

ECE521 week 3:
23/26 January 2017



UNIVERSITY OF
TORONTO

Outline

- **Probabilistic interpretation of linear regression**
 - Maximum likelihood estimation (MLE)
 - Maximum a posteriori (MAP) estimation
- Bias-variance trade-off
- Linear basis function models
- Classification

Outline

- **Probabilistic interpretation of linear regression**
 - Maximum likelihood estimation (MLE)
 - Maximum a posteriori (MAP) estimation
- Prerequisite:
 - Comfortable with: Bayes' rule; multivariate Gaussian; L2 norm; matrix vector products; linear algebra
- Learning objective:
 - Understand the relationship between Gaussian and squared L2 loss
 - Understand the probabilistic assumptions we make in modelling and their computational gain
 - Know how to derive MLE and MAP estimation from some assumptions about the likelihood and the prior
 - e.g. Given likelihood and priors are independent Gaussian, derive the L2 penalized loss
 - Understand how to derive learning algorithms from MLE and MAP
 - 1. Make some assumptions about the likelihood and prior
 - 2. Derive the loss function from log likelihood and log prior
 - 3. Minimize the loss to find the parameters

Parametric Distributions

- We want to model the probability distribution $P(\mathbf{x}, \mathbf{y}|W)$ of the random variables \mathbf{x}, \mathbf{y} given a finite set of observations: $\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \dots, (\mathbf{x}^{(M)}, \mathbf{y}^{(M)})\}$

$$\mathbf{x}^{(m)} \in \mathbb{R}^N, \quad \mathbf{y}^{(m)} \in \mathbb{R}^D$$

Fitting/learning a parametric model is to determine W given the dataset

- Under our model $P(\mathbf{x}, \mathbf{y}|W)$, the data likelihood is expressed as a joint distribution of all the observations

$$P(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \dots, \mathbf{x}^{(M)}, \mathbf{y}^{(M)} | W)$$

Parametric Distributions

- We will also assume that the training examples are independent and identically distributed (i.i.d.) random variables
- The independence assumption leads to a factorization of the data likelihood.
- The identically distributed assumption corresponds to each data point sharing the same parameters W .

$$P(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \dots, \mathbf{x}^{(M)}, \mathbf{y}^{(M)} | W) = \prod_{m=1}^M P(\mathbf{x}^{(m)}, \mathbf{y}^{(m)} | W)$$

- This is a fundamental assumption of machine learning.

Parametric Distributions

- We will focus on the maximum likelihood estimation (MLE) of the parameters W^*
- This corresponds to finding a set of parameters that maximizes the log-likelihood of the dataset:

$$\begin{aligned} W^* &= \arg \max_W \log P(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \dots, \mathbf{x}^{(M)}, \mathbf{y}^{(M)} | W) \\ &= \arg \max_W \sum_{m=1}^M \log P(\mathbf{x}^{(m)}, \mathbf{y}^{(m)} | W) \end{aligned}$$

- We often use the notation that is the minimization of the negative log-likelihood function:

$$\begin{aligned} W^* &= \arg \min_W - \sum_{m=1}^M \log P(\mathbf{x}^{(m)}, \mathbf{y}^{(m)} | W) \\ &= \arg \min_W \mathcal{L}(W) \end{aligned}$$

Supervised Learning

- Defining the log-likelihood of a single data point is equivalent to finding a distance measure between model predictions & true targets
- In supervised learning tasks, e.g. regression, classification, the goal is to learn the conditional distribution $P(\mathbf{y}|\mathbf{x})$, which is useful for making predictions
- We can either directly model the predictive distribution:

$$P(\mathbf{x}, \mathbf{y}|W) = P(\mathbf{y}|\mathbf{x}, W)P(\mathbf{x}) \quad \text{discriminative approach}$$

↑
modelling outputs

or we can model the input of the system and apply Bayes' rule:

$$P(\mathbf{x}, \mathbf{y}|W) = P(\mathbf{x}|\mathbf{y}, W)P(\mathbf{y}) \quad \text{generative approach}$$

↑
modelling inputs

Regression

- We will take the discriminative approach here and model the predictive distribution as a multivariate Gaussian distribution. The likelihood of a single training case:

$$\log P(\mathbf{x}^{(m)}, \mathbf{y}^{(m)} | W) = \log P(\mathbf{y}^{(m)} | \mathbf{x}^{(m)}, W) + \log P(\mathbf{x}^{(m)})$$

- Using the Gaussian distribution has an implicit assumption that the desired outputs are the entire set of real numbers

$$\begin{aligned} \log P(\mathbf{y}^{(m)} | \mathbf{x}^{(m)}, W) &= \log \mathcal{N}(\mathbf{y}^{(m)}; \boldsymbol{\mu}(\mathbf{x}^{(m)}), \Sigma(\mathbf{x}^{(m)})) \\ &= -\frac{1}{2} \log \det\{\Sigma(\mathbf{x}^{(m)})\} - \frac{1}{2} (\mathbf{y}^{(m)} - \boldsymbol{\mu}(\mathbf{x}^{(m)}))^T \Sigma(\mathbf{x}^{(m)})^{-1} (\mathbf{y}^{(m)} - \boldsymbol{\mu}(\mathbf{x}^{(m)})) \end{aligned}$$

- Further assume the covariance matrix $\Sigma(\mathbf{x}^{(m)}) = I$ is an identity, i.e. the output dimensions are independent

$$\begin{aligned} \log P(\mathbf{y}^{(m)} | \mathbf{x}^{(m)}, W) &= -\frac{1}{2} (\mathbf{y}^{(m)} - \boldsymbol{\mu}(\mathbf{x}^{(m)}))^T (\mathbf{y}^{(m)} - \boldsymbol{\mu}(\mathbf{x}^{(m)})) + c \\ &= -\frac{1}{2} \|\mathbf{y}^{(m)} - \boldsymbol{\mu}(\mathbf{x}^{(m)})\|_2^2 + c \end{aligned}$$

sum-of-squares loss

Linear Regression

- The negative log-likelihood of the dataset, under the assumptions of i.i.d. data and independent output dimensions, looks like:

$$-\log P(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \dots, \mathbf{x}^{(M)}, \mathbf{y}^{(M)} | W) = \sum_{m=1}^M \frac{1}{2} \|\mathbf{y}^{(m)} - \boldsymbol{\mu}(\mathbf{x}^{(m)})\|_2^2 + c$$

- Linear regression parameterizes the mean of the multivariate Gaussian distribution as a weighted sum of the input features:

$$\boldsymbol{\mu}(\mathbf{x}^{(m)}) = \hat{y}(\mathbf{x}^{(m)}) = \begin{bmatrix} \sum_{n=1}^N w_{n,1} x_n^{(m)} + b_1 \\ \sum_{n=1}^N w_{n,2} x_n^{(m)} + b_2 \\ \vdots \\ \sum_{n=1}^N w_{n,D} x_n^{(m)} + b_D \end{bmatrix} = W^T \mathbf{x}^{(m)} + b$$

- W is the weight matrix: it learns the importance of each input feature

- b is the bias vector: it learns the average y value from the dataset

Linear Regression

- The MLE of linear regression is equivalent to minimizing a sum-of-squares loss function:

$$\begin{aligned} W^*, b^* &= \arg \min_{W, b} -\log P(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \dots, \mathbf{x}^{(M)}, \mathbf{y}^{(M)} | W) \\ &= \arg \min_{W, b} \sum_{m=1}^M \frac{1}{2} \|W^T \mathbf{x}^{(m)} + b - \mathbf{y}^{(m)}\|_2^2 = \mathcal{L}(W, b) \end{aligned}$$

- This is also sometime called the least square estimator for finding the closest projection matrix for a set of linear equations

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)T} \\ \mathbf{x}^{(2)T} \\ \vdots \\ \mathbf{x}^{(M)T} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)T} \\ \mathbf{y}^{(2)T} \\ \vdots \\ \mathbf{y}^{(M)T} \end{bmatrix} \quad \mathbf{X}W \approx \mathbf{Y} \quad \Rightarrow \quad W = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

Linear Regression

- To encode our belief about the solution space $W \in \mathcal{W}$ of a problem, we may also define a prior distribution of the parameters $P(W)$

data likelihood: $P(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \dots, \mathbf{x}^{(M)}, \mathbf{y}^{(M)} | W)$

prior: $P(W)$

posterior: $P(W | \mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \dots, \mathbf{x}^{(M)}, \mathbf{y}^{(M)})$

- Posterior distribution is proportional to the product of the likelihood and the prior

$$P(W | \mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \dots, \mathbf{x}^{(M)}, \mathbf{y}^{(M)}) = \frac{P(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \dots, \mathbf{x}^{(M)}, \mathbf{y}^{(M)} | W) P(W)}{P(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \dots, \mathbf{x}^{(M)}, \mathbf{y}^{(M)})}$$
$$\propto P(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \dots, \mathbf{x}^{(M)}, \mathbf{y}^{(M)} | W) P(W)$$

Linear Regression

- Intuition: machine learning models are trained on finite datasets. The datasets are often too small or noisy to learn underlying statistical structures and patterns.

Consider a simple coin-toss experiment involving a fair coin but we do not know whether it's fair or not. We would like to estimate the probability of the heads $P(H)$ through some repeated experiments. We hit a lucky strike and see 5 heads in a roll. We decide to estimate from these 5 points:

$$\text{MLE of } P(H) = \#H / \#\text{tosses} = 5 / 5 = 1$$

It is completely misleading to believe the coin will always land on heads. This is a form of overfitting to the training set. We would like to consider the **maximum a posteriori (MAP)** estimator instead.

Linear Regression

- Designing a prior is like finding a regularizer for the likelihood model
- Notice the entries in the weight matrix are all real values. Assume every weight has the same variance and there is no covariance between weights
- We can assign a zero-mean multivariate Gaussian distribution to the weight matrix, whose covariance matrix is identity. (We usually assume no prior knowledge about the biases. Why?)
- The log Gaussian prior of the weight matrix is given by:

$$\begin{aligned}\log P(W) &= \sum_{n=1}^N \sum_{d=1}^D \log \mathcal{N}(w_{n,d}; 0, \frac{1}{\lambda}) \\ &= - \sum_{n=1}^N \sum_{d=1}^D \frac{1}{2} \lambda w_{n,d}^2 + c = - \sum_{n=1}^N \frac{\lambda}{2} \|W_n\|_2^2 + c\end{aligned}$$

Linear Regression

- The maximum a posteriori (MAP) estimator for the linear regression with Gaussian prior is:

$$\begin{aligned} W^* &= \arg \min_W -\log P(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}, \dots, \mathbf{x}^{(M)}, \mathbf{y}^{(M)} | W) - \log P(W) \\ &= \arg \min_W \sum_{m=1}^M \frac{1}{2} \|W^T \mathbf{x}^{(m)} + b - \mathbf{y}^{(m)}\|_2^2 + \frac{\lambda}{2} \sum_{n=1}^N \|W_n\|_2^2 \end{aligned}$$

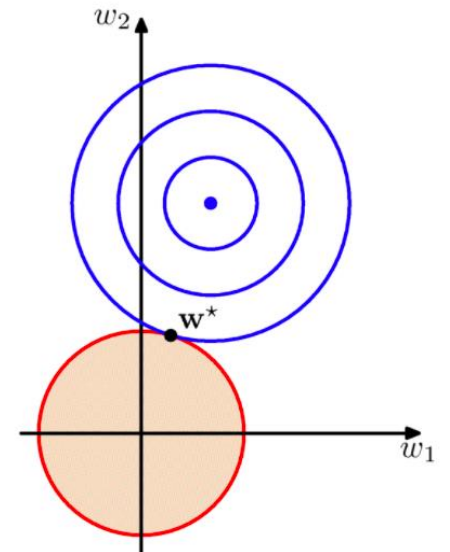
- This is also known as the ℓ_2 -penalized linear regression or ridge regression. The Gaussian prior on the weights is therefore equivalent to weight decay.

Linear Regression

- Weight decay or ℓ_2 weight regularization is one way to prevent models from overfitting. It regularizes the capacity of a model by making the weights close to zero.

$$\mathcal{L} = \mathcal{L}_{\mathcal{D}}(W) + \mathcal{L}_W(W)$$

$$= \sum_{m=1}^M \frac{1}{2} \|W^T \mathbf{x}^{(m)} + b - \mathbf{y}^{(m)}\|_2^2 + \frac{\lambda}{2} \sum_{n=1}^N \|W_n\|_2^2$$



Outline

- **Probabilistic interpretation of linear regression**
 - Maximum likelihood estimation (MLE)
 - Maximum a posteriori (MAP) estimation
- **Learning objective:**
 - Understand the relationship between Gaussian and squared L2 loss
 - Understand the probabilistic assumptions we make in modelling and their computational gain
 - Know how to derive MLE and MAP estimation from some assumptions about the likelihood and the prior
 - e.g. Given likelihood and priors are independent Gaussian, derive the L2 penalized loss
 - Understand how to derive learning algorithms from MLE and MAP
 - 1. Make some assumptions about the likelihood and prior
 - 2. Derive the loss function from log likelihood and log prior
 - 3. Minimize the loss to find the parameters

Outline

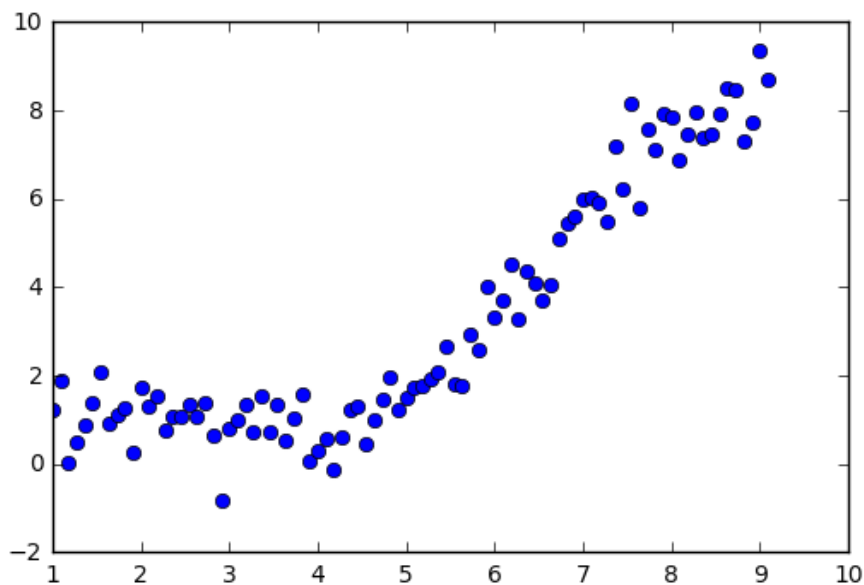
- Probabilistic interpretation of linear regression
 - Maximum likelihood estimation (MLE)
 - Maximum a posteriori (MAP) estimation
- **Bias-variance trade-off**
- Linear basis function models
- Classification

Outline

- **Bias-variance trade-off**
- Prerequisite:
 - Comfortable with: multivariate calculus; conditional expectation; expectation of continuous RVs; variance of continuous RVs; L2 norm; complete-the-square for quadratic function
- Learning objective:
 - Understand the expected loss function
 - Understand the optimal regressor under an expected loss function (The theoretical best thing to do to solve a regression task)
 - Obtain intuition about conditional mean, conditional median and conditional mode and the corresponding loss functions
 - Understand the bias-variance trade-off of a model formally under the expected loss framework
 - Know how to use complete-the-square to derive the theoretical results under squared l2 loss

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 scalar target y with joint probability distribution: $P(\mathbf{x}, y)$
- Our goal is to predict target y given a new value for \mathbf{x} :
 - for regression: y is a real-valued continuous target
 - for classification: y is a categorical variable representing class labels



We would like to answer the question:

What is the optimal curve to fit to this data?

Regression

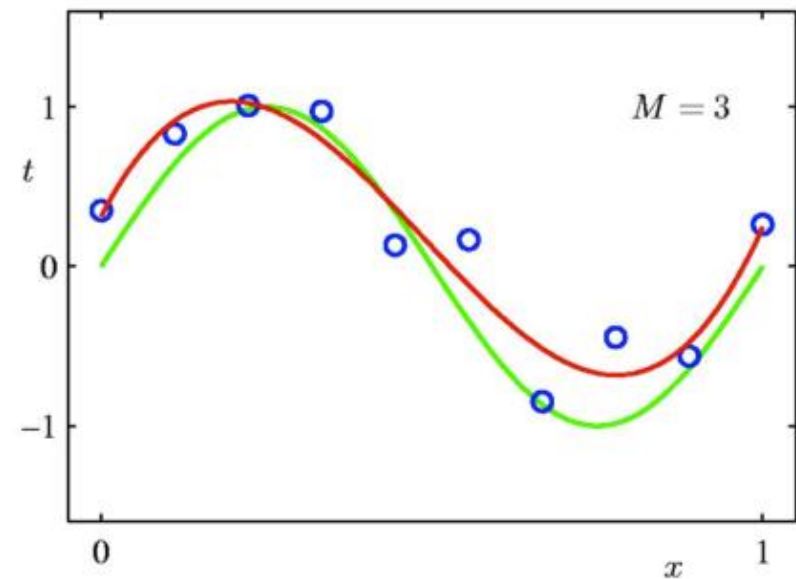
- Let $\mathbf{x} \in \mathbb{R}^N$ denote a real valued input vector and $y \in \mathbb{R}$ denote a real valued random target (output) variable.
- The decision step consists of finding an estimate $\hat{y}(\mathbf{x})$ for each input \mathbf{x} .
- The average, or expected, loss is given by:

$$\mathbb{E}[\mathcal{L}] = \int \int \mathcal{L}(y, \hat{y}(\mathbf{x})) P(\mathbf{x}, y) d\mathbf{x} dy$$

- If we use squared error loss function:

$$\mathbb{E}[\mathcal{L}] = \int \int (y - \hat{y}(\mathbf{x}))^2 P(\mathbf{x}, y) d\mathbf{x} dy$$

We will study optimality using the expected loss framework



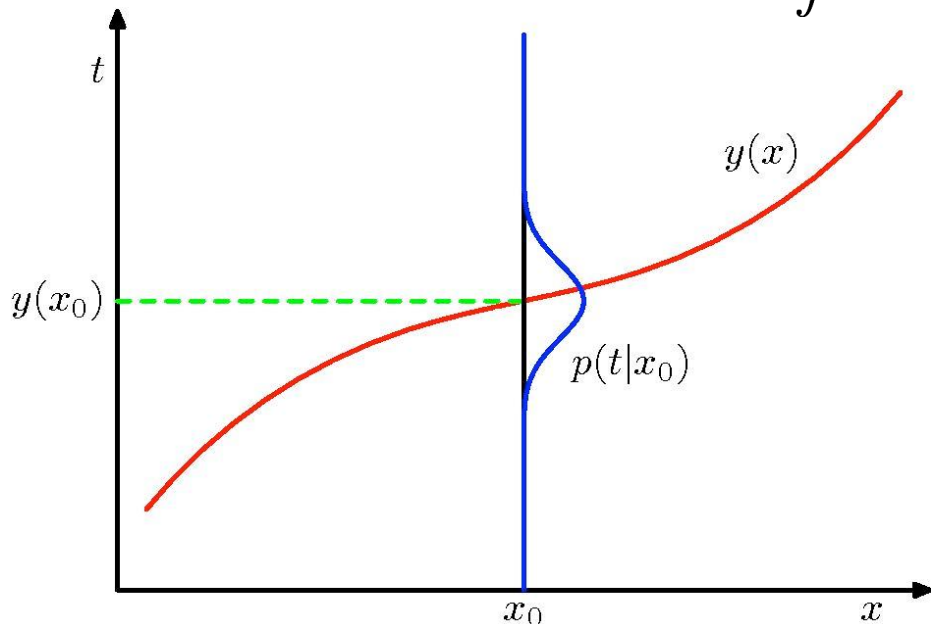
Squared Error Loss Function

- If we use squared error loss (L2 loss), we obtain:

$$\mathbb{E}[\mathcal{L}] = \int \int (y - \hat{y}(\mathbf{x}))^2 P(\mathbf{x}, y) d\mathbf{x} dy$$

- Our goal is to choose $\hat{y}(\mathbf{x})$ so as to minimize the expected squared loss.
- The **optimal** solution (if we assume a completely flexible function) is the conditional mean:

$$\hat{y}(\mathbf{x}) = \int y P(y|\mathbf{x}) dy = \mathbf{E}[y|\mathbf{x}]$$



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

Squared Error Loss Function

- If we use squared error loss (L2 loss), we obtain:

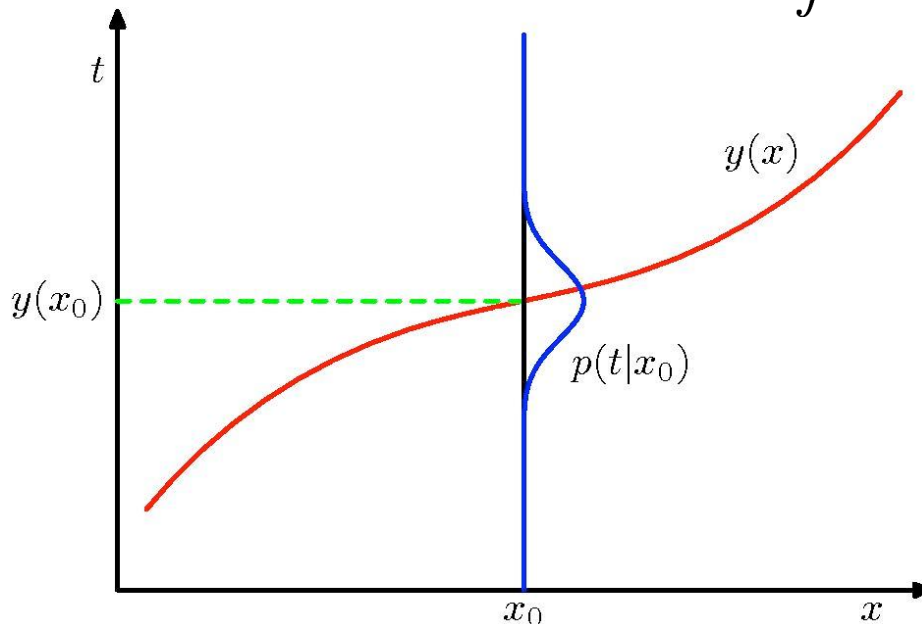
$$\mathbb{E}[\mathcal{L}] = \int \int (y - \hat{y}(\mathbf{x}))^2 P(\mathbf{x}, y) d\mathbf{x} dy$$

- Our goal is to choose $\hat{y}(\mathbf{x})$ so as to minimize the expected squared loss.
- The **optimal** solution (if we assume a completely flexible function) is the conditional mean:

$$\hat{y}(\mathbf{x}) = \int y P(y|\mathbf{x}) dy = \mathbf{E}[y|\mathbf{x}]$$

Derivation 1: use calculus of variate (beyond this course)

$$\frac{\partial \mathbb{E}[\mathcal{L}]}{\partial \hat{y}(\mathbf{x})} = \int \int \frac{\partial}{\partial \hat{y}(\mathbf{x})} (y - \hat{y}(\mathbf{x}))^2 P(\mathbf{x}, y) d\mathbf{x} dy = 0$$



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

Squared Error Loss Function

- We can derive the optimality result by first add and subtract the conditional expected targets:

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

Derivation 2: complete-the-square

↑
This term is zero under
expectation of $P(y|\mathbf{x})$

- Plugging into expected loss:

$$\mathbb{E}[\mathcal{L}] = \int (\hat{y}(\mathbf{x}) - \mathbb{E}[y|\mathbf{x}])^2 P(\mathbf{x}) d\mathbf{x} + \int \text{var}[y|\mathbf{x}] P(\mathbf{x}) d\mathbf{x}$$

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

intrinsic variability of the
target values.

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

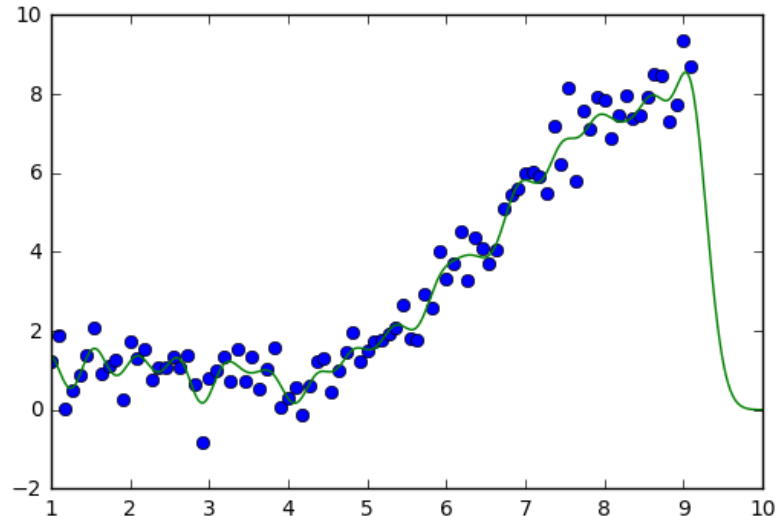
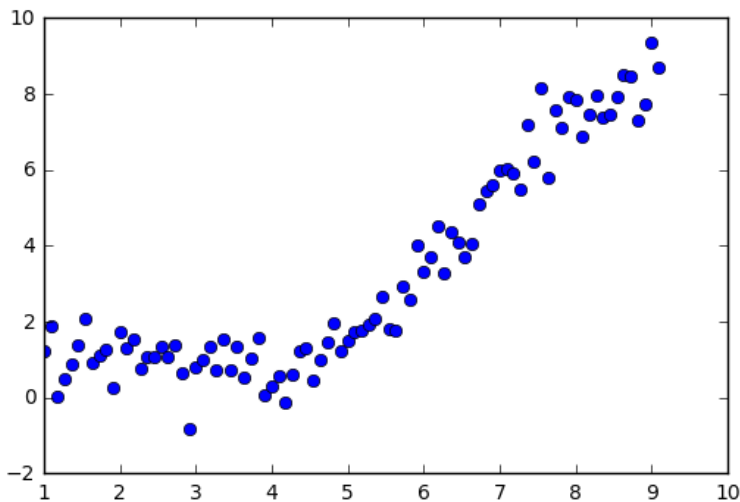
Squared Error Loss Function

- The only terms that are relevant to our model in the expected loss is:

$$\mathbb{E}[\mathcal{L}] = \int (\hat{y}(\mathbf{x}) - \mathbb{E}[y|\mathbf{x}])^2 P(\mathbf{x}) d\mathbf{x} + \text{data noise}$$

- Therefore, the **optimal** regression model under L2 loss is a model that predicts the conditional mean of the target values

$$\hat{y}(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}]$$



Other Loss Functions

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

$$\mathbb{E}[\mathcal{L}] = \int \int \|y - \hat{y}(\mathbf{x})\|_q^q P(\mathbf{x}, y) d\mathbf{x} dy$$

- 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 $q = 0$.

Other Loss Functions

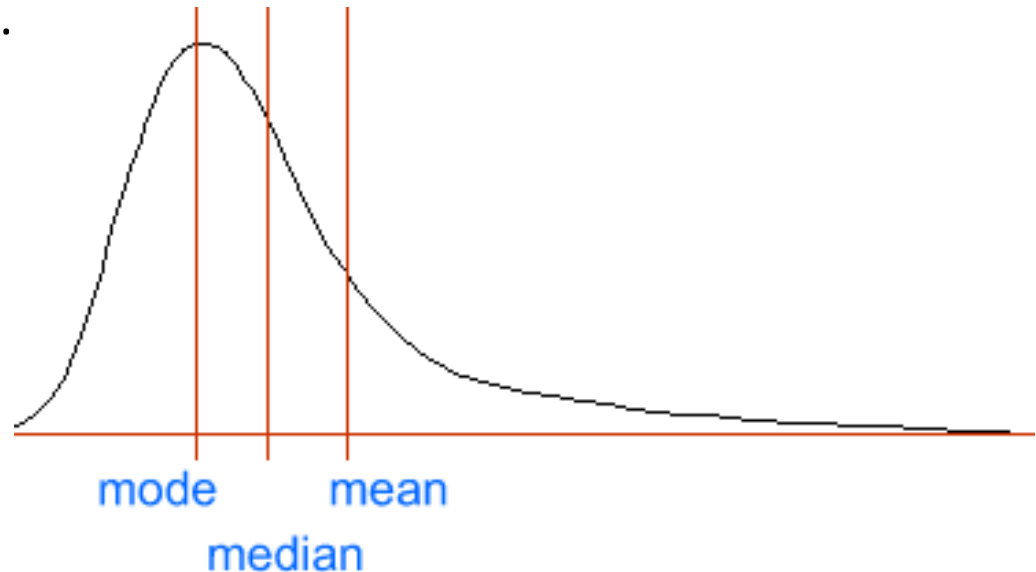
- Simple generalization of the squared loss, called the *Minkowski* loss:

$$\mathbb{E}[\mathcal{L}] = \int \int \|y - \hat{y}(\mathbf{x})\|_q^q P(\mathbf{x}, y) d\mathbf{x} dy$$

- 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 $q = 0$.

Intuition: mean
can be a very bad
idea

- Smaller q is robust to outliers



Other Loss Functions

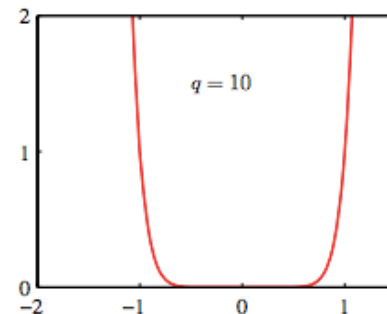
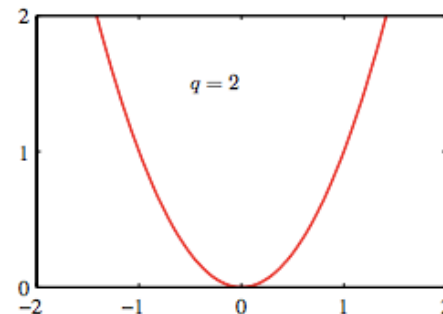
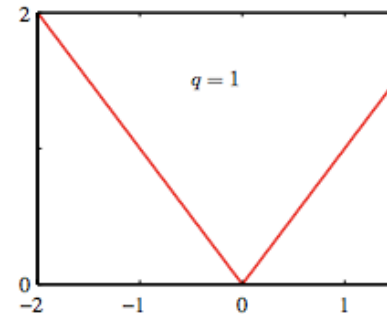
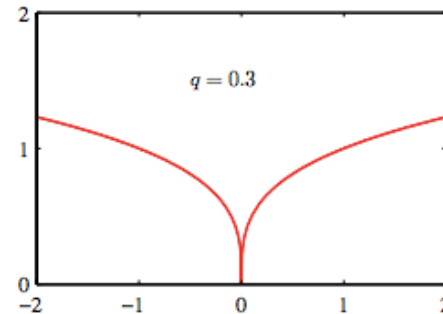
- Simple generalization of the squared loss, called the *Minkowski* loss:

$$\mathbb{E}[\mathcal{L}] = \int \int \|y - \hat{y}(\mathbf{x})\|_q^q P(\mathbf{x}, y) d\mathbf{x} dy$$

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

Intuition: plot loss function w.r.t. the difference

- Smaller q is robust to outliers



Other Loss Functions

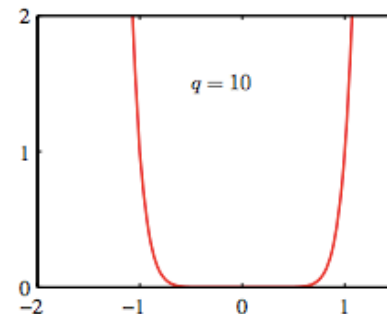
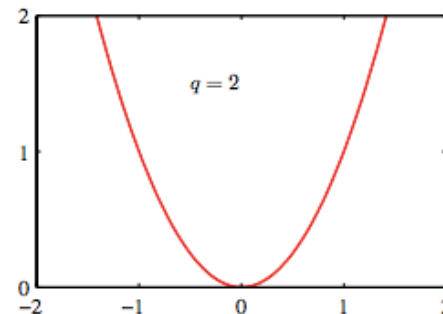
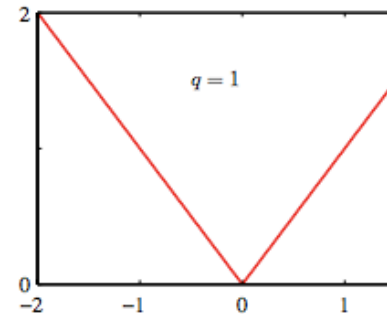
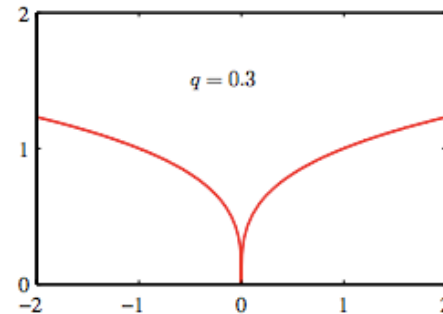
- Simple generalization of the squared loss, called the *Minkowski* loss:

$$\mathbb{E}[\mathcal{L}] = \int \int \|y - \hat{y}(\mathbf{x})\|_q^q P(\mathbf{x}, y) d\mathbf{x} dy$$

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

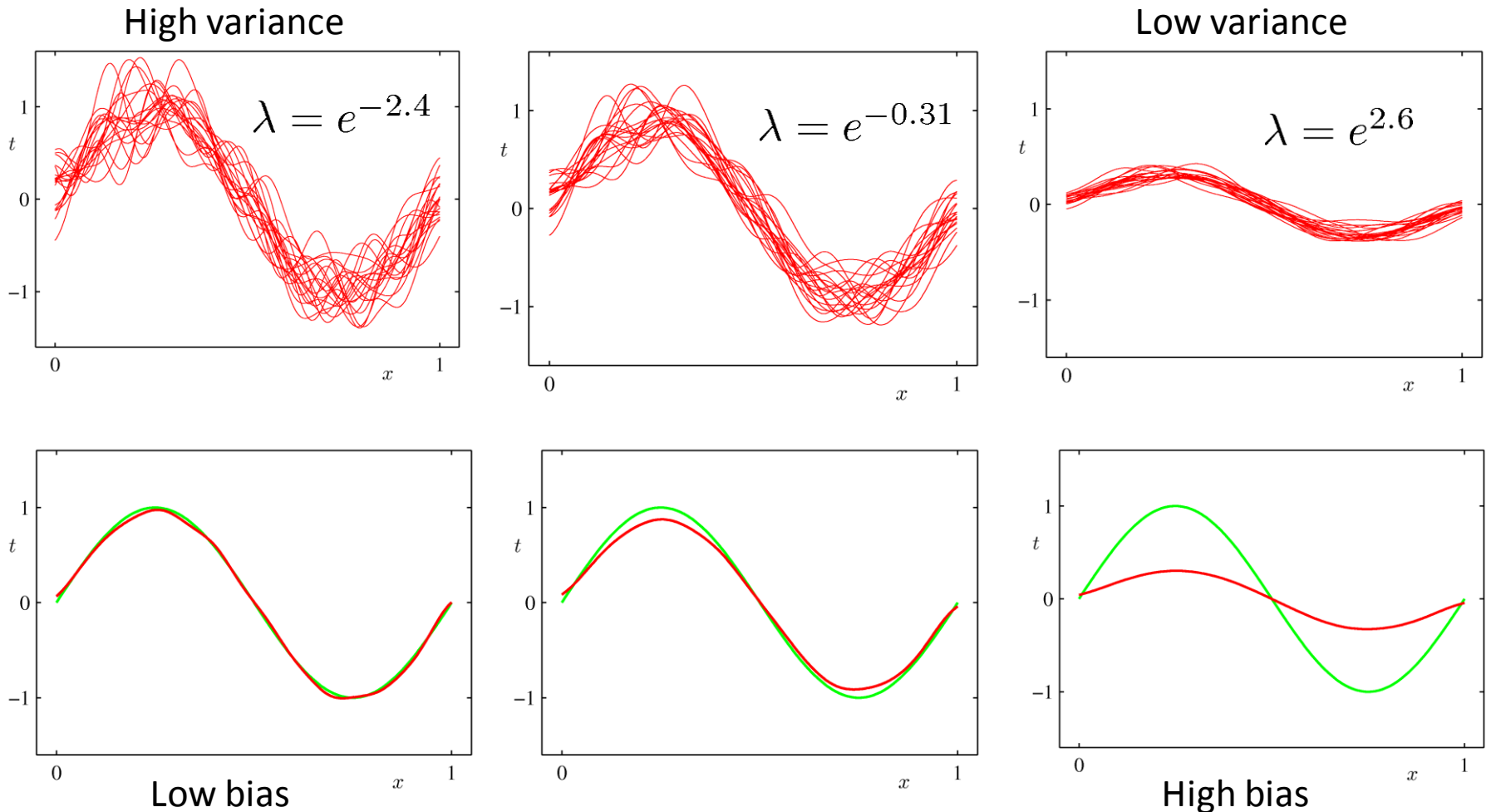
Intuition: plot loss function w.r.t. the difference

- Smaller q is robust to outliers

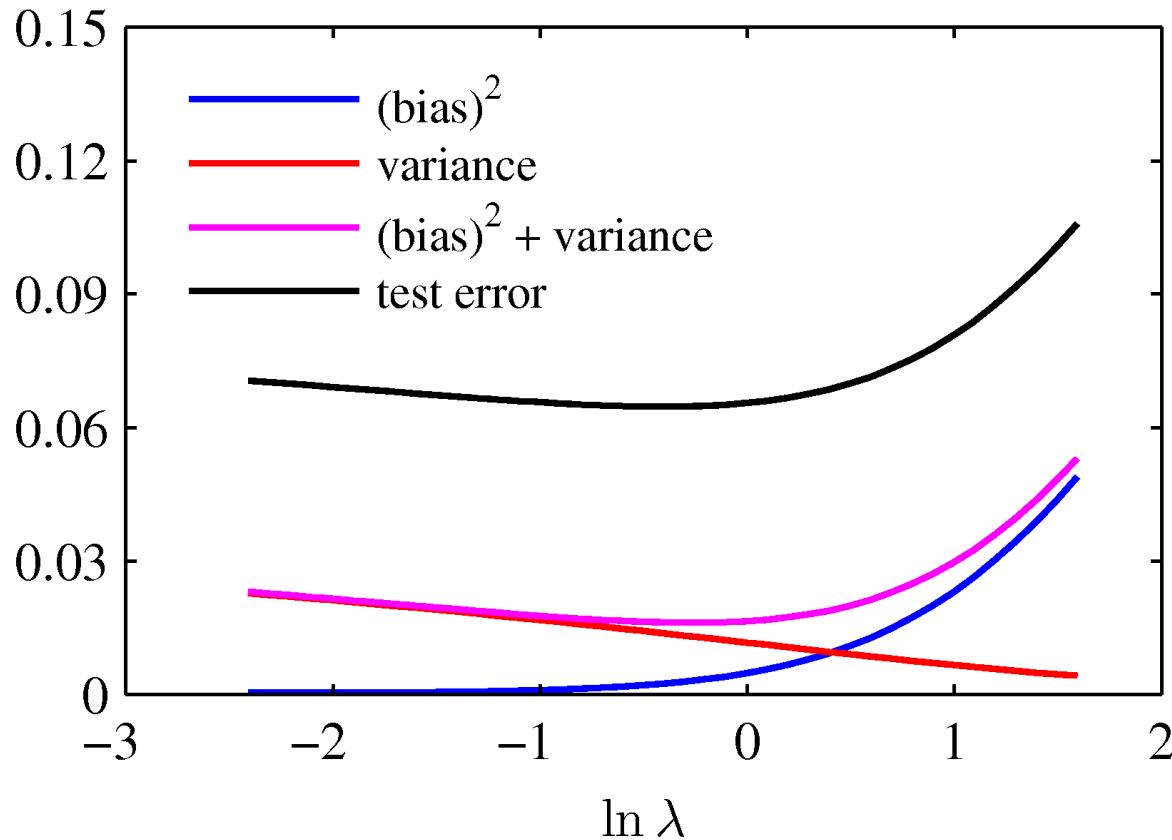


Bias-Variance Trade-off

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



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.

Bias-variance trade-off

- Remember the expected loss

$$\mathbb{E}[\mathcal{L}] = \int (\hat{y}(\mathbf{x}) - \mathbb{E}[y|\mathbf{x}])^2 P(\mathbf{x}) d\mathbf{x} + \text{data noise}$$

- We can derive another important result from the L2 loss by noticing:

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

Bias-variance trade-off

- Remember the expected loss

$$\mathbb{E}[\mathcal{L}] = \int (\hat{y}(\mathbf{x}) - \mathbb{E}[y|\mathbf{x}])^2 P(\mathbf{x}) d\mathbf{x} + \text{data noise}$$

- We can derive another important result from the L2 loss by noticing:

$$\begin{aligned} & \int (\hat{y}(\mathbf{x}) - \mathbb{E}[y|\mathbf{x}])^2 P(\mathbf{x}) d\mathbf{x} \\ &= \int (\mathbb{E}[\hat{y}(\mathbf{x})] - \mathbb{E}[y|\mathbf{x}])^2 P(\mathbf{x}) d\mathbf{x} + \int (\hat{y}(\mathbf{x}) - \mathbb{E}[\hat{y}(\mathbf{x})])^2 P(\mathbf{x}) d\mathbf{x} \end{aligned}$$

↑
This term is zero under
expectation of $P(y|\mathbf{x})$

Bias-variance trade-off

- Remember the expected loss

$$\mathbb{E}[\mathcal{L}] = \int (\hat{y}(\mathbf{x}) - \mathbb{E}[y|\mathbf{x}])^2 P(\mathbf{x}) d\mathbf{x} + \text{data noise}$$

- We can derive another important result from the L2 loss by noticing:

$$\begin{aligned} & \int (\hat{y}(\mathbf{x}) - \mathbb{E}[y|\mathbf{x}])^2 P(\mathbf{x}) d\mathbf{x} \\ &= \int (\underbrace{\mathbb{E}[\hat{y}(\mathbf{x})]}_{\substack{\uparrow \\ \text{bias term}}} - \mathbb{E}[y|\mathbf{x}])^2 P(\mathbf{x}) d\mathbf{x} + \int (\hat{y}(\mathbf{x}) - \underbrace{\mathbb{E}[\hat{y}(\mathbf{x})]}_{\substack{\uparrow \\ \text{Variance of the model prediction}}})^2 P(\mathbf{x}) d\mathbf{x} \end{aligned}$$

Bias-variance trade-off

- Remember the expected loss

$$\mathbb{E}[\mathcal{L}] = \int (\hat{y}(\mathbf{x}) - \mathbb{E}[y|\mathbf{x}])^2 P(\mathbf{x}) d\mathbf{x} + \text{data noise}$$

- We can derive another important result from the L2 loss by noticing:

$$\mathbb{E}[\mathcal{L}] = \text{model bias} + \text{model variance} + \text{data noise}$$

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 M , we could generate new training sets, M , by sampling examples from D uniformly and with replacement.
 - This is called bagging and it works quite well in practice.
- Given enough computation, we can use the Bayesian framework:
 - Combine the predictions of many models using the posterior probability of each parameter vector as the combination weight.

Outline

- **Bias-variance trade-off**
- Learning objective:
 - Understand the expected loss function
 - Understand the optimal regressor under an expected loss function
(The theoretical best thing to do to solve a regression task)
 - Obtain intuition about conditional mean, conditional median and conditional mode and the corresponding loss functions
 - Understand the bias-variance trade-off of a model formally under the expected loss framework
 - Know how to use complete-the-square to derive the theoretical results under squared l_2 loss

Outline

- Probabilistic interpretation of linear regression
 - Maximum likelihood estimation (MLE)
 - Maximum a posteriori (MAP) estimation
- Bias-variance trade-off
- **Linear basis function models**
- Classification

Outline

- **Linear basis function models**
- Prerequisite:
 - Comfortable with: matrix vector products; linear algebra; linear projection; column space
- Learning objective:
 - Understand intuition behind feature expansion and why infinite features can represent any function
 - Understand any transformation to the dataset does not effect the property of the models

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:

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

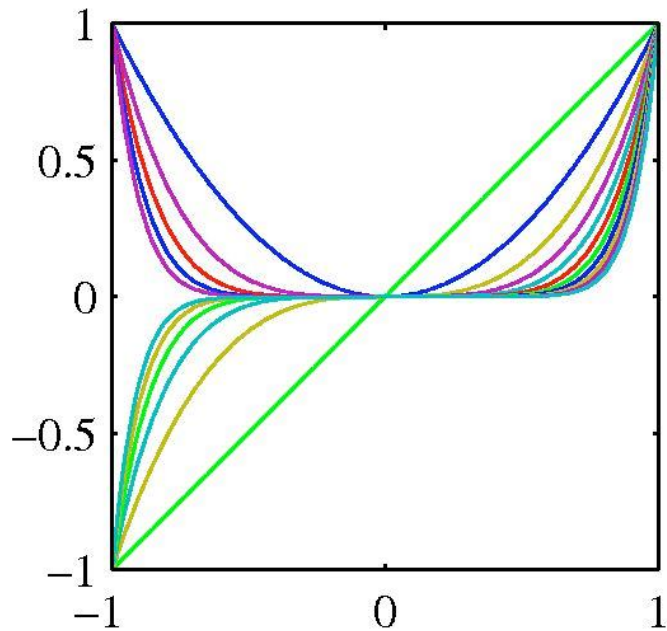
where $\phi_j(\mathbf{x})$ are known as basis functions.

- Typically $\phi_0(\mathbf{x}) = 1$ so that w_0 acts as a bias (or intercept).
- In the simplest case, we use linear basis 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:

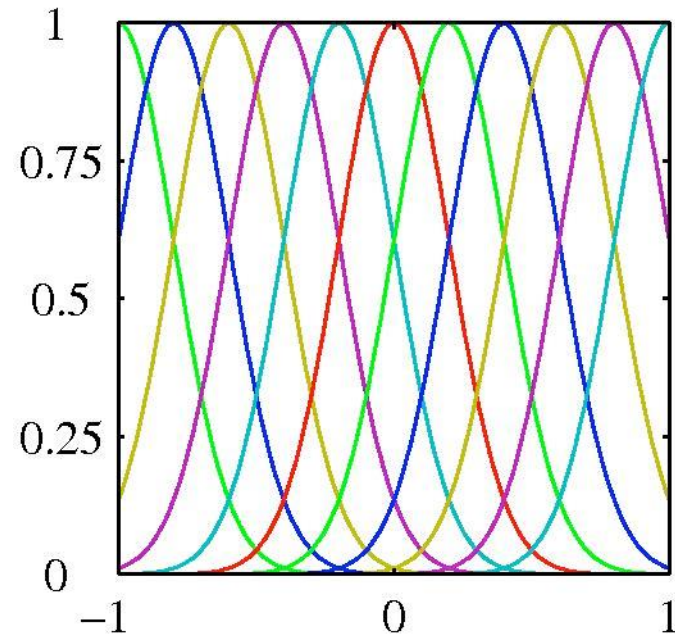
$$\phi_j(x) = x^j.$$



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

Gaussian basis functions:

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

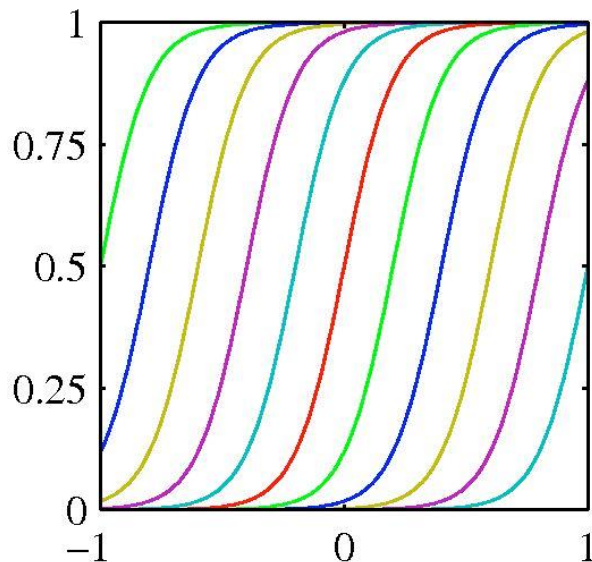


Basis functions are local: small changes in \mathbf{x} only affect nearby basis functions.
 μ_j and s control location and scale (width).

Linear Basis Function Models

Sigmoidal basis functions

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

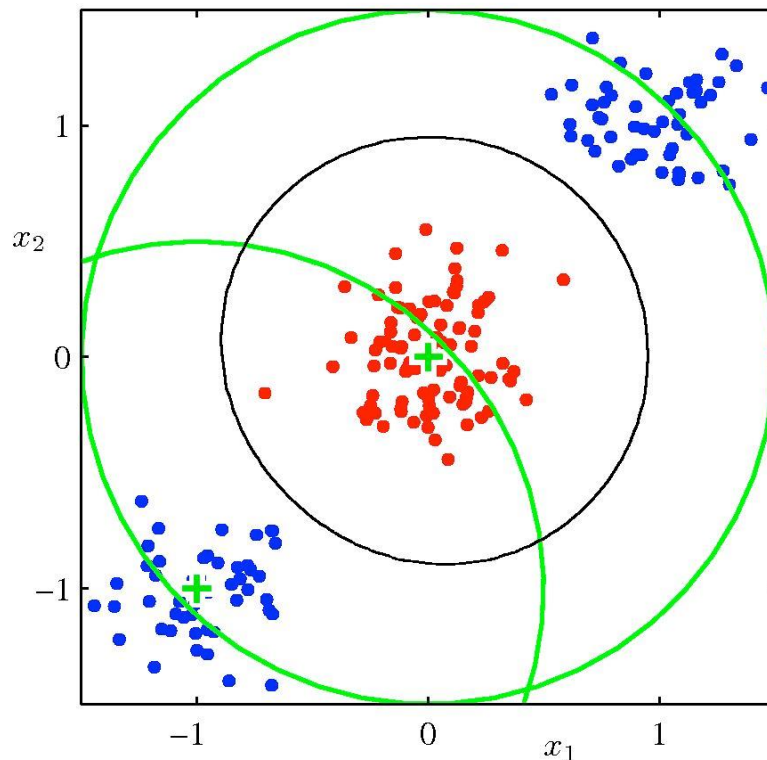


Basis functions are local: small changes in \mathbf{x} only affect nearby basis functions. μ_j and s control location and scale (slope).

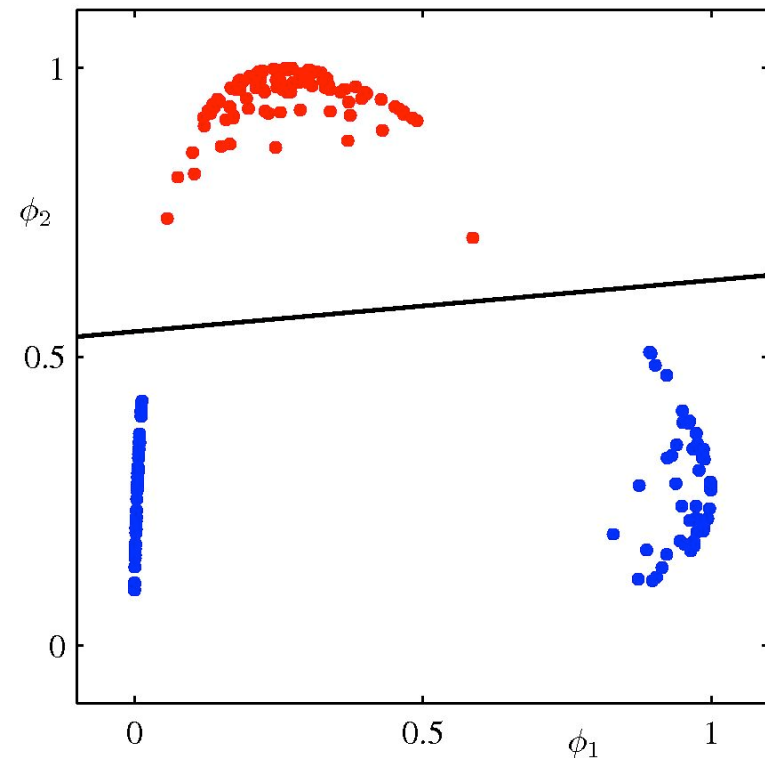
- Decision boundaries will be linear in the feature space ϕ , but would correspond to nonlinear boundaries in the original input space \mathbf{x} .
- Classes that are linearly separable in the feature space $\phi(\mathbf{x})$ need not be linearly separable in the original input space.

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 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 for LBFMs

- 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 for LBFMs

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:

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

MLE Optimal Solution for LBFMs

Differentiating and setting to zero yields:

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

Solving for \mathbf{w} , we get:

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

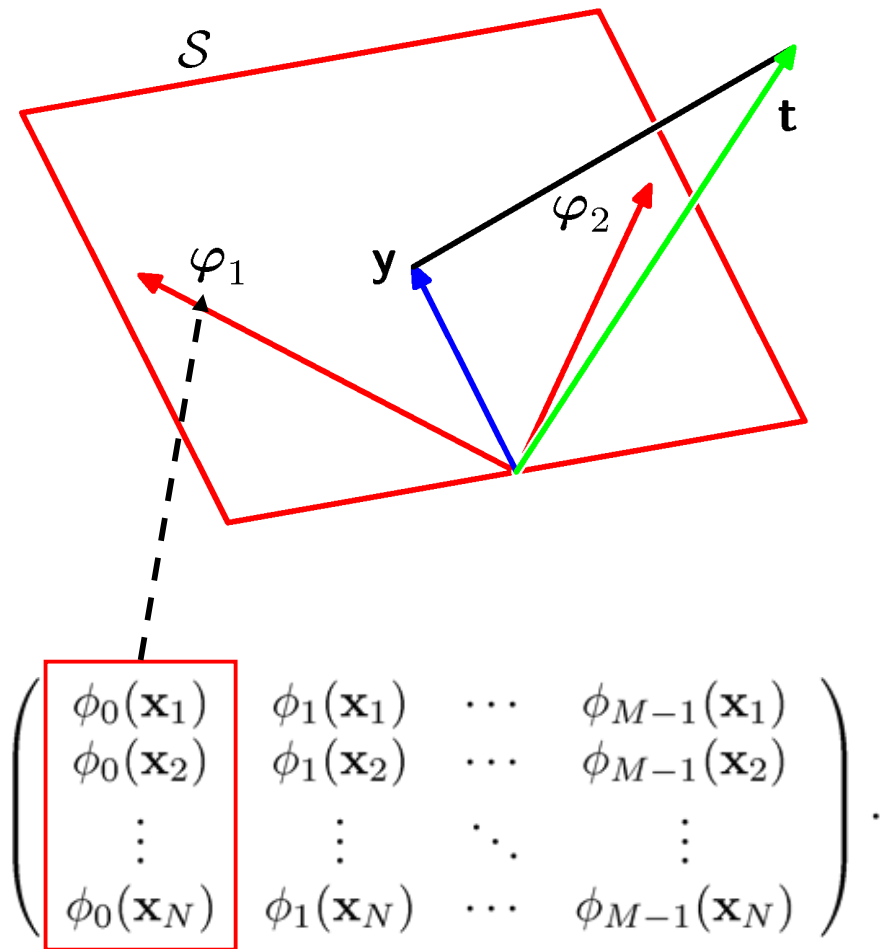
The Moore-Penrose pseudo-inverse, $\boldsymbol{\Phi}^\dagger$.

where $\boldsymbol{\Phi}$ is 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 $\mathbf{t} = [t_1, t_2, \dots, t_N]^T$ 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 $M < N$ then the M basis functions, $\phi_j(\mathbf{x}_n)$, 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 .

Iterative Learning for LBFMs

- The training data examples are presented one at a time, and the model parameters 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 at time t (indicated by a blue arrow)
- η : learning rate (indicated by a blue arrow)
- ∇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 and LBFMs

- 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:

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

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).

Outline

- Probabilistic interpretation of linear regression
 - Maximum likelihood estimation (MLE)
 - Maximum a posteriori (MAP) estimation
- Bias-variance trade-off
- Linear basis function models
- **Classification**

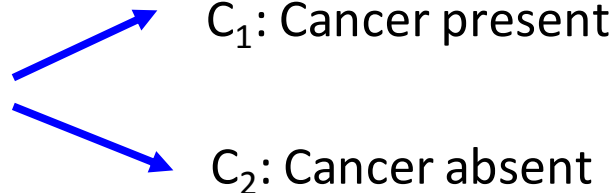
Example: Classification

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: $t=0$ corresponds to class C_1 , and $t=1$ corresponds to C_2 .

Inference Problem: Determine the joint distribution $p(\mathbf{x}, C_k)$ 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

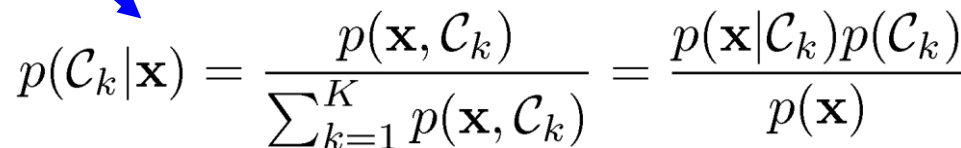
Informally: Given a new X-ray image, our goal is to decide which of the two classes that image should be assigned to.

- 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

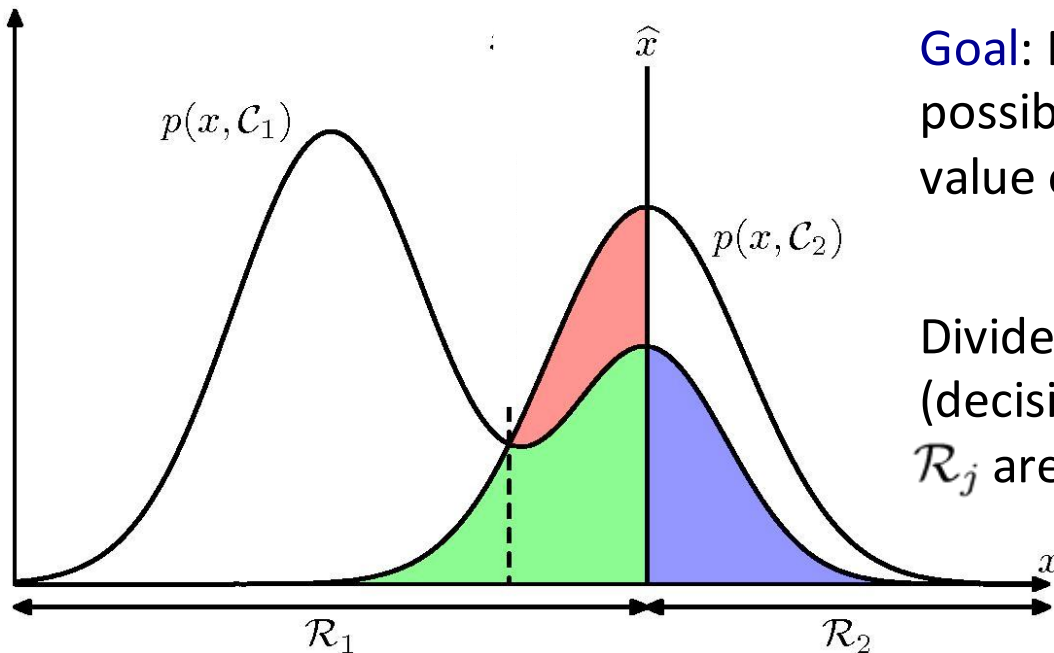
prior probability for class C_k


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

Bayes' Rule

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

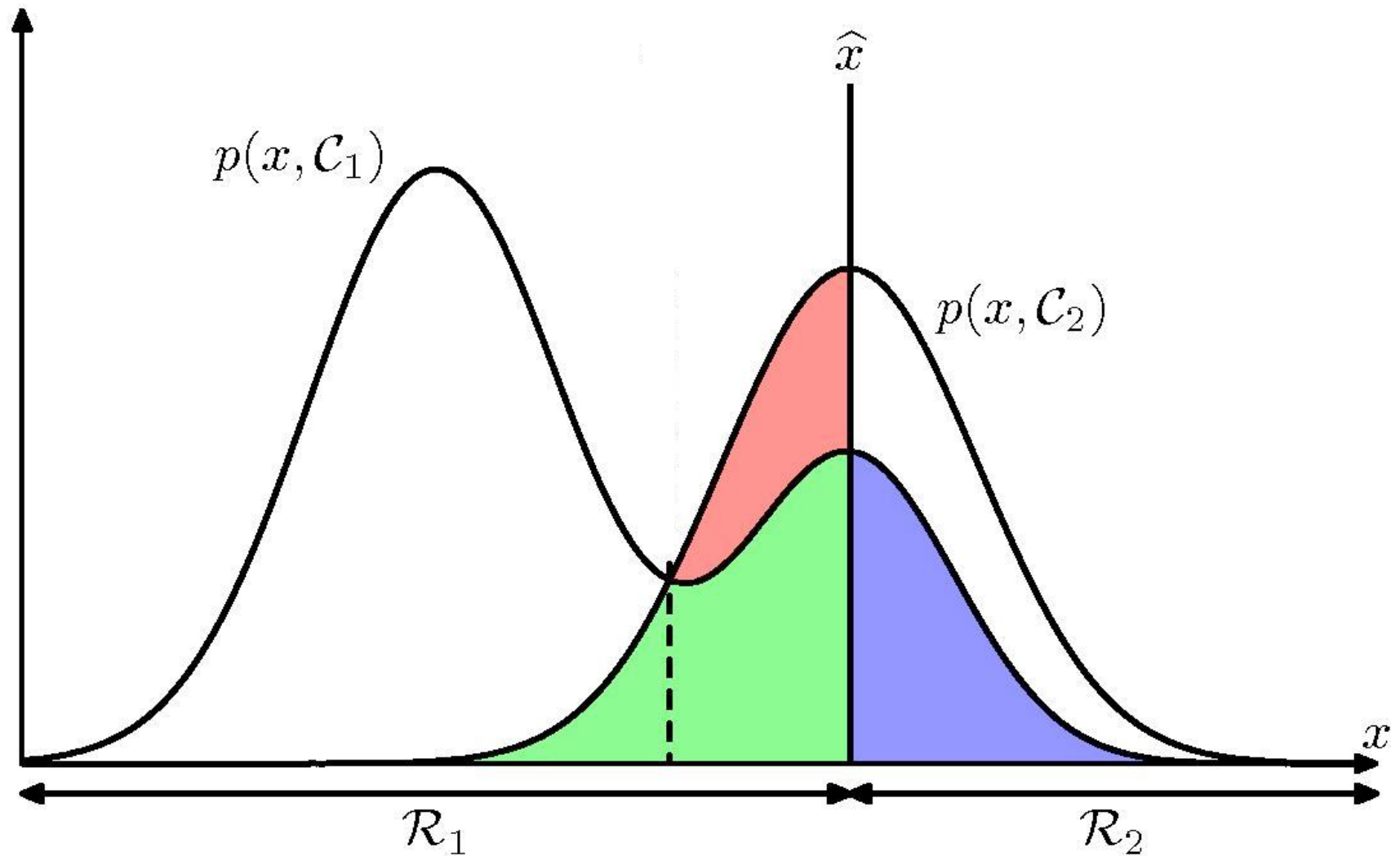
Divide the input space into regions \mathcal{R}_j (decision regions), such that all points in \mathcal{R}_j are assigned to class \mathcal{C}_j .

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

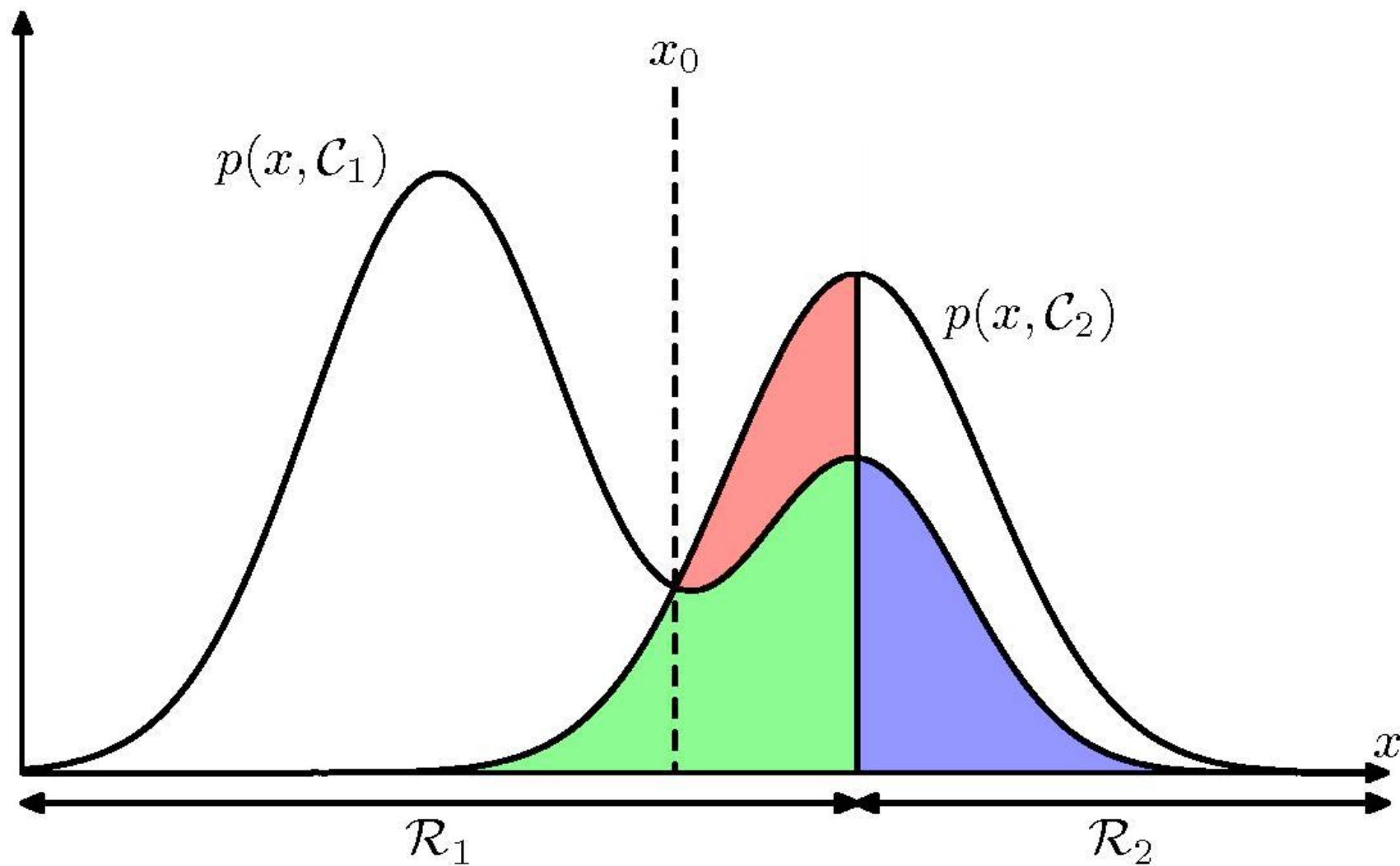
blue region: input belongs to class \mathcal{C}_1 , but is assigned to \mathcal{C}_2

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

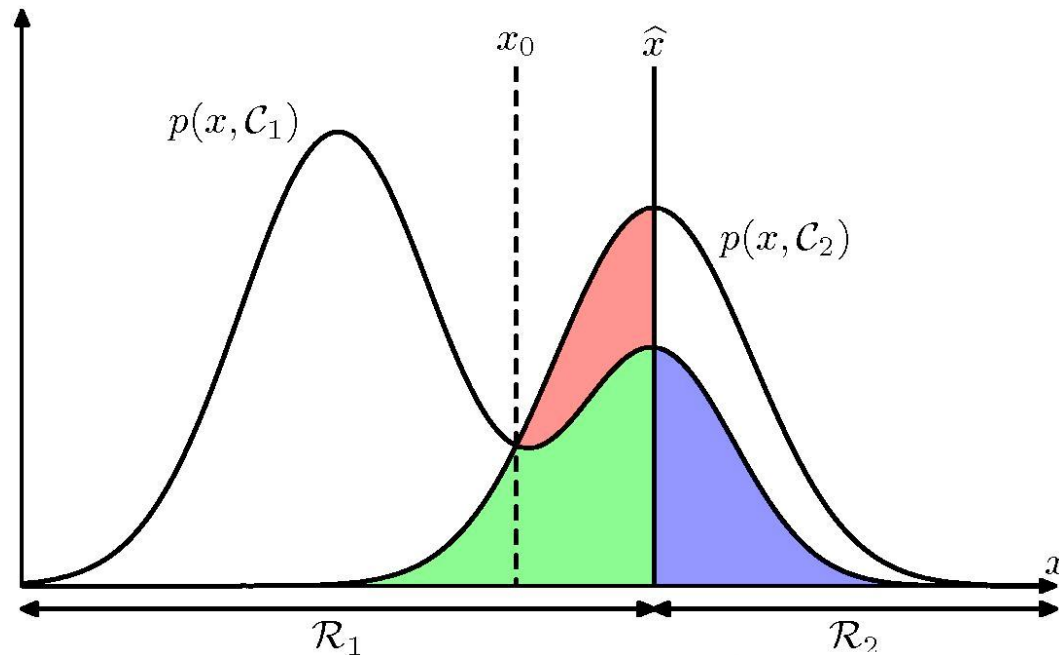
Minimizing Misclassification Rate



Minimizing Misclassification Rate



Minimizing Misclassification Rate



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

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 \mathbf{x} to the class for which the posterior probability $p(\mathcal{C}_k|\mathbf{x})$ 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
We incur a loss of L_{kj} (k, j element of a loss matrix).

Consider medical diagnosis example: example of a loss matrix:

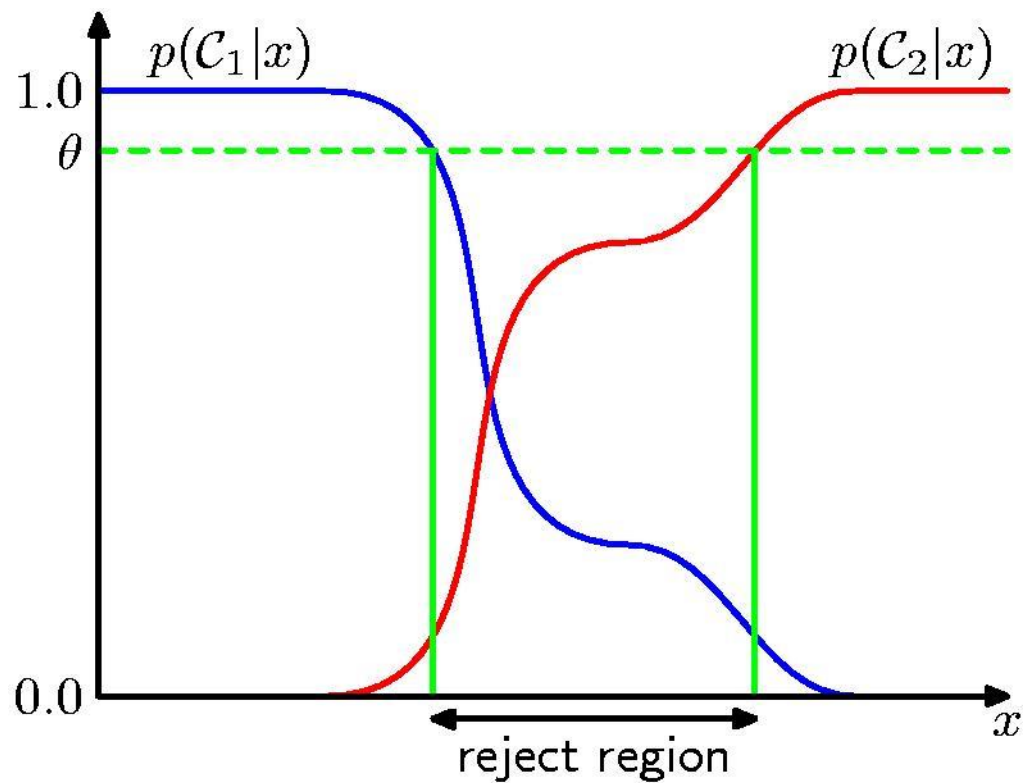
		Decision	
		cancer	normal
Truth	cancer	0	1000
	normal	1	0

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



Outline

- Probabilistic interpretation of linear regression
 - Maximum likelihood estimation (MLE)
 - Maximum a posteriori (MAP) estimation
- Bias-variance trade-off
- Linear basis function models
- Classification
- **Cross validation**

Cross Validation

If the data is plentiful, we can divide the dataset into three subsets:

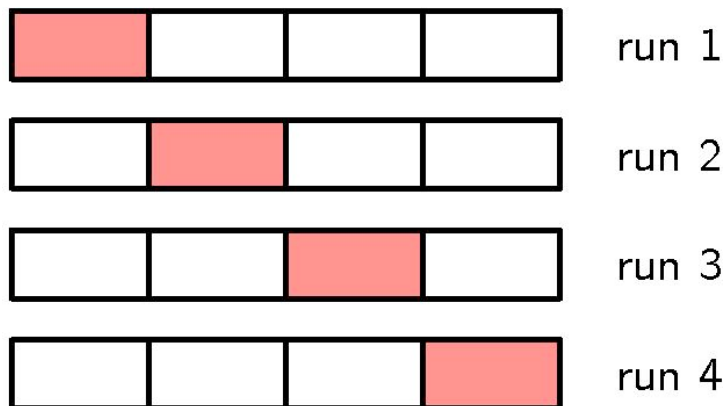
- **Training Data:** used to fitting/learning the parameters of the model.
- **Validation Data:** not used for learning but for selecting the model, or choosing the amount of regularization that works best. E.g. the weight decay coefficient
- **Test Data:** used to get performance of the final model.

For many applications, the supply of data for training and testing is limited.

To build good models, we may want to use as much training data as possible.

If the validation set is small, we get noisy estimate of the predictive performance.

S fold cross-validation



- The data is partitioned into S groups.
- Then $S-1$ of the groups are used for training the model, which is evaluated on the remaining group.
- Repeat procedure for all S possible choices of the held-out group.
- Performance from the S runs are averaged.