Gaussian Process Methods in Machine Learning

Jonathan Scarlett
scarlett@comp.nus.edu.sg

Lecture 0: Bayesian Modeling and Regression

CS6216, Semester 1, AY2021/22

Outline of Lectures

- **Lecture 0: Bayesian Modeling and Regression**
- Lecture 1: Gaussian Processes, Kernels, and Regression
- Lecture 2: Optimization with Gaussian Processes
- Lecture 3: Advanced Bayesian Optimization Methods
- Lecture 4: GP Methods in Non-Bayesian Settings

Outline: This Lecture

- This lecture
  1. Linear regression
  2. Non-linear regression
  3. Feature spaces and kernels
  4. Parametric vs. non-parametric regression
  5. Optimization

Background Material

• It is important that you are comfortable with the basics of probability and linear algebra. See the pre-requisite material document for a very brief summary, or contact me if you would like any pointers to more detailed material.

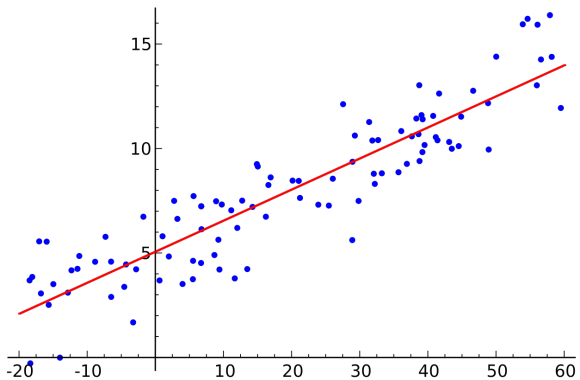
• First we will recap the concepts of linear regression, ridge regression, features, and kernels fairly quickly (since I am assuming you have seen it before). If you need a more detailed summary of these, see the CS5339 lecture notes (Lectures 4 and 5):

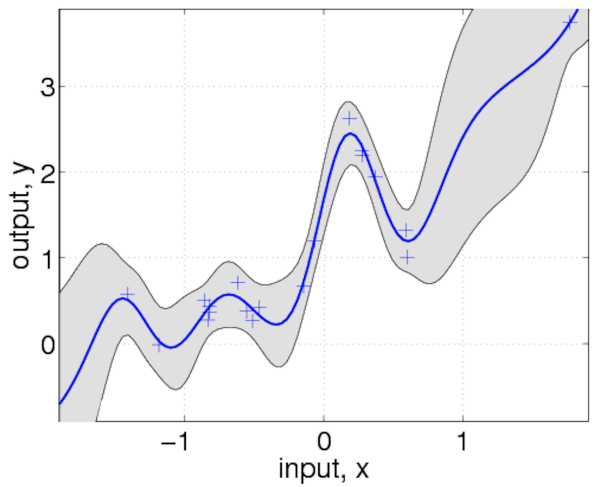
https://www.comp.nus.edu.sg/~scarlett/CS5339_notes/

Video recordings are also available (see LumiNUS for the link)

Regression Analysis

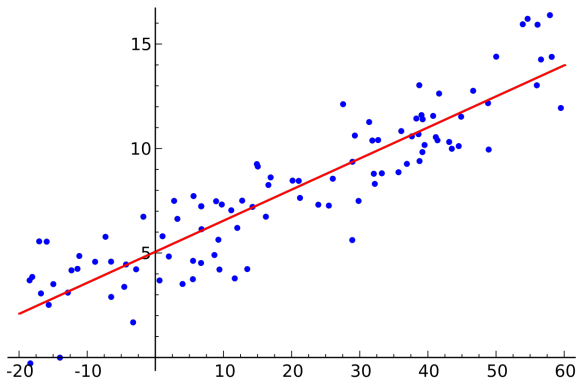
- **Goal:**
  - Determine relationship between input vector $\mathbf{x} \in \mathbb{R}^d$ and output variable $y \in \mathbb{R}$
  - Given samples $(\mathbf{x}_t, y_t)$ for $t = 1, \ldots, n$ (this is the data set)

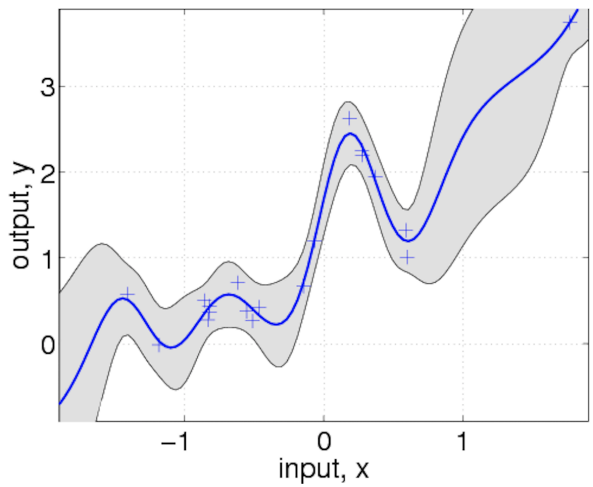




Regression Analysis

- **Goal:**
  - Determine relationship between input vector $\mathbf{x} \in \mathbb{R}^d$ and output variable $y \in \mathbb{R}$
  - Given samples $(\mathbf{x}_t, y_t)$ for $t = 1, \ldots, n$ (this is the data set)





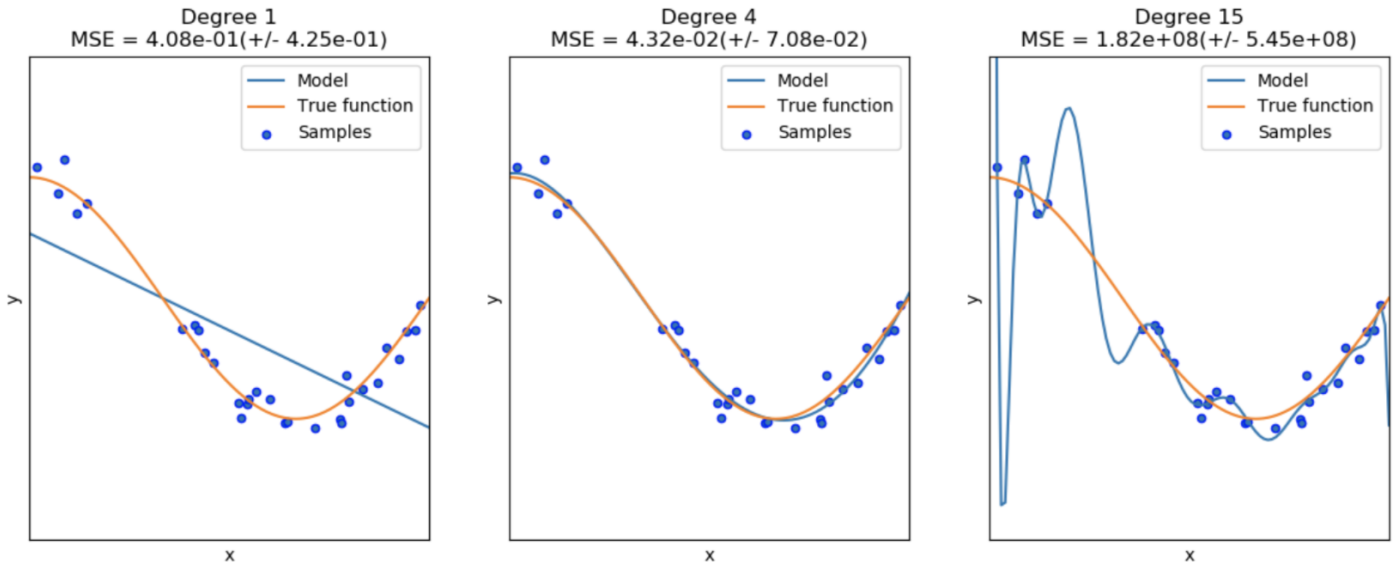
- **Motivation:**
  - Often permits interpretability (e.g., factors impacting a medical diagnosis)
  - Useful for prediction (e.g., predict financial value of an asset)

Examples

- Given a data set $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$, we want to learn the relationship between $\mathbf{x}$ and y, and be able to predict the label y' corresponding to a new input $\mathbf{x}'$
  - e.g., y is the next stock price, $\mathbf{x}$ contains a number of previous prices
  - e.g., y is the temperature at location $\mathbf{x}$
  - e.g., y measures the effectiveness of a medicine represented by $\mathbf{x}$
  - e.g., y is the number of sales of a product represented by $\mathbf{x}$
  - e.g., y is the age of the person in the image represented by $\mathbf{x}$
  - ...

Challenge

- Choosing a "richer" model is not always a good idea!
- Example from [scikit-learn.com]:



Notes:

- A learning algorithm is only as "good" as the model it is based on
- All models are "wrong", but some models are useful
- Preventing underfitting and overfitting is a significant challenge

Linear Regression

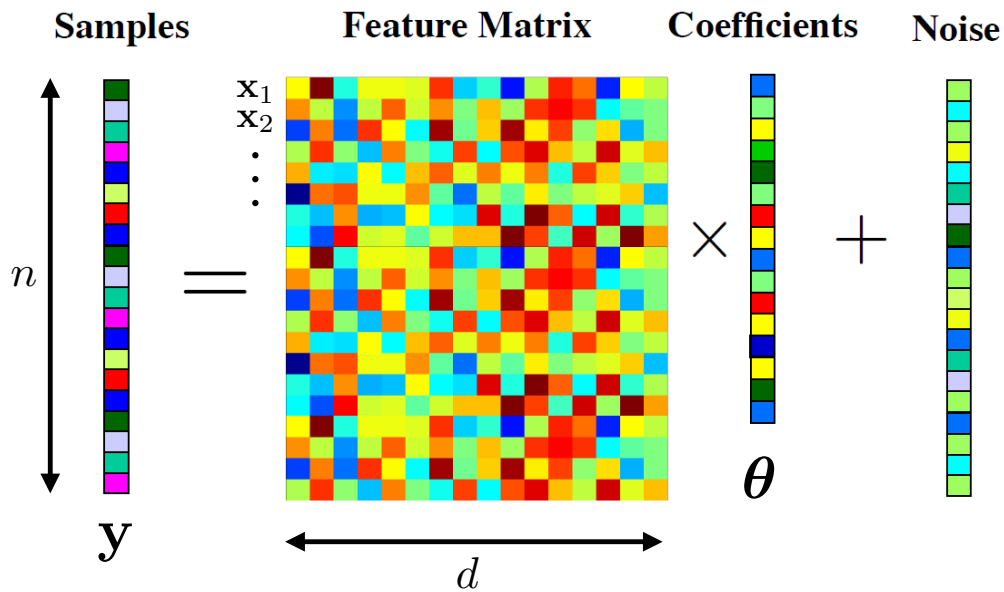
- **Linear regression:**
  - Assume (approximately) linear relation between $\mathbf{x}$ and y:

$$y = \boldsymbol{\theta}^T \mathbf{x} + z$$

$$= \sum_{j=1}^{d} x_j \theta_j + z$$

for some unknown parameter $\boldsymbol{\theta} \in \mathbb{R}^d$, where z is possible noise

  - Problem: Most relations in practical problems are highly non-linear



Linear Regression Approaches

- If we are given a data set $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$, how do we learn the "best" parameter $\boldsymbol{\theta}$ achieving $y_t \approx \boldsymbol{\theta}^T \mathbf{x}_t$?

- Several related approaches:
  1. Model each $(\mathbf{x}_t, y_t)$ as being independently drawn from a distribution $P_{\boldsymbol{\theta}}(\mathbf{x}, y)$ parametrized by $\boldsymbol{\theta}$, and estimate these parameters using maximum likelihood (ML) estimation:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \prod_{t=1}^{n} P_{\boldsymbol{\theta}}(y_t | \mathbf{x}_t)$$

Linear Regression Approaches

- If we are given a data set $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$, how do we learn the "best" parameter $\boldsymbol{\theta}$ achieving $y_t \approx \boldsymbol{\theta}^T \mathbf{x}_t$?

- Several related approaches:
  1. Model each $(\mathbf{x}_t, y_t)$ as being independently drawn from a distribution $P_{\boldsymbol{\theta}}(\mathbf{x}, y)$ parametrized by $\boldsymbol{\theta}$, and estimate these parameters using maximum likelihood (ML) estimation:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \prod_{t=1}^{n} P_{\boldsymbol{\theta}}(y_t|\mathbf{x}_t)$$

  2. Model both $\boldsymbol{\theta}$ and each $(\mathbf{x}_t, y_t)$ as being random, and use Bayesian inference to find $P(\boldsymbol{\theta}|\mathcal{D})$ (more soon!)

Linear Regression Approaches

- If we are given a data set $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$, how do we learn the "best" parameter $\boldsymbol{\theta}$ achieving $y_t \approx \boldsymbol{\theta}^T \mathbf{x}_t$?

- Several related approaches:
  1. Model each $(\mathbf{x}_t, y_t)$ as being independently drawn from a distribution $P_{\boldsymbol{\theta}}(\mathbf{x}, y)$ parametrized by $\boldsymbol{\theta}$, and estimate these parameters using maximum likelihood (ML) estimation:
  $$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \prod_{t=1}^{n} P_{\boldsymbol{\theta}}(y_t|\mathbf{x}_t)$$
  2. Model both $\boldsymbol{\theta}$ and each $(\mathbf{x}_t, y_t)$ as being random, and use Bayesian inference to find $P(\boldsymbol{\theta}|\mathcal{D})$ (more soon!)
  3. Avoid any explicit model on $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$, and simply try to look for a good linear predictor.

Maximum Likelihood Estimation

• Suppose the data set $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$ is known to consist of independent samples generated via

$$y_t = \boldsymbol{\theta}^T \mathbf{x}_t + z_t$$

with $z_t \sim N(0, \sigma^2)$, for some unknown $\boldsymbol{\theta}$.

Maximum Likelihood Estimation

• Suppose the data set $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$ is known to consist of independent samples generated via

$$y_t = \boldsymbol{\theta}^T \mathbf{x}_t + z_t$$

with $z_t \sim N(0, \sigma^2)$, for some unknown $\boldsymbol{\theta}$.

•The associated likelihood function is

$$L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{t=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(- \frac{(y_t - \boldsymbol{\theta}^T \mathbf{x}_t)^2}{2\sigma^2} \right).$$

Maximum Likelihood Estimation

• Suppose the data set $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$ is known to consist of independent samples generated via

$$y_t = \boldsymbol{\theta}^T \mathbf{x}_t + z_t$$

with $z_t \sim N(0, \sigma^2)$, for some unknown $\boldsymbol{\theta}$.

•The associated likelihood function is

$$L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{t=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(- \frac{(y_t - \boldsymbol{\theta}^T \mathbf{x}_t)^2}{2\sigma^2} \right).$$

• Maximizing L is equivalent to maximizing its log, but the latter is more convenient to work with:

$$\log L(\boldsymbol{\theta}; \mathcal{D}) = \text{const.} - \frac{1}{2\sigma^2} \sum_{t=1}^{n} (y_t - \boldsymbol{\theta}^T \mathbf{x}_t)^2, \tag{1}$$

where const. represents a term that does not depend on $\boldsymbol{\theta}$.

Least Squares

- By the previous slide, maximizing likelihood is equivalent to least squares:

$$\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \mathcal{D}) \iff \min_{\boldsymbol{\theta}} \sum_{t=1}^{n} (y_t - \boldsymbol{\theta}^T \mathbf{x}_t)^2.$$

Least Squares

- By the previous slide, maximizing likelihood is equivalent to least squares:

$$\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \mathcal{D}) \iff \min_{\boldsymbol{\theta}} \sum_{t=1}^{n} (y_t - \boldsymbol{\theta}^T \mathbf{x}_t)^2.$$

- By some basic matrix algebra, this has a closed form solution:

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

where $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$ and $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} \in \mathbb{R}^{n \times d}$

  - Proof: Lecture 4 of https://www.comp.nus.edu.sg/~scarlett/CS5339_notes/

- Problem: Can be highly sensitive to noise if $\mathbf{X}^T \mathbf{X}$ is not "well-conditioned" (this can be formalized by the bias-variance trade-off)

Regularized Least Squares

- A more stable solution can be obtained by regularization: **(Ridge Regression)**

$$\min_{\boldsymbol{\theta}} \sum_{t=1}^{n} (y_t - \boldsymbol{\theta}^T \mathbf{x}_t)^2 + \lambda \sum_{i=1}^{d} \theta_i^2$$

The higher the parameter λ, the more small entries of $\boldsymbol{\theta}$ are favored.

Regularized Least Squares

- A more stable solution can be obtained by regularization: **(Ridge Regression)**

$$\min_{\boldsymbol{\theta}} \sum_{t=1}^{n} (y_t - \boldsymbol{\theta}^T \mathbf{x}_t)^2 + \lambda \sum_{i=1}^{d} \theta_i^2$$

The higher the parameter λ, the more small entries of $\boldsymbol{\theta}$ are favored.

- Similar closed-form expression to the non-regularized case:

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$

  - Even if $\mathbf{X}^T \mathbf{X}$ poorly-conditioned, adding $\lambda \mathbf{I}$ makes it well-conditioned

Regularized Least Squares

- A more stable solution can be obtained by regularization: **(Ridge Regression)**

$$\min_{\boldsymbol{\theta}} \sum_{t=1}^{n} (y_t - \boldsymbol{\theta}^T \mathbf{x}_t)^2 + \lambda \sum_{i=1}^{d} \theta_i^2$$

The higher the parameter λ, the more small entries of $\boldsymbol{\theta}$ are favored.

- Similar closed-form expression to the non-regularized case:

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$

  - Even if $\mathbf{X}^T \mathbf{X}$ poorly-conditioned, adding $\lambda \mathbf{I}$ makes it well-conditioned

- More regularization (higher λ) tends to:
  - increase bias ($\mathbb{E}[\hat{\boldsymbol{\theta}}]$ is further from the true $\boldsymbol{\theta}$)
  - decrease variance ($\hat{\boldsymbol{\theta}}$ varies less with respect to the noise z_t)

By a suitable balance of these, a good choice of λ can reduce $\mathbb{E}[(\boldsymbol{\theta}_{\text{true}} - \hat{\boldsymbol{\theta}})^2]$

Note:

- Regularization term $\lambda||\boldsymbol{\theta}||^2$ reduces sensitivity to noise, and can help prevent overfitting (example to come shortly)

- Higher bias, lower variance

Non-Linear Regression

- In some cases, the application under consideration might naturally lend itself to a specific non-linear model

- **Examples of non-linear regression:**
  - Logistic regression (e.g., classification)
  - Poisson regression (e.g., low-light imaging, queuing)
  - Generalized linear models
  - ...

Non-Linear Regression

- In some cases, the application under consideration might naturally lend itself to a specific non-linear model

- **Examples of non-linear regression:**
  - Logistic regression (e.g., classification)
  - Poisson regression (e.g., low-light imaging, queuing)
  - Generalized linear models
  - ...

- Problem: *Choosing a "good" non-linear model can be extremely difficult*

(Non)-Linear Regression with Features

- **Introducing features:**
  - Choose an appropriate feature space $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_{d'}(\mathbf{x}))$
  - Assume (approximately) linear relation between $\boldsymbol{\phi}(\mathbf{x})$ and y:

$$y = \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}) + z \tag{2}$$

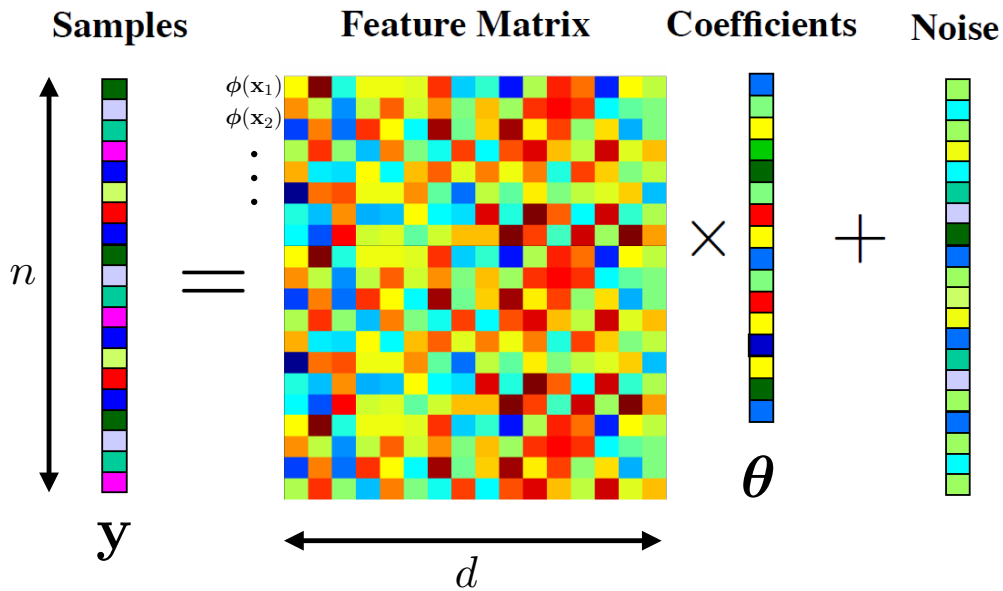
where z is possible noise

(Non)-Linear Regression with Features

- **Introducing features:**
  - Choose an appropriate feature space $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_{d'}(\mathbf{x}))$
  - Assume (approximately) linear relation between $\phi(\mathbf{x})$ and y:

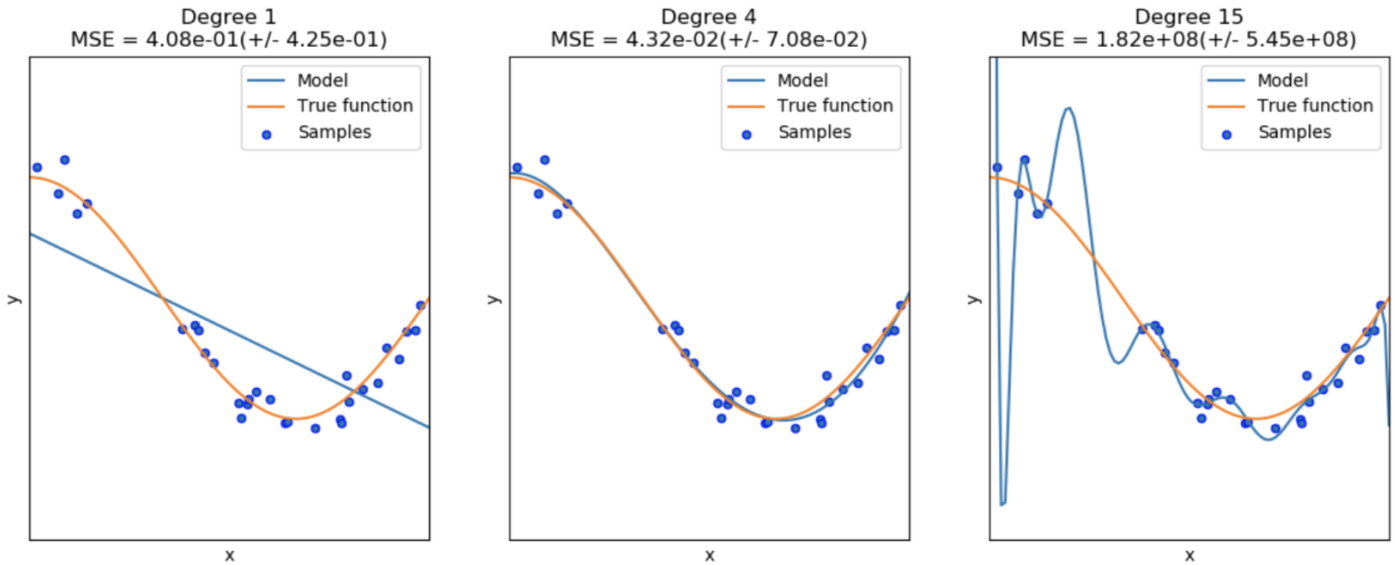
$$y = \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}) + z \tag{2}$$

    where z is possible noise
  - Example: Polynomial regression for $x \in \mathbb{R}$: $\boldsymbol{\phi}(x) = (1, x, x^2, \ldots, x^p)$
  - Problem: *Designing "good" features can be very difficult*



Toy Example (I)

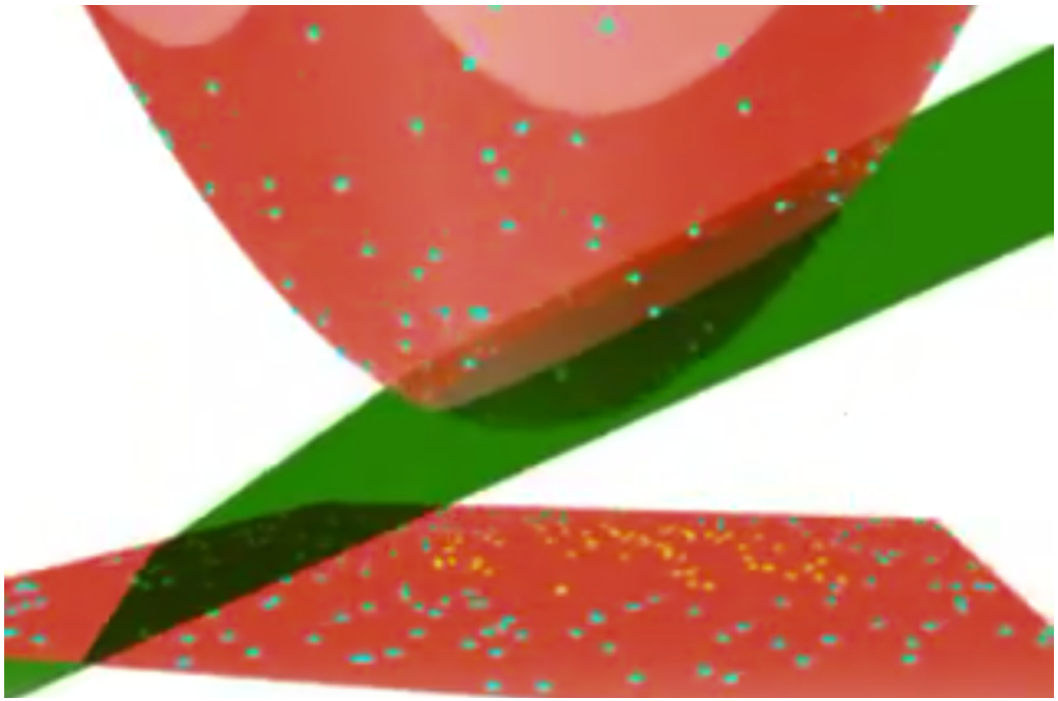
- Fitting polynomials of increasing degree [scikit-learn.com]:



  - **Note:** To avoid the right-hand scenario, we can limit how large the polynomial degree p is, or large p may be OK if we choose the right amount of regularization

Toy Example (II)

- Example from [http://www.youtube.com/watch?v=3liCbRZPrZA]:



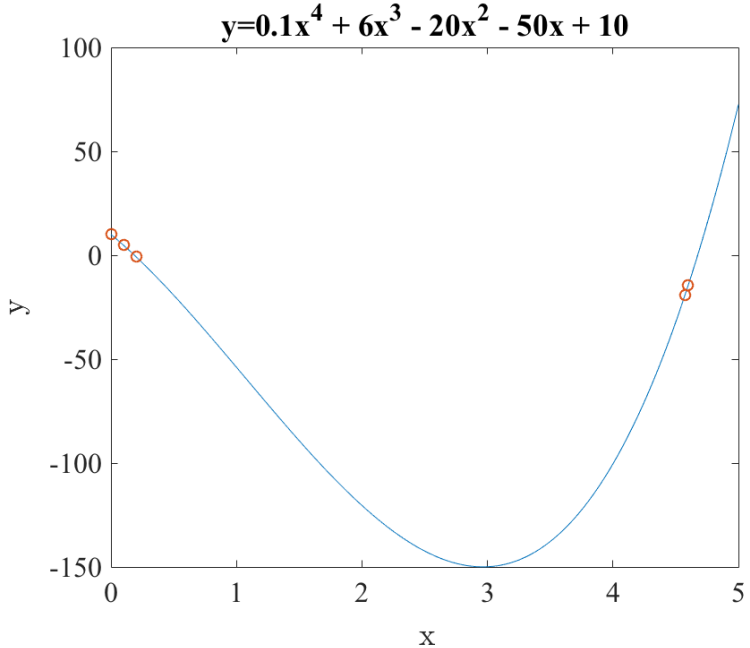
  - Bottom: Labeled inputs in 2D space (not linearly separable)
  - Top: Inputs mapped to 3D space (linearly separable)

Exercise: Why does regularization help?

- 4-th degree polynomial regression: $\phi(x) = [1, x, x^2, x^3, x^4]^T$ where x is scalar.

- Then $\langle \boldsymbol{\theta}, \phi(x) \rangle = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$, so the target y is being modeled (possibly incorrectly) as a 4-th degree polynomial

- **Un-regularized least squares from 5 data samples:**



Questions:

1. Why does the least-squares $\boldsymbol{\theta}$ yield exactly $y_t = \langle \boldsymbol{\theta}, \phi(x_t) \rangle$ for each data point $t = 1, \ldots, 5$?
2. What happens to the blue curve if the second-right most y_t is shifted up or down?
3. How will this behavior change with regularization? ($\lambda \sum_i \theta_i^2$ term)

Introducing Kernels

- Many machine learning algorithms depend on the data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ only through the pairwise inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^T \mathbf{x}_j$
  - Ridge regression (to be shown shortly)
  - Support vector machine (in "dual" form)
  - Nearest-neighbor methods
  - Any algorithm only depending on distances and angles between points

Introducing Kernels

- Many machine learning algorithms depend on the data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ only through the pairwise inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^T \mathbf{x}_j$
  - Ridge regression (to be shown shortly)
  - Support vector machine (in "dual" form)
  - Nearest-neighbor methods
  - Any algorithm only depending on distances and angles between points

- We know that moving to feature spaces can help, so we could map each $\mathbf{x}_i \to \boldsymbol{\phi}(\mathbf{x}_i)$ and apply the algorithm using $\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$

Introducing Kernels

• Many machine learning algorithms depend on the data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ only through the pairwise inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^T \mathbf{x}_j$

- Ridge regression (to be shown shortly)
- Support vector machine (in "dual" form)
- Nearest-neighbor methods
- Any algorithm only depending on distances and angles between points

• We know that moving to feature spaces can help, so we could map each $\mathbf{x}_i \to \phi(\mathbf{x}_i)$ and apply the algorithm using $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$

• A kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ can be thought of as an inner product in a *possibly implicit* feature space

- **Key idea.** There are clever choices of $\phi(\cdot)$ ensuring that we can efficiently compute $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ *without ever explicitly mapping to the feature space*
- The implicit feature space may be infinite-dimensional, so we could not explicitly map to it even if we wanted to.

Introducing Kernels

• Many machine learning algorithms depend on the data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ only through the pairwise inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^T \mathbf{x}_j$

- Ridge regression (to be shown shortly)
- Support vector machine (in "dual" form)
- Nearest-neighbor methods
- Any algorithm only depending on distances and angles between points

• We know that moving to feature spaces can help, so we could map each $\mathbf{x}_i \to \phi(\mathbf{x}_i)$ and apply the algorithm using $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$

• A kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ can be thought of as an inner product in a *possibly implicit* feature space

- **Key idea.** There are clever choices of $\phi(\cdot)$ ensuring that we can efficiently compute $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ *without ever explicitly mapping to the feature space*
- The implicit feature space may be infinite-dimensional, so we could not explicitly map to it even if we wanted to.

• **Intuition.** The kernel function is a measure of similarity

Examples of Kernels

- We will see more examples of kernels later, and only state some simple ones here
- **Linear kernel:** $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$
- **Polynomial kernel:** $k(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^p$
- **Radial basis function (RBF) kernel:** $k(\mathbf{x}, \mathbf{x}') = e^{-\|\mathbf{x}-\mathbf{x}'\|^2/(2\ell)}$
- **Kernels on abstract data types:**
  - Simple example on sets: $k(S, S') = |S \cap S'|$
  - Other data types: Strings, documents, graphs, molecules, etc.

Notes:

- Learning algorithms depending only on $\langle \mathbf{x}, \mathbf{x}' \rangle$ can be kernelized

- Kernels allow us to implicitly work in "large" feature spaces

- Intuitively, $k(\mathbf{x}, \mathbf{x}')$ is a measure of similarity between $\mathbf{x}$ and $\mathbf{x}'$

Exercise

• **Question 1.** Is the trivial choice $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$ consistent with the idea that the kernel measures similarity?

• **Question 2.** Why can any algorithm depending on $\mathbf{x}_1, \ldots, \mathbf{x}_n$ only through pairwise distances $\|\mathbf{x}_i - \mathbf{x}_j\|$ and angles $\mathrm{angle}(\mathbf{x}_i, \mathbf{x}_j)$ be kernelized?

• **Question 3.** How do we measure similarity between text data? e.g.,

- $\mathbf{x}_1 =$ "This sentence is the first string in my data set"
- $\mathbf{x}_2 =$ "The second string in my data set is this sentence"
- $\mathbf{x}_3 =$ "Tihs sentance is the third stirng in my dataset"

Kernel Ridge Regression (I)

- We saw the ridge regression estimator:

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}.$$

- Equivalent form:

$$\hat{\boldsymbol{\theta}} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y}.$$

  - This is easy to prove but non-trivial to see immediately
  - See Lecture 5 of CS5339 notes for the details

- Substituting into $\hat{y}(\mathbf{x}') = \hat{\boldsymbol{\theta}}^T\mathbf{x}' = (\mathbf{x}')^T\hat{\boldsymbol{\theta}}$ gives

$$\hat{y}(\mathbf{x}') = (\mathbf{x}')^T\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y}.$$

Kernel Ridge Regression (II)

• **Crucial observation.** The prediction depends on the data only through inner products, since

$$(\mathbf{x}')^T\mathbf{X}^T = \begin{bmatrix} \langle \mathbf{x}', \mathbf{x}_1 \rangle \\ \vdots \\ \langle \mathbf{x}', \mathbf{x}_n \rangle \end{bmatrix}, \quad \mathbf{X}\mathbf{X}^T = \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{x}_1 \rangle & \dots & \langle \mathbf{x}_1, \mathbf{x}_n \rangle \\ \vdots & \ddots & \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_1 \rangle & \dots & \langle \mathbf{x}_n, \mathbf{x}_n \rangle \end{bmatrix}.$$

Kernel Ridge Regression (II)

• **Crucial observation.** The prediction depends on the data only through inner products, since

$$(\mathbf{x}')^T\mathbf{X}^T = \begin{bmatrix} \langle \mathbf{x}', \mathbf{x}_1 \rangle \\ \vdots \\ \langle \mathbf{x}', \mathbf{x}_n \rangle \end{bmatrix}, \quad \mathbf{X}\mathbf{X}^T = \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{x}_1 \rangle & \dots & \langle \mathbf{x}_1, \mathbf{x}_n \rangle \\ \vdots & \ddots & \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_1 \rangle & \dots & \langle \mathbf{x}_n, \mathbf{x}_n \rangle \end{bmatrix}.$$

• **Kernel trick.** Replacing inner products by kernel evaluations gives

$$\hat{y}(\mathbf{x}') = \mathbf{k}(\mathbf{x}')(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y},$$

where

$$\mathbf{k}(\mathbf{x}') = \begin{bmatrix} k(\mathbf{x}', \mathbf{x}_1) \\ \vdots \\ k(\mathbf{x}', \mathbf{x}_n) \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}.$$

This is known as kernel ridge regression.

Kernel Ridge Regression

- **Interpretation.** The prediction rule

$$\hat{y}(\mathbf{x}') = \mathbf{k}(\mathbf{x}')(\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{y},$$

can *roughly* be interpreted as follows:

For a new point $\mathbf{x}'$, the estimate $\hat{y}$ is a weighted sum of the $\{y_t\}_{t=1}^n$ in the data set, with higher weights given when $\mathbf{x}_t$ is more similar to $\mathbf{x}'$.

- "Similarity" here is measured by the kernel $k(\mathbf{x}_t, \mathbf{x}')$
- e.g., if $k(\mathbf{x}, \mathbf{x}') = e^{-\|\mathbf{x}-\mathbf{x}'\|^2}$ then nearby points are weighted more

Non-Parametric Regression

• Kernel methods corresponding to infinite-dimensional $\phi(\mathbf{x})$ can usually be considered as non-parametric

• Idea of non-parametric methods:

- Avoid introducing an explicit input-output relationship and its associated parameters (e.g., θ)
- Instead, construct a model via some (explicit or implicit) form of interpolation of the available samples

Non-Parametric Regression

• Kernel methods corresponding to infinite-dimensional $\phi(\mathbf{x})$ can usually be considered as non-parametric

• Idea of non-parametric methods:
  - Avoid introducing an explicit input-output relationship and its associated parameters (e.g., $\boldsymbol{\theta}$)
  - Instead, construct a model via some (explicit or implicit) form of interpolation of the available samples

• **Simple examples:**
  - For the 1D case $x \in \mathbb{R}$, can interpolate by "joining the dots"
  - A non-parametric approach that applies more generally (nearest-neighbors):
    - Given a new $\mathbf{x}$, find the sample $\mathbf{x}_1, \ldots, \mathbf{x}_n$ closest to it
    - Predict y to be the same as the corresponding y_i
    - *This can work surprisingly well in some cases*
  - Generalizations: e.g., k-nearest neighbors

Bayesian Modeling and Terminology

- Consider a class of models parametrized by $\boldsymbol{\theta} \in \mathbb{R}^d$ (e.g., $y = \boldsymbol{\theta}^T \mathbf{x} + z$)
- Distinct viewpoints:
  - **Frequentist view.** The parameter $\boldsymbol{\theta}$ is just some fixed vector that we don't know
  - **Bayesian view.** We can encode our belief of the possible/likely values of $\boldsymbol{\theta}$ through a distribution $p(\boldsymbol{\theta})$ (e.g., $\boldsymbol{\theta} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$)

Bayesian Modeling and Terminology

- Consider a class of models parametrized by $\boldsymbol{\theta} \in \mathbb{R}^d$ (e.g., $y = \boldsymbol{\theta}^T \mathbf{x} + z$)
- Distinct viewpoints:
  - **Frequentist view.** The parameter $\boldsymbol{\theta}$ is just some fixed vector that we don't know
  - **Bayesian view.** We can encode our belief of the possible/likely values of $\boldsymbol{\theta}$ through a distribution $p(\boldsymbol{\theta})$ (e.g., $\boldsymbol{\theta} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$)
- **Bayes' rule:**

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}$$

which reads in Bayesian terminology as

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$$

Advantages and Disadvantages of Bayesian Approach

Advantages.

- Natural way to incorporate prior knowledge
- Gives not only a prediction, but a full posterior distribution (e.g., to provide estimates of the level of (un)certainty)
- State-of-the-art performance in many applications

Advantages and Disadvantages of Bayesian Approach

Advantages.

- Natural way to incorporate prior knowledge
- Gives not only a prediction, but a full posterior distribution (e.g., to provide estimates of the level of (un)certainty)
- State-of-the-art performance in many applications

Disadvantages.

- Choosing a prior can be difficult
- With an incorrect prior, can have very undesirable behavior (e.g., claiming high confidence but actually being completely wrong)
- Exact posterior calculation usually impossible, need to approximate (e.g., with Monte Carlo or variational methods)
- Even with approximations, considerable computation time is often required

Notes:

- Bayesian models can provide much more than just a "point estimate" of $\boldsymbol{\theta}$
- (but must be interpreted with care)

Bayesian Perspective on Linear Regression & Ridge Regression

- **Useful observation.** Gaussian prior & Gaussian noise $\implies$ Gaussian posterior

- A simple setup:
  - Linear model $y = \boldsymbol{\theta}^T \mathbf{x} + z$ with random $\boldsymbol{\theta}$
  - Gaussian prior $\boldsymbol{\theta} \sim N(\mathbf{0}, \mathbf{I})$
  - Gaussian noise $z \sim N(0, \sigma^2)$ with independence between samples

Bayesian Perspective on Linear Regression & Ridge Regression

- **Useful observation.** Gaussian prior & Gaussian noise $\implies$ Gaussian posterior

- A simple setup:
  - Linear model $y = \boldsymbol{\theta}^T \mathbf{x} + z$ with random $\boldsymbol{\theta}$
  - Gaussian prior $\boldsymbol{\theta} \sim N(\mathbf{0}, \mathbf{I})$
  - Gaussian noise $z \sim N(0, \sigma^2)$ with independence between samples

- Since the posterior of $\boldsymbol{\theta}$ is Gaussian, it is fully specified by its mean and covariance matrix. The mean is given as follows: (proof outline on next slide)

$$\boldsymbol{\mu}_n = (\mathbf{X}^T \mathbf{X} + \sigma^2 \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$

where $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$ and $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} \in \mathbb{R}^{n \times d}$

  - Matches regularized least squares (ridge regression) with $\lambda \leftrightarrow \sigma^2$
  - The posterior covariance matrix also admits a closed-form expression

Bayesian Posterior Derivation

- Both $v(\mathbf{x}) = \langle \boldsymbol{\theta}, \mathbf{x} \rangle$ and $v(\mathbf{x}') = \langle \boldsymbol{\theta}, \mathbf{x}' \rangle$ have mean zero, so their covariance is

$$\begin{aligned} \mathrm{Cov}[v(\mathbf{x}), v(\mathbf{x}')] &= \mathbb{E}[(\mathbf{x}^T\boldsymbol{\theta})((\mathbf{x}')^T\boldsymbol{\theta})] = \mathbb{E}[(\mathbf{x}^T\boldsymbol{\theta})(\boldsymbol{\theta}^T\mathbf{x}')] \\ &= \mathbf{x}^T\mathbb{E}[\boldsymbol{\theta}\boldsymbol{\theta}^T]\mathbf{x}' = \mathbf{x}^T\mathbf{x}' = \langle \mathbf{x}, \mathbf{x}' \rangle, \end{aligned}$$

since $\mathbb{E}[\boldsymbol{\theta}\boldsymbol{\theta}^T] = \mathbf{I}$ for $\boldsymbol{\theta} \sim N(\mathbf{0}, \mathbf{I})$.

- Using this property and the fact that $y_t \sim N(v(\mathbf{x}_t), \sigma^2)$, we can deduce that

$$\begin{bmatrix} v' \\ \mathbf{y} \end{bmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} \|\mathbf{x}'\|^2 & (\mathbf{x}')^T\mathbf{X}^T \\ \mathbf{X}\mathbf{x}' & \mathbf{X}\mathbf{X}^T + \sigma^2\mathbf{I} \end{bmatrix}\right),$$

where v' is short for $v(\mathbf{x}')$.

- Applying the conditional Gaussian formula gives that $(v'|\mathbf{y})$ has mean

$$(\mathbf{x}')^T\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \sigma^2\mathbf{I})^{-1}\mathbf{y},$$

and variance $\|\mathbf{x}'\|^2 - (\mathbf{x}')^T\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \sigma^2\mathbf{I})^{-1}\mathbf{X}\mathbf{x}'$ (not needed here).

- But with $\boldsymbol{\mu}_n$ denoting the posterior mean of $\boldsymbol{\theta}$, we also have $\mathbb{E}[v(\mathbf{x}')] = (\mathbf{x}')^T\boldsymbol{\mu}_n$. Equating with the above equation gives $\boldsymbol{\mu}_n = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \sigma^2\mathbf{I})^{-1}\mathbf{y}$, which we have already seen is equivalent to the formula on the previous slide.

References

[1] Christopher M. Bishop.
Pattern Recognition and Machine Learning (Information Science and Statistics).
Springer-Verlag, 2006.