum_{l=1}^{L} y^{(l)}(x).$$

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

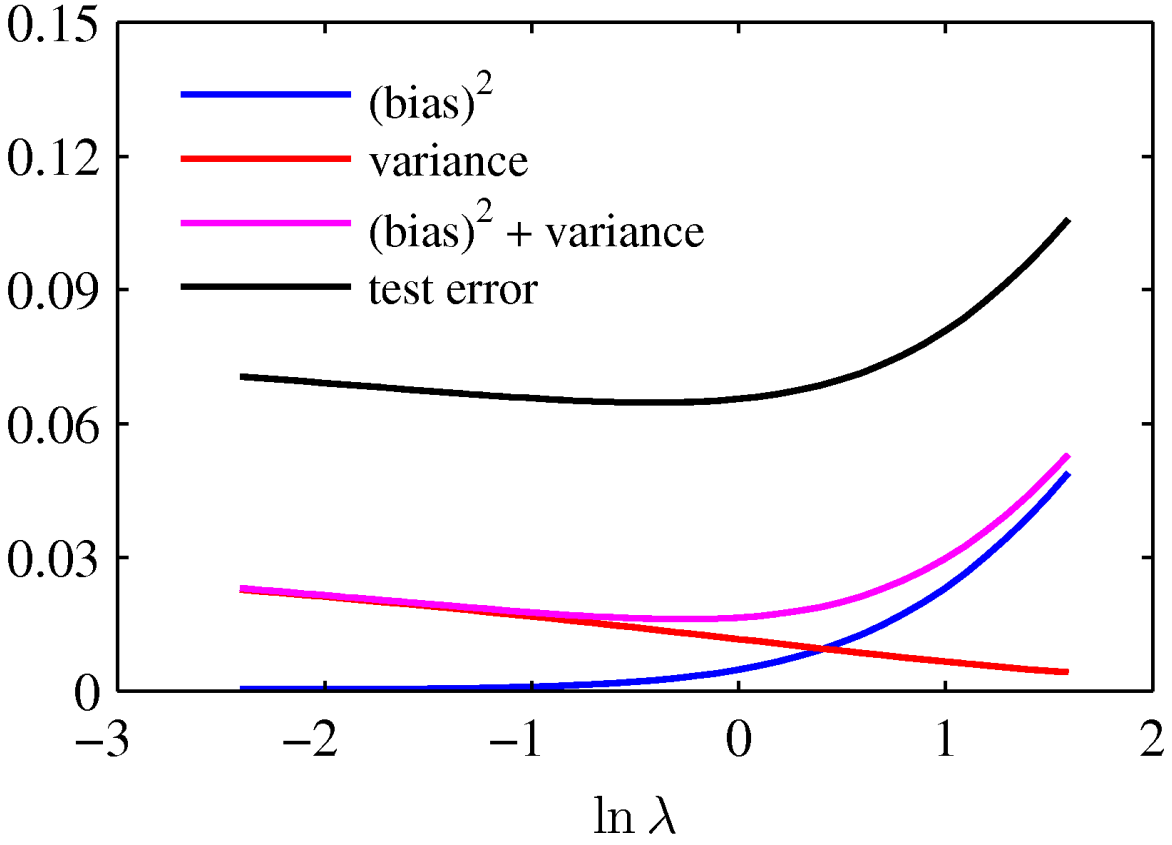
• And the integrated squared bias and variance are given by:

$$(\text{bias})^2 = \frac{1}{N}\sum_{n=1}^{N}\left[\bar{y}(x_n) - h(x_n)\right]^2$$

$$\text{variance} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{L}\sum_{l=1}^{L}\left[y^{(l)}(x_n) - \bar{y}(x_n)\right]^2$$

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 λ) has high bias, and under-regularized model (low λ) 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 set, 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 generates new training sets, N, by sampling examples from D uniformly and with replacement.

- This is called bagging and work 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.