COMP90051
Statistical Machine Learning

Workshop Week 9

Xudong Han

https://github.com/HanXudong/COMP90051_2020_S1

Bayesian Regression

- Frequentist V.S. Bayesian
- Bayesian regression with known variance
- Bayesian model selection
- Bayesian regression with unknown variance

Frequentist V.S. Bayesian

- Frequentist (MLE)
  Generally reduces to minimizing the negative log-likelihood. Returns a point-estimate.

$$\theta_{MLE} = \text{argmax}_\theta \; p(X|\theta) = \text{argmax}_\theta \prod_i^n p(x_i|\theta) = \text{argmax}_\theta \sum_i^n \log p(x_i|\theta)$$

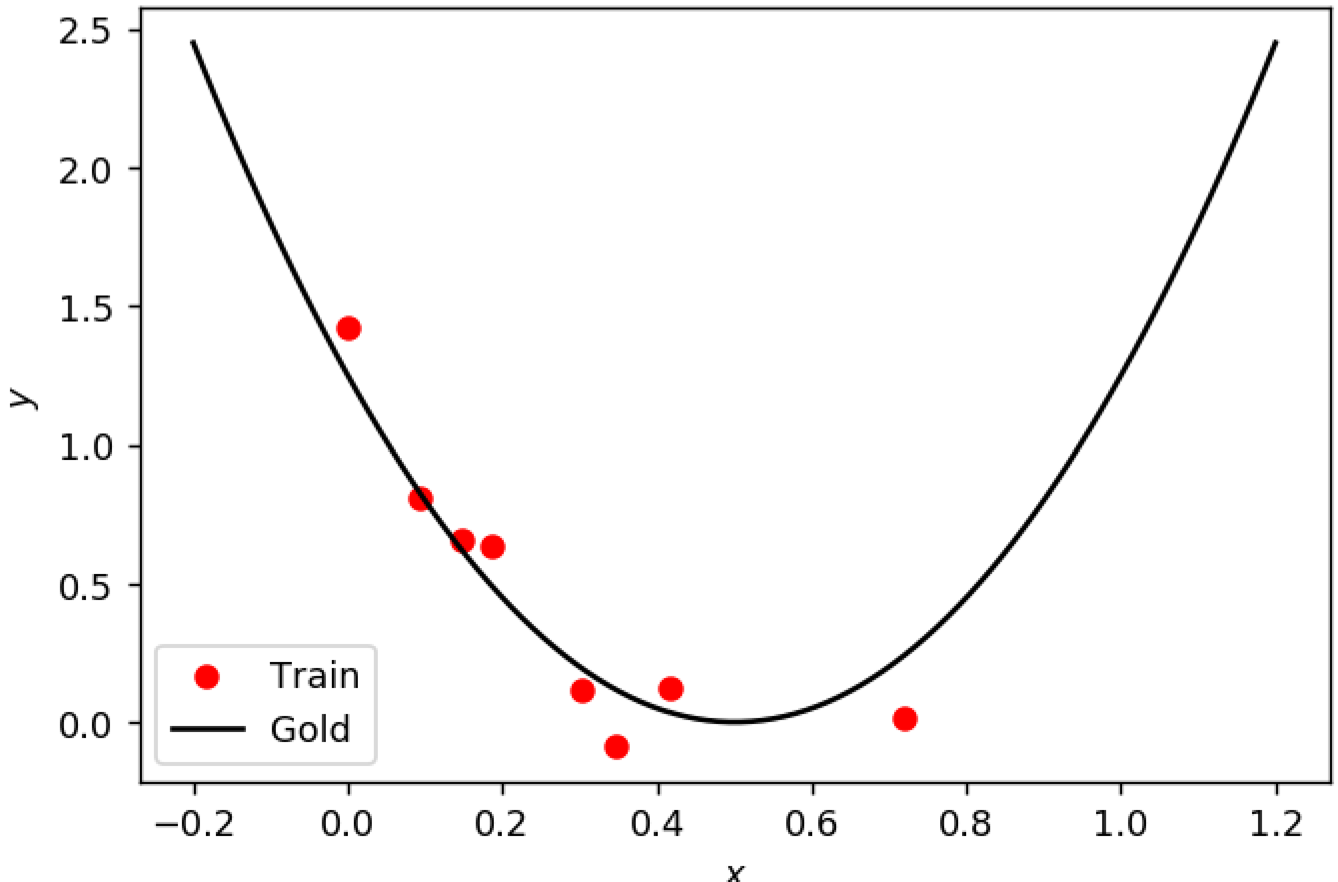
- Bayesian:

$$p(X|\theta) = \frac{\prod_i^n p(\theta|x_i)p(\theta)}{\int d\theta \prod_i^n p(\theta|x_i)p(\theta)}$$

1. Regression data set

$$x \sim \text{Uniform}[0, 1]$$

$$y|x, \sigma^2 \sim \text{Normal}\left[5\left(x - \frac{1}{2}\right)^2, \sigma^2\right]$$



Polynomial basis functions

Since the relationship between y and x is non-linear, we'll apply polynomial basis expansion to degree d.

$$\mathbf{\Phi} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ 1 & x_2 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^d \end{bmatrix}$$

2. Bayesian regression with known variance

- Prior

$$w|\gamma \sim \text{Normal}(\vec{\mathbf{0}}, \gamma^2 \mathrm{I}_m)$$

- Likelihood

$$\mathrm{p}(y|X, w, \sigma) = \prod_{i=1}^{n} p\left(y_i \middle| \overrightarrow{X_i}, \vec{w}, \sigma\right)$$

Since $\mathrm{y_i}|\ \overrightarrow{X_i}, \vec{w}, \sigma \sim \text{Normal}(\overrightarrow{X_i}^T \vec{w},\ \sigma^2)$,

$$\vec{y}|X, \vec{w}\ , \sigma \sim \text{Multivariate Normal}(\mathrm{X}\vec{w}, \sigma^2 I_n)$$

Bayesian regression with known variance

Given this formulation, the next step is to solve for the posterior over $\mathbf{w}$

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma, \gamma) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma)p(\mathbf{w}|\gamma)}{p(\mathbf{y}|\mathbf{X}, \sigma)}$$

where $\mathbf{X} \in \mathbb{R}^{n \times m}$ is the feature matrix and $\mathbf{y} \in \mathbb{R}^n$ is the vector of target values for each instance.

In lectures, we derived the following solution:

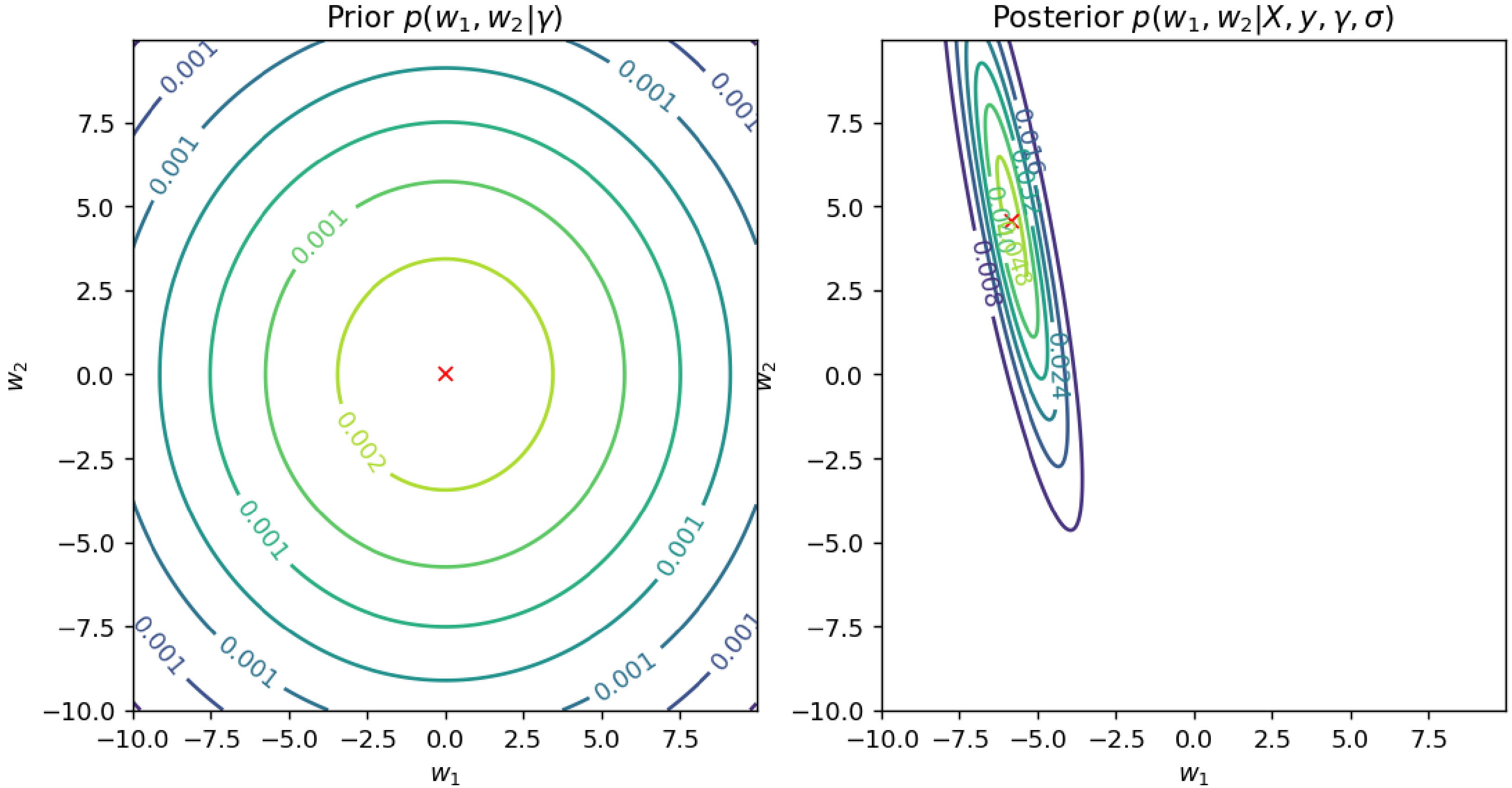
$$\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma, \gamma \sim \mathrm{Normal}(\mathbf{w}_N, \mathbf{V}_N)$$

where $\mathbf{V}_N = \sigma^2 \left(\mathbf{X}^\intercal \mathbf{X} + \frac{\sigma^2}{\gamma^2} \mathbf{I}_m \right)^{-1}$ and $\mathbf{w}_N = \frac{1}{\sigma^2} \mathbf{V}_N \mathbf{X}^\intercal \mathbf{y}$.

numpy.linalg.inv() Compute the (multiplicative) inverse of a matrix.
https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.inv.html#numpy.linalg.inv

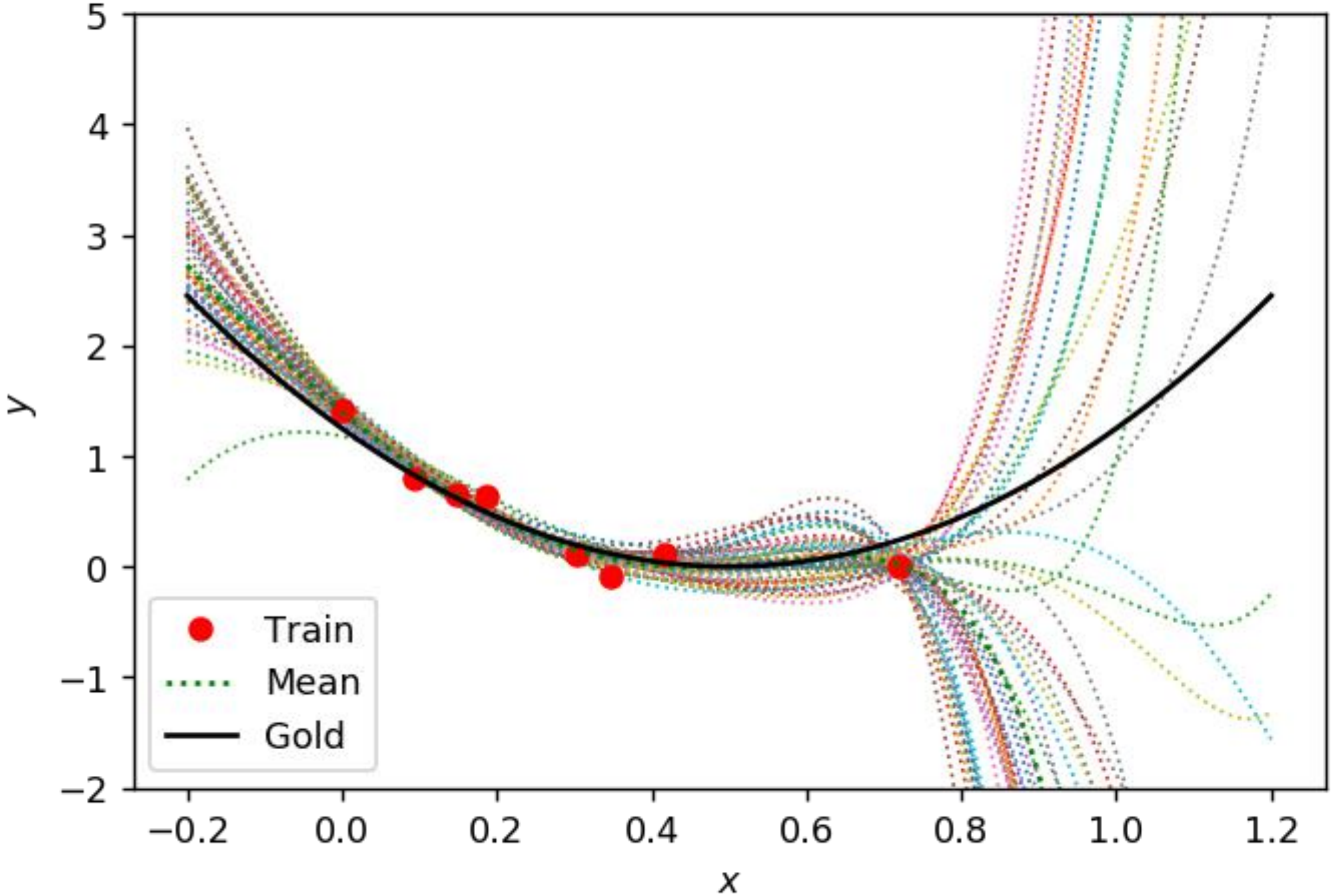
np.Identity / np.eye Return a 2-D array with ones on the diagonal and zeros elsewhere.
https://github.com/numpy/numpy/blob/v1.9.1/numpy/core/numeric.py#L2125

plot the prior and posterior over w_1, w_2



Discussion question: Can you explain why the prior and the posterior are so different? How is this related to the dataset? Why are the ellipses in the posterior not aligned to the axes? *You might want to change the parameter indices from 0,1 to other pairs to get a better idea of the full posterior.*

Bayesian inference



The Bayesian Predictive Distribution

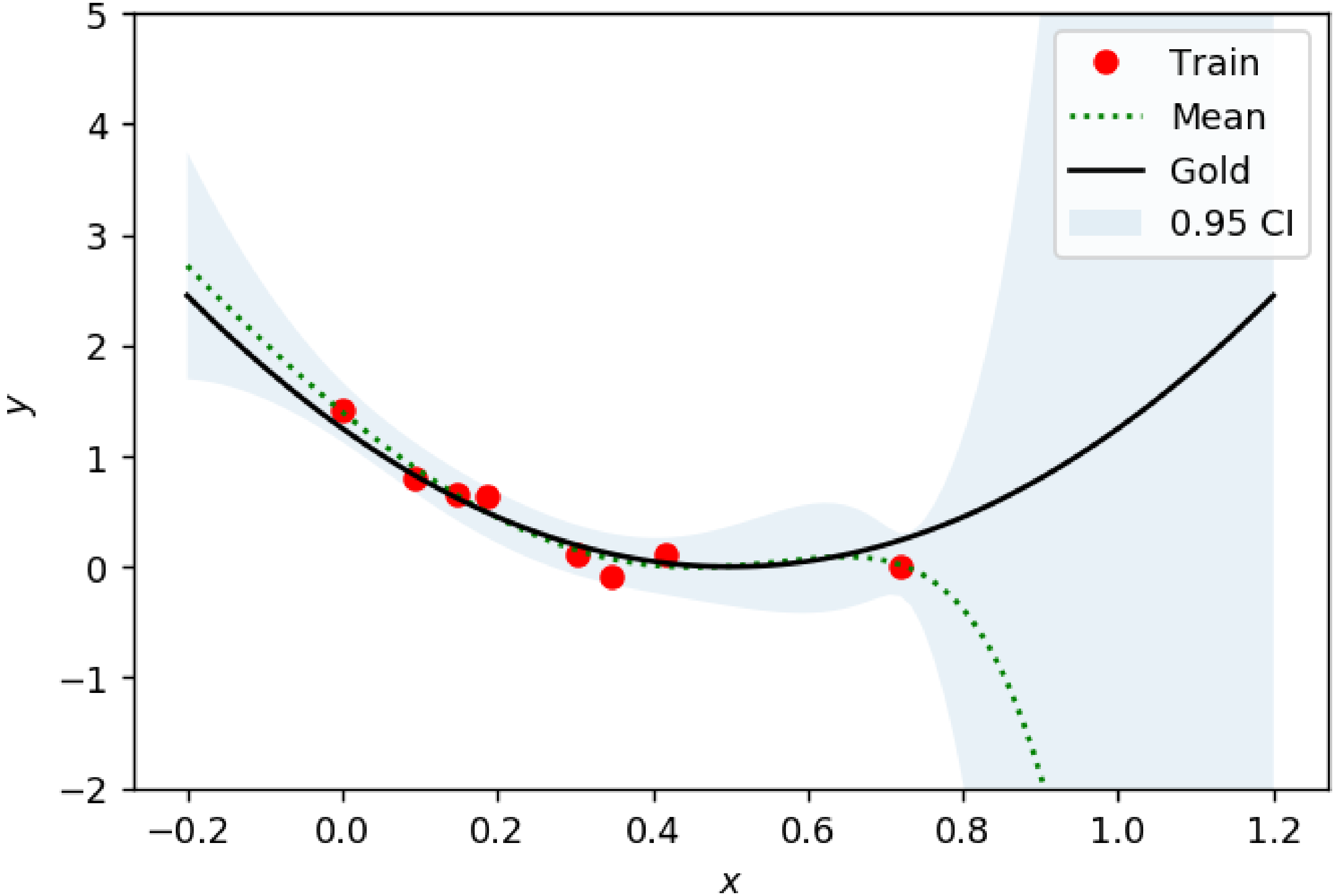
Thanks to conjugacy, the predictive distribution can be found in closed form in our toy problem.

$$y_* | \mathbf{x}_*, \mathbf{w}_N, \mathbf{V}_N, \sigma = \text{Normal}\left[\langle \mathbf{x}^*, \mathbf{w}_N \rangle, \sigma_N^2(\mathbf{x}^*)\right]$$

$$\sigma_N^2(\mathbf{x}^*) = \sigma^2 + (\mathbf{x}^*)^T \mathbf{V}_N \mathbf{x}^*$$

```
def target_std(X, V_N, sigma):
    """
    Compute the predictive standard deviation for the target variable, given X, V_N and sigma

    Arguments
    =========
    X : numpy array, shape: (n_instances, n_features)
        feature matrix
    V_N : numpy array, shape: (n_features, n_features)
        covariance parameter

    Returns
    =======
    std : numpy array, shape: (n_instances,)
        predictive standard deviation for each instance in X
    """
    # your code here #

    variance = ...
    std = np.sqrt(variance)

    return std
```

Bayesian inference



Bayesian model selection