Advanced Probabilistic Machine Learning
SSY316

Bayesian Linear Regression

Alexandre Graell i Amat
alexandre.graell@chalmers.se
https://sites.google.com/site/agraellamat

October 31 and November 7, 2023

CHALMERS

Supervised machine learning

Goal: Given a training set $\mathcal{D} = \{(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)\}$, identify an algorithm to predict the label t for a new (yet unseen) point $\boldsymbol{x}$

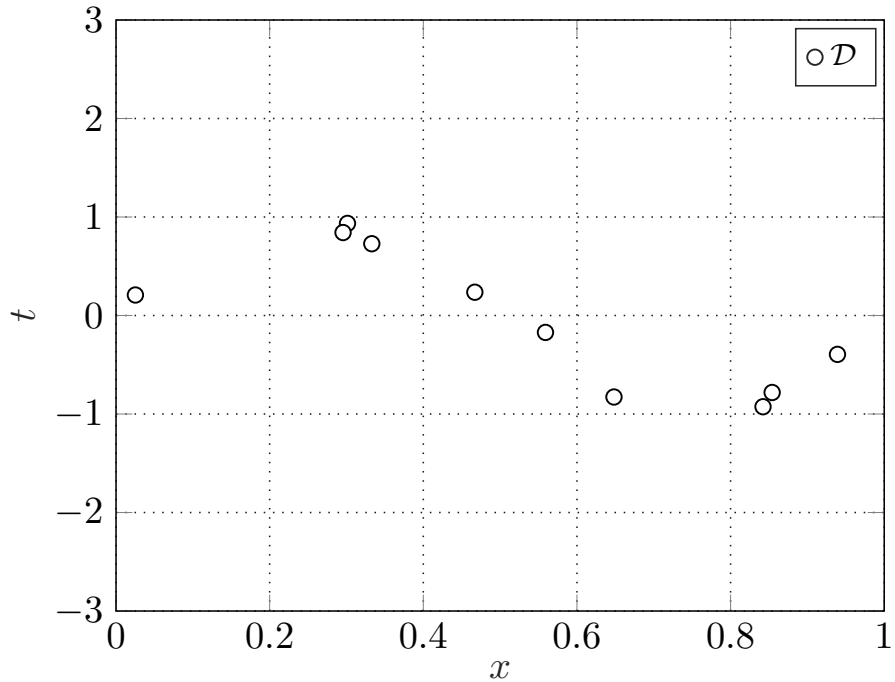
- Learning a model from labeled data
- Predicting output of new data based on the learned model

- $\boldsymbol{x}_i$: free variables (features, covariates, domain points, explanatory variables)
- t_i: target variables, dependent on $\boldsymbol{x}_i$ (dependent variables, labels, responses)

Supervised machine learning

Goal: Given a training set $\mathcal{D} = \{(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)\}$, identify an algorithm to predict the label t for a new (yet unseen) point $\boldsymbol{x}$.

Impossible if no information on the mechanism relating $\boldsymbol{x}$ and t!



We may assume that $\boldsymbol{x}$ and t are related via a function $t = \tilde{f}(\boldsymbol{x})$

Goal: Find best possible approximation of $\tilde{f}(\boldsymbol{x})$, $f(\boldsymbol{x})$ (prediction model). Predict the outcome t for $\boldsymbol{x}$ as $\hat{t} = f(\boldsymbol{x})$.

Supervised machine learning

Goal: Identify a predictive algorithm that minimizes the error in the prediction of a new label t for an unobserved $\boldsymbol{x}$ (generalization loss).

Two approaches:

- Frequentist approach
- Bayesian approach

Rephrase supervised machine learning as a Bayesian inference problem.

Inference

Learning: Treat **x** and t as random variables

Predict t given the observation of **x** under assumption joint distribution $p(\boldsymbol{x}, t)$ is known.

- Loss function $\ell(t, \hat{t})$: cost (loss or risk) incurred when the correct value is t while the estimate is $\hat{t}$
- Quadratic loss function:

$$\ell(t, \hat{t}) = (t - \hat{t})^2$$

- Optimal prediction $\hat{t}^*(\boldsymbol{x})$: $\hat{t}(\boldsymbol{x})$ that minimizes generalization loss (generalization error)

$$L_p(\hat{t}) = \mathbb{E}_{\mathbf{xt} \sim p_{\mathbf{x},\mathbf{t}}}[\ell(\mathbf{t}, \hat{t}(\mathbf{x}))]$$

Inference

- Solution:

$$\begin{aligned}\hat{t}^*(\boldsymbol{x}) &= \arg\min_{\hat{t}} L_p(\hat{t}) \\ &= \arg\min_{\hat{t}} \mathbb{E}_{\mathbf{x}\mathsf{t}\sim p_{\mathbf{x},\mathsf{t}}}[\ell(\mathsf{t},\hat{t}(\mathbf{x}))] \\ &= \arg\min_{\hat{t}} \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}}\left[\mathbb{E}_{\mathsf{t}\sim p_{\mathsf{t}|\mathbf{x}}}[\ell(\mathsf{t},\hat{t}(\mathbf{x}))]\right] \\ &= \arg\min_{\hat{t}} \mathbb{E}_{\mathsf{t}\sim p_{\mathsf{t}|\mathbf{x}}}[\ell(\mathsf{t},\hat{t}(\mathbf{x}))]\end{aligned}$$

Optimal prediction a function of $p(t|\boldsymbol{x})$ and the loss function.

- For the loss function $\ell(t,\hat{t}) = (t-\hat{t})^2$,

$$\hat{t}^*(\boldsymbol{x}) = \mathbb{E}_{\mathsf{t}|\mathbf{x}}[\mathsf{t}|\boldsymbol{x}]$$

Supervised machine learning

Goal (revisited): Obtaining predictor $\hat{t}(\boldsymbol{x})$ that performs close to optimal predictor $\hat{t}^*(\boldsymbol{x})$ based only on training set $\mathcal{D}$ (without knowledge of $p(\boldsymbol{x}, t)$).

- Closeness measured as

$$L_p(\hat{t}) - L_p(\hat{t}^*)$$

Supervised machine learning: Frequentist approach

- Training data points $(\mathbf{x}_i, \mathsf{t}_i) \in \mathcal{D}$ are i.i.d RVs drawn from a true (unknown) distribution $p(\boldsymbol{x}, t)$,

$$(\mathbf{x}_i, \mathsf{t}_i) \sim_{\text{i.i.d}} p(\boldsymbol{x}, t) \quad i = 1, \ldots, N$$

- Since $p(\boldsymbol{x}, t)$ unknown, cannot find optimal prediction via

$$\hat{t}^*(\boldsymbol{x}) = \arg\min_{\hat{t}} \mathbb{E}_{\mathsf{t}|\mathbf{x}}[\ell(\mathsf{t}, \hat{t})|\boldsymbol{x}]$$

Supervised machine learning: Frequentist approach

Solutions:

1. Separate learning and inference: Learn an approximation of $p(t|\boldsymbol{x})$ based on $\mathcal{D}$ ($p_{\mathcal{D}}(t|\boldsymbol{x})$) and use it in

$$\hat{t}^*(\boldsymbol{x}) = \arg\min_{\hat{t}} \mathbb{E}_{\mathsf{t}|\mathsf{x}}[\ell(\mathsf{t}, \hat{t})|\boldsymbol{x}]$$

   to obtain

$$\hat{t}_{\mathcal{D}}(\boldsymbol{x}) = \arg\min_{\hat{t}} \mathbb{E}_{\mathsf{t}\sim p_{\mathcal{D}}(\mathsf{t}|\mathsf{x})}[\ell(\mathsf{t}, \hat{t})|\boldsymbol{x}]$$

2. Direct inference via empirical risk minimization (ERM): Learn directly an approximation of optimal decision rule ($\hat{t}_{\mathcal{D}}(\cdot)$) by minimizing an empirical estimate of generalization loss,

$$\hat{t}^*_{\mathcal{D}}(\boldsymbol{x}) = \arg\min_{\hat{t}} \; L_{\mathcal{D}}(\hat{t}) = \arg\min_{\hat{t}} \; \frac{1}{N}\sum_{i=1}^{N} \ell(t_i, \hat{t}(\boldsymbol{x}_i))$$

Separate learning and inference

How do we learn an approximation $p_{\mathcal{D}}(t|\boldsymbol{x})$ of $p(t|\boldsymbol{x})$ based on $\mathcal{D}$?

Idea:

- Select a family of parametric probabilistic models (hypothesis class)
- Learn model parameters to fit $\mathcal{D}$

Linear regression

- We are interested in values of a function $t(\boldsymbol{x}) : \mathbb{R}^d \longrightarrow \mathbb{R}$, $\boldsymbol{x} = (x_1, \ldots, x_d)^\mathsf{T}$
- We have some observations of this mapping, $\mathcal{D} = \{(\boldsymbol{x}_i, t_i)\}_{i=1}^N$

Goal: predict t for new input $\boldsymbol{x}$ $\longrightarrow$ Learn an accurate prediction function $\hat{t}(\boldsymbol{x})$ (regression function) from $\mathcal{D}$.

Linear regression:

Assumes relationship between $\boldsymbol{x}$ and t is linear:

$$\begin{aligned} t(\boldsymbol{x}) &= w_0 + w_1 x_1 + \ldots + w_d x_d \\ &= \mu(\boldsymbol{x}, \boldsymbol{w}) \end{aligned}$$

Linear regression

Linear regression (2):

Linear combinations of fixed nonlinear functions of the input variables,

$$\mu(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x})$$

- $\phi_j(\boldsymbol{x})$: basis functions
- w_0: bias, allows for any fixed offset
- Convenient to define dummy basis function $\phi_0(\boldsymbol{x}) = 1$, so that

$$\mu(\boldsymbol{x}, \boldsymbol{w}) = \sum_{j=0}^{M} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x})$$

with $\boldsymbol{w} = (w_0, \ldots, w_M)^{\mathsf{T}}$ and $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_0(\boldsymbol{x}), \ldots, \phi_M(\boldsymbol{x}))^{\mathsf{T}}$

Linear regression IBM

Examples of basis functions

- Polynomial linear regression:

$$\mu(x, \boldsymbol{w}) = \sum_{j=0}^{M} w_j x^j = \boldsymbol{w}^\mathsf{T} \boldsymbol{\phi}(x)$$

  with $\boldsymbol{w} = (w_0, \ldots, w_M)^\mathsf{T}$ and $\boldsymbol{\phi}(x) = (1, x, x^2 \ldots, x^M)^\mathsf{T}$

- Gaussian basis functions:

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

- Sigmoidal basis functions:

$$\phi_j(x) = \sigma\left(\frac{x - \alpha_j}{s}\right)$$

  with

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

Linear regression

- Can express our uncertainty over value of $\mu(\boldsymbol{x}, \boldsymbol{w})$ as

$$t(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}) + \varepsilon = \mu(\boldsymbol{x}, \boldsymbol{w}) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \beta^{-1})$$

- Given $\boldsymbol{x}$, t modeled by probabilistic model

$$\mathsf{t}|\mathbf{x} = \boldsymbol{x} \sim \mathcal{N}\left(\mu(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1}\right)$$

  equivalently,

$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = p(t|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}\left(t|\mu(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1}\right), \qquad \boldsymbol{\theta} = (\boldsymbol{w}, \beta)$$

Discriminative vs. generative probabilistic models

Discriminative probabilistic models:

- Posterior (predictive) distribution $p(t|\boldsymbol{x})$ assumed to belong to a hypothesis class $p(t|\boldsymbol{x}, \boldsymbol{\theta})$ defined by $\boldsymbol{\theta}$
- $\boldsymbol{\theta}$ learned from $\mathcal{D}$
- Once model is learned, compute

$$\hat{t}_{\mathcal{D}}(\boldsymbol{x}) = \arg\min_{\hat{t}} \ \mathbb{E}_{\mathrm{t} \sim p_{\mathcal{D}}(t|\boldsymbol{x})}[\ell(\mathrm{t}, \hat{t})|\boldsymbol{x}]$$

Example: Linear regression

1. Learn $\boldsymbol{\theta} = (\boldsymbol{w}, \beta)$ based on $\mathcal{D}$
2. Optimal prediction (under quadratic loss function):

$$\hat{t}_{\mathcal{D}}(\boldsymbol{x}) = \mathbb{E}_{\mathrm{t} \sim p(\mathrm{t}|\boldsymbol{x}, \boldsymbol{\theta}^*_{\mathcal{D}})}[\mathrm{t}|\boldsymbol{x}]$$

i.e.,

$$\hat{t}_{\mathcal{D}}(\boldsymbol{x}) = \mu(\boldsymbol{x}, \boldsymbol{w}^*_{\mathcal{D}})$$

Discriminative vs. generative probabilistic models

Generative probabilistic models:

- Models $p(\boldsymbol{x}, t)$ as being part of a parametric family $p(\boldsymbol{x}, t|\boldsymbol{\theta})$
- Models also $p(\boldsymbol{x})$
- Generative: capacity to generate a realization of $\mathbf{x}$ by using marginal $p(\boldsymbol{x}|\boldsymbol{\theta})$
- Once model is learned, obtain $p(t|\boldsymbol{x}, \boldsymbol{\theta})$ applying Bayes' and compute

$$\hat{t}_{\mathcal{D}}(\boldsymbol{x}) = \arg\min_{\hat{t}} \ \mathbb{E}_{\mathrm{t} \sim p_{\mathcal{D}}(t|\boldsymbol{x})}[\ell(\mathrm{t}, \hat{t})|\boldsymbol{x}]$$

Observations:

- Make stronger assumptions $\longrightarrow$ may lead to more significant bias
- Ability to deal with missing data or latent variables (semi-supervised learning)

Maximum likelihood learning

- Hypothesis class $p(t|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(\mu(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1})$: e.g., definition of the polynomial degree M (defines *capacity* of the class)
- Specific model $p(t|\boldsymbol{x}, \boldsymbol{\theta})$: selection of $\boldsymbol{\theta} = (\boldsymbol{w}, \beta)$ (learned from $\mathcal{D}$)

How do we learn the model parameters $\boldsymbol{\theta}$ (for a given hypothesis class)?

Idea: Maximum likelihood (ML) learning
Select $\boldsymbol{\theta}$ such that $\mathcal{D}$ has maximum probability of being observed

- Need to write distribution observed labels $t_{\mathcal{D}} \in \mathcal{D}$ given $x_{\mathcal{D}}$,

$$p(t_{\mathcal{D}}|x_{\mathcal{D}}, \boldsymbol{w}, \beta) = \prod_{i=1}^{N} p(t_i|\boldsymbol{x}_i, \boldsymbol{w}, \beta)$$
$$= \prod_{i=1}^{N} \mathcal{N}\left(t_i|\mu(\boldsymbol{x}_i, \boldsymbol{w}), \beta^{-1}\right)$$

Maximum likelihood learning

Log-likelihood function:

$$\ln p(t_{\mathcal{D}}|x_{\mathcal{D}}, \boldsymbol{w}, \beta) = \sum_{i=1}^{N} \ln p(t_i|\boldsymbol{x}_i, \boldsymbol{w}, \beta)$$

$$= -\frac{\beta}{2} \sum_{i=1}^{N} \left(t_i - \mu(\boldsymbol{x}_i, \boldsymbol{w})\right)^2 + \frac{N}{2} \ln \frac{\beta}{2\pi}$$

ML criterion (cross-entropy loss or log-loss):

$$(\boldsymbol{w}_{\mathsf{ML}}, \beta_{\mathsf{ML}}) = \arg\max_{\boldsymbol{w},\beta} \; p(t_{\mathcal{D}}|x_{\mathcal{D}}, \boldsymbol{w}, \beta)$$

$$= \arg\max_{\boldsymbol{w},\beta} \; \frac{1}{N} \ln p(t_{\mathcal{D}}|x_{\mathcal{D}}, \boldsymbol{w}, \beta)$$

$$= \arg\min_{\boldsymbol{w},\beta} \; -\frac{1}{N} \ln p(t_{\mathcal{D}}|x_{\mathcal{D}}, \boldsymbol{w}, \beta)$$

$$= \arg\min_{\boldsymbol{w},\beta} \; \frac{\beta}{2N} \sum_{i=1}^{N} \left(t_i - \mu(\boldsymbol{x}_i, \boldsymbol{w})\right)^2 - \frac{1}{2} \ln \frac{\beta}{2\pi}$$

Maximum likelihood learning

If only interested in learning posterior mean:

$$\begin{aligned}\boldsymbol{w}_{\mathsf{ML}} &= \arg\min_{\boldsymbol{w}} \ \frac{\beta}{2N}\sum_{i=1}^{N}(t_i - \mu(\boldsymbol{x}_i, \boldsymbol{w}))^2 \\ &= \arg\min_{\boldsymbol{w}} \ \frac{1}{N}\sum_{i=1}^{N}(t_i - \mu(\boldsymbol{x}_i, \boldsymbol{w}))^2 \\ &= \arg\min_{\boldsymbol{w}} \ L_{\mathcal{D}}(\boldsymbol{w})\end{aligned}$$

$L_{\mathcal{D}}(\boldsymbol{w})$: training loss

Criterion coincides with direct inference via empirical risk minimization if we parametrize the predictor as $\hat{t}(\boldsymbol{x}) = \mu(\boldsymbol{x}, \boldsymbol{w})$!

Minimizing $L_{\mathcal{D}}(\boldsymbol{w})$ can be solved in closed form: $\boldsymbol{w}_{\mathsf{ML}} = (\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{t}_{\mathcal{D}}$, with $\boldsymbol{\Phi} = (\boldsymbol{\phi}(\boldsymbol{x}_1)\ \boldsymbol{\phi}(\boldsymbol{x}_2)\cdots\boldsymbol{\phi}(\boldsymbol{x}_N))^{\mathsf{T}}$

Maximum likelihood learning

To predict value of t for new input $\boldsymbol{x}$, we use $\boldsymbol{w}_{\mathsf{ML}}$:

$$\hat{t} = \boldsymbol{w}_{\mathsf{ML}}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}) = \left((\boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{t}_{\mathcal{D}} \right)^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x})$$

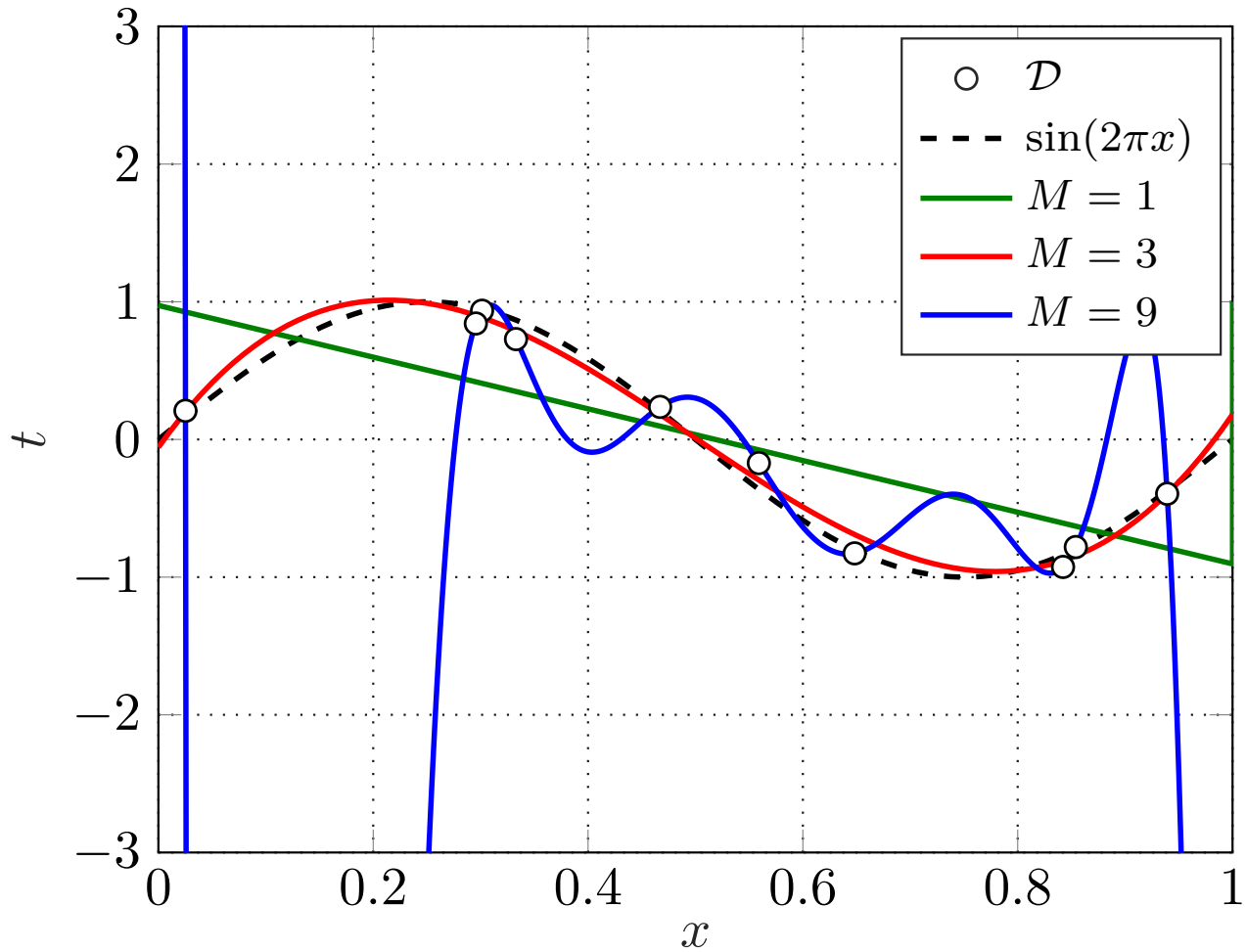
Once $\boldsymbol{w}_{\mathsf{ML}}$ is determined, we can determine β_{ML} as:

$$\beta_{\mathsf{ML}} = \arg\min_{\beta} \; \frac{\beta}{2N} \sum_{i=1}^{N} \left(t_i - \mu(\boldsymbol{x}_i, \boldsymbol{w}_{\mathsf{ML}}) \right)^2 - \frac{1}{2} \ln \frac{\beta}{2\pi}$$

i.e.,

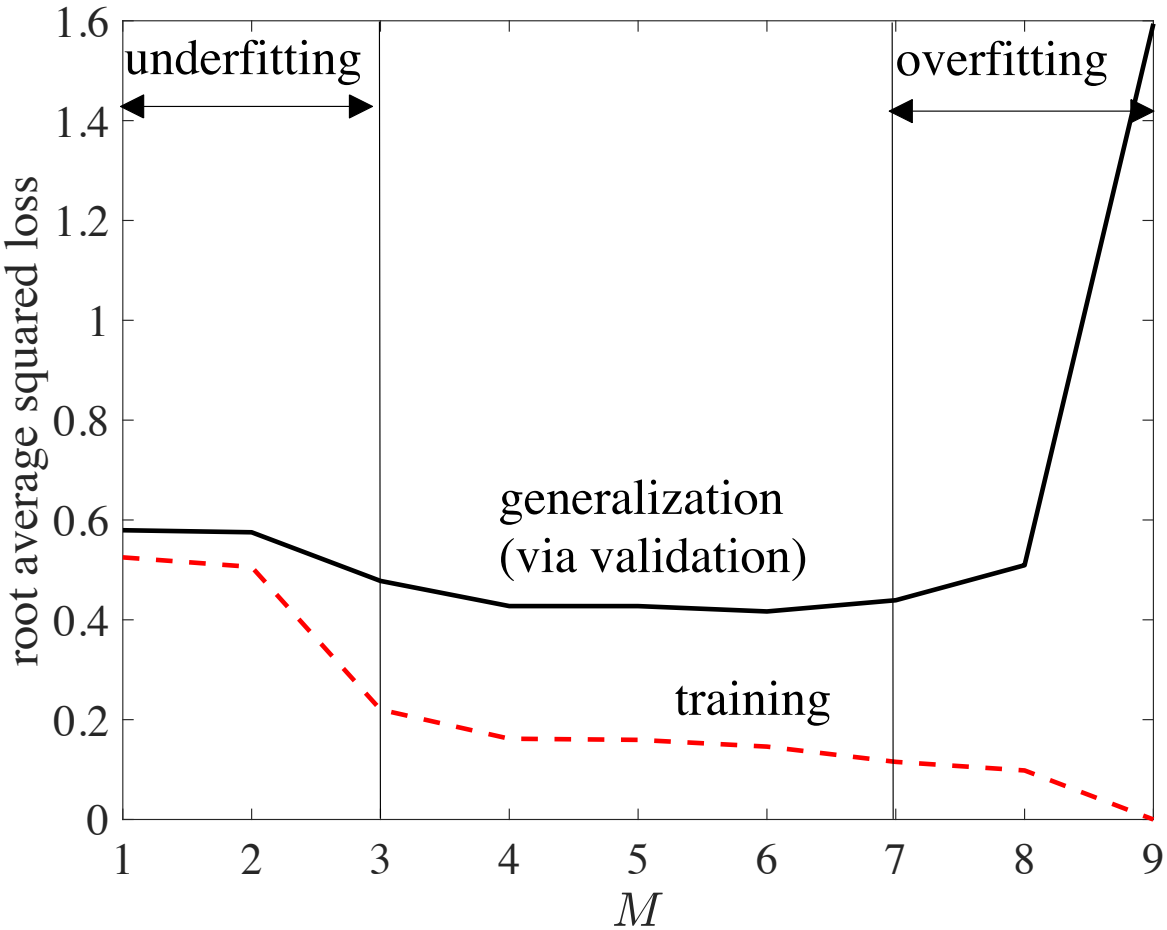
$$\frac{1}{\beta_{\mathsf{ML}}} = \frac{1}{N} \sum_{i=1}^{N} \left(t_i - \mu(\boldsymbol{x}_i, \boldsymbol{w}_{\mathsf{ML}}) \right)^2 = L_{\mathcal{D}}(\boldsymbol{w}_{\mathsf{ML}})$$

Maximum likelihood learning



- $p(x, t) = p(x)p(t|x)$, x $\sim \mathcal{U}(0, 1)$, t|x $\sim \mathcal{N}(\sin(2\pi x), 0.1)$
- Optimal predictor under quadratic loss: $\hat{t}^* = \sin(2\pi x)$
- ML predictor: $\hat{t}_{\mathsf{ML}}(x) = \mu(x, \boldsymbol{w}_{\mathsf{ML}})$

Training and generalization loss



- Training loss: $L_{\mathcal{D}}(\boldsymbol{w}_{\mathsf{ML}}) = \frac{1}{N}\sum_{i=1}^{N}\left(t_i - \mu(x_i, \boldsymbol{w}_{\mathsf{ML}})\right)^2$
- Generalization loss: $L_p(\boldsymbol{w}_{\mathsf{ML}}) = \mathbb{E}_{\mathbf{x},\mathsf{t}}[\ell(\mathsf{t}, \hat{t}_{\mathsf{ML}}(\mathbf{x}))] = \mathbb{E}_{\mathbf{x},\mathsf{t}}[\ell(\mathsf{t}, \boldsymbol{w}_{\mathsf{ML}}^{\mathsf{T}}\phi(\boldsymbol{x}))]$

Training and generalization loss

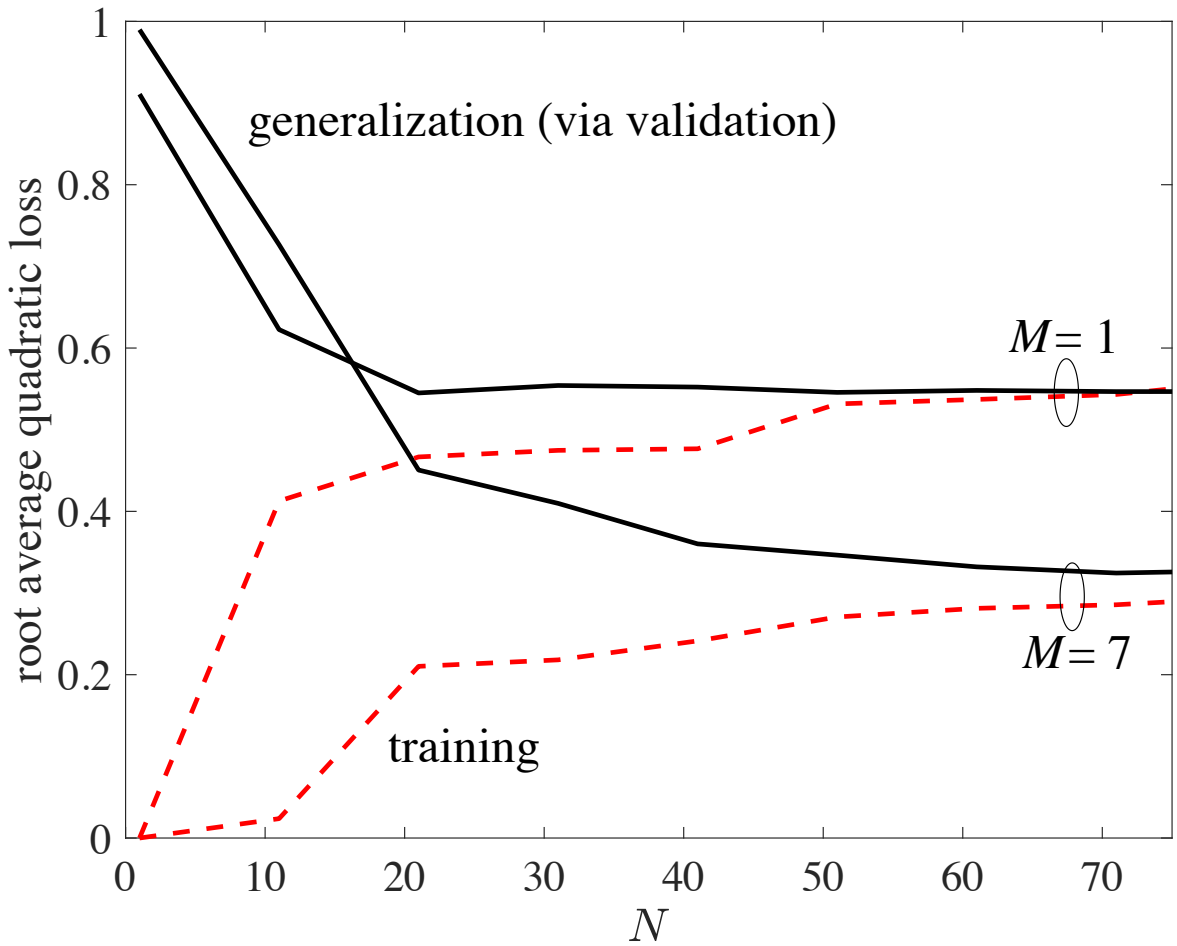
- As the number or data points increases, overfitting is avoided
- When $\mathcal{D}$ is big compared to the number of parameters in $\boldsymbol{\theta}$,

$$L_{\mathcal{D}}(\boldsymbol{w}) \simeq L_p(\boldsymbol{w})$$

- For large N,

$$\boldsymbol{w}_{\text{ML}} \longrightarrow \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} L_p(\boldsymbol{w})$$

Training and generalization loss



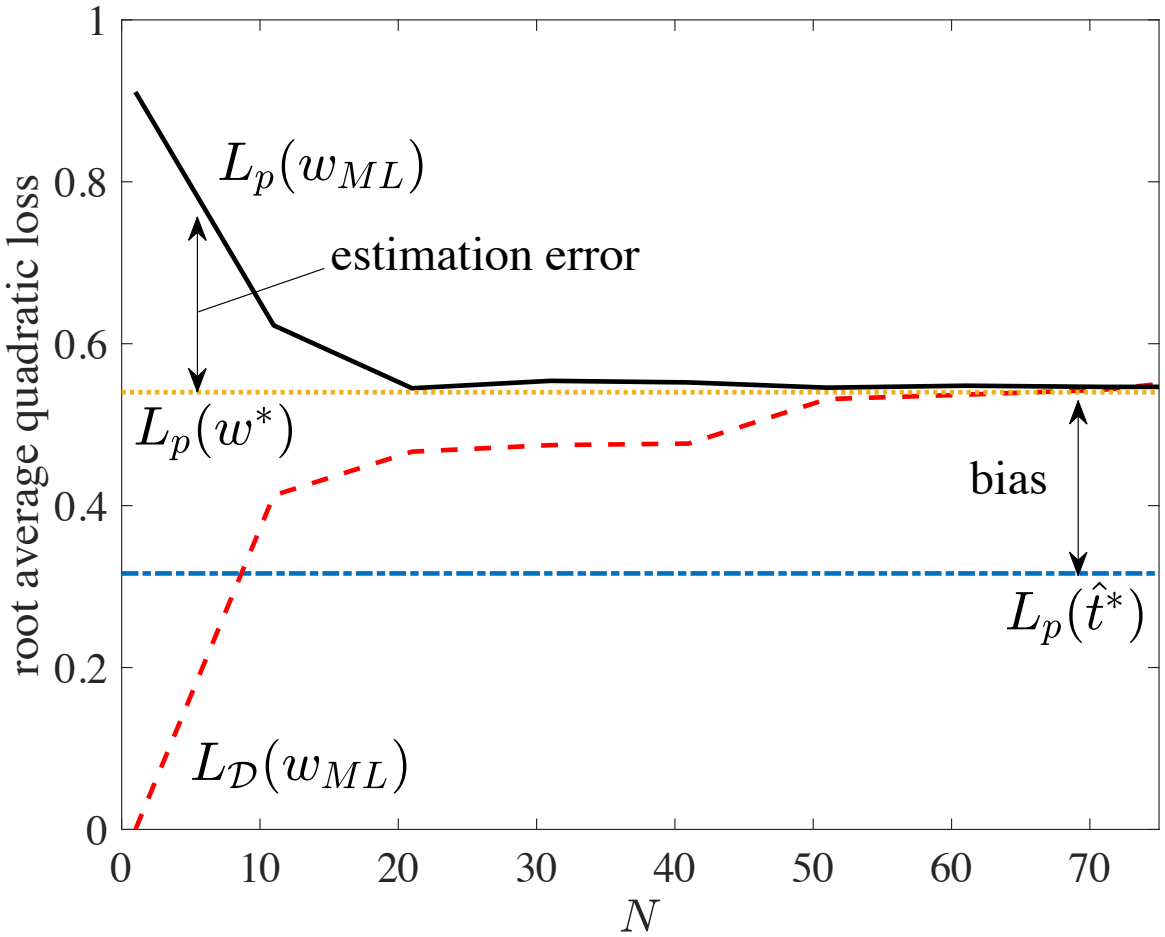
- Squared root of $L_{\mathcal{D}}(\boldsymbol{w}_{\mathsf{ML}})$ and $L_p(\boldsymbol{w}_{\mathsf{ML}})$
- Overfitting decreases with increasing N

Bias and estimation error

$$L_p(\boldsymbol{w}_{\text{ML}}) = L_p(\hat{t}^*) + \underbrace{(L_p(\boldsymbol{w}^*) - L_p(\hat{t}^*))}_{\text{bias}} + \underbrace{(L_p(\boldsymbol{w}_{\text{ML}}) - L_p(\boldsymbol{w}^*))}_{\text{estimation error}}$$

- $L_p(\hat{t}^*)$: generalization loss of optimal predictor (minimum achievable loss)
- Bias (approximation error): caused by choice of hypothesis class
- Estimation error (generalization gap): Caused by the fact that N is not large enough

Bias and estimation error



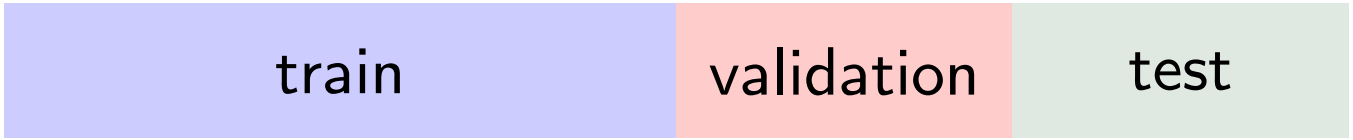
- Estimation error decreases with N and vanishes for large N

Learning and validation

We would like to choose model (hyperparameters and parameters) such that generalization error $L_p(\hat{t}) = \mathbb{E}_{\mathbf{x},\mathrm{t}}[\ell(\mathrm{t}, \hat{t}(\mathbf{x}))]$ is minimized

. . . but it depends on $p(\boldsymbol{x}, t)$ (unknown)!

Learning and validation



Validation: Divide available data into three sets

- Training set $\mathcal{D}$: To fit models
- Validation set $\mathcal{V}$: To choose hypothesis class via evaluation of approximation of the generalization error

$$L_p \approx \frac{1}{N_{\mathsf{v}}} \sum_{i=1}^{N_{\mathsf{v}}} \ell(t_i, \mu(\boldsymbol{x}_i, \boldsymbol{w}))$$

- For selected hypothesis class retrain $\boldsymbol{\theta}$ based on $\mathcal{D} \cup \mathcal{V}$
- Test set $\mathcal{T}$: To produce estimate generalization error obtained with final model

Typically: 50% – 25% – 25%

Cross-validation

Pitfall of validation:

Part of available data not used for training!

Validation suitable when having plenty of data.

Alternative: k-fold cross-validation

1. Randomly partition data points into k partitions (folds)
2. For each partition $\kappa \in \{1, \ldots, k\}$, train model over all other $k-1$ partitions
3. Compute generalization error on the κ-th partition
4. Generalization error estimated as average over all partitions
5. Choose hypothesis class that minimizes estimate of generalization error in step 4

Supervised machine learning: The frequentist approach

Goal: Given a training set $\mathcal{D} = \{(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)\}$, identify an algorithm to predict the label t for a new (yet unseen) point $\boldsymbol{x}$.

Linear regression model:

$$t(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}) + \varepsilon = \mu(\boldsymbol{x}, \boldsymbol{w}) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \beta^{-1})$$

Assumptions:

1. **x**: observed random variable
2. t: observed random variable
3. ε: unknown random variable
4. β: (un)known deterministic variable
5. $\boldsymbol{w}$: unknown deterministic

Supervised machine learning: The Bayesian approach

Goal: Given a training set $\mathcal{D} = \{(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)\}$, identify an algorithm to predict the label t for a new (yet unseen) point $\boldsymbol{x}$.

Linear regression model:

$$t(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}) + \varepsilon = \mu(\boldsymbol{x}, \boldsymbol{w}) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \beta^{-1})$$
$$\boldsymbol{w} \sim p(\boldsymbol{w})$$

Assumptions:

1. $\mathbf{x}$: observed random variable
2. t: observed random variable
3. ε: unknown random variable
4. β: (un)known deterministic variable
5. $\boldsymbol{w}$: unknown random variable

Provides the uncertainty of the prediction

Maximum a posteriori learning

- Observation: ML learning introduces a tension between bias (larger M) and estimation error (smaller M)
- MAP: Enables a finer control of bias and estimation error

Key idea: Leverage prior information available on the behavior of parameters in the absence, or presence, of overfitting.

- Observation: A large value of $\|\boldsymbol{w}\|$ a manifestation of overfitting $\longrightarrow$ introduce prior on $\boldsymbol{w}$ that gives lower probability to larger values,

$$\boldsymbol{w} \sim \mathcal{N}(0, \alpha^{-1}\boldsymbol{I})$$

i.e., $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|0, \alpha^{-1}\boldsymbol{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left(-\frac{\alpha}{2}\boldsymbol{w}^{\mathsf{T}}\boldsymbol{w}\right)$

How do we choose the prior distribution?

Common approach: choose a conjugate prior

- Likelihood function

$$p(t_{\mathcal{D}}|x_{\mathcal{D}}, \boldsymbol{w}, \beta) = \prod_{i=1}^{N} \mathcal{N}\left(t_i|\mu(\boldsymbol{x}_i, \boldsymbol{w}), \beta^{-1}\right)$$

Exponential of a quadratic function of $\boldsymbol{w} \longrightarrow$ Conjugate given by a Gaussian distribution of the form

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_0, \boldsymbol{S}_0)$$

Maximum a posteriori learning

MAP criterion:

$$(\boldsymbol{w}_{\text{MAP}}, \beta) = \arg\max_{\boldsymbol{w},\beta} \; p(t_{\mathcal{D}}, \boldsymbol{w} | x_{\mathcal{D}}, \beta)$$

$$= \arg\max_{\boldsymbol{w},\beta} \; p(\boldsymbol{w}) \prod_{i=1}^{N} p(t_i | \boldsymbol{x}_i, \boldsymbol{w}, \beta)$$

Equivalently,

$$(\boldsymbol{w}_{\text{MAP}}, \beta) = \arg\min_{\boldsymbol{w},\beta} \; -\sum_{i=1}^{N} \ln p(t_i | \boldsymbol{x}_i, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w})$$

If β a known constant,

$$\boldsymbol{w}_{\text{MAP}} = \arg\min_{\boldsymbol{w}} \; -\sum_{i=1}^{N} \ln p(t_i | \boldsymbol{x}_i, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w})$$

MAP: Equivalent to maximizing posterior distribution of $\boldsymbol{w}$ given the available data, $p(\boldsymbol{w}|\mathcal{D})$.

Maximum a posteriori learning

Assuming $\boldsymbol{w} \sim \mathcal{N}(0, \alpha^{-1}\boldsymbol{I})$, i.e., $p(\boldsymbol{w}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left(-\frac{\alpha}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{w}\right)$:

$$
\begin{aligned}
\boldsymbol{w}_{\mathsf{MAP}} &= \arg\min_{\boldsymbol{w}} \; -\sum_{i=1}^{N} \ln p(t_i|\boldsymbol{x}_i, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w}) \\
&= \arg\min_{\boldsymbol{w}} \; -\left(-\frac{\beta}{2}\sum_{i=1}^{N}\left(\mu(\boldsymbol{x}_i, \boldsymbol{w}) - t_i\right)^2 + \frac{N}{2}\ln\frac{\beta}{2\pi}\right) - \ln p(\boldsymbol{w}) \\
&= \arg\min_{\boldsymbol{w}} \; \frac{\beta}{2}\sum_{i=1}^{N}\left(\mu(\boldsymbol{x}_i, \boldsymbol{w}) - t_i\right)^2 - \ln p(\boldsymbol{w}) \\
&= \arg\min_{\boldsymbol{w}} \; \frac{1}{N}\sum_{i=1}^{N}\left(\mu(\boldsymbol{x}_i, \boldsymbol{w}) - t_i\right)^2 - \frac{2}{N\beta}\ln p(\boldsymbol{w}) \\
&= \arg\min_{\boldsymbol{w}} \; \frac{1}{N}\sum_{i=1}^{N}\left(\mu(\boldsymbol{x}_i, \boldsymbol{w}) - t_i\right)^2 + \frac{\alpha}{N\beta}\boldsymbol{w}^\mathsf{T}\boldsymbol{w} \\
&= \arg\min_{\boldsymbol{w}} \; L_{\mathcal{D}}(\boldsymbol{w}) + \frac{\lambda}{N}\|\boldsymbol{w}\|^2 \longrightarrow \boldsymbol{w}_{\mathsf{MAP}} = (\lambda\boldsymbol{I} + \boldsymbol{\Phi}^\mathsf{T}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{t}_{\mathcal{D}}
\end{aligned}
$$

Maximum a posteriori learning

MAP: Equivalent to maximizing posterior distribution of $\boldsymbol{w}$ given the available data, $p(\boldsymbol{w}|\mathcal{D})$.

Consider $p(\boldsymbol{w}|t_{\mathcal{D}}, x_{\mathcal{D}}, \beta)$:

- Due to conjugate Gaussian prior distribution, posterior also Gaussian,

$$p(\boldsymbol{w}|t_{\mathcal{D}}, x_{\mathcal{D}}, \beta) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_N, \boldsymbol{S}_N)$$

  with

$$\boldsymbol{m}_N = \boldsymbol{S}_N \left(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \beta\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{t}\right)$$
$$\boldsymbol{S}_N^{-1} = \boldsymbol{S}_0^{-1} + \beta\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi}$$

- We assume $\boldsymbol{w} \sim \mathcal{N}(0, \alpha^{-1}\boldsymbol{I})$,

$$\boldsymbol{m}_N = \beta\boldsymbol{S}_N\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{t}_{\mathcal{D}}$$
$$\boldsymbol{S}_N^{-1} = \alpha\boldsymbol{I} + \beta\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi}$$

Maximum a posteriori learning

Applying Bayes',

$$
\begin{aligned}
p(\boldsymbol{w}|t_{\mathcal{D}}, x_{\mathcal{D}}, \beta) &= \frac{p(t_{\mathcal{D}}|x_{\mathcal{D}}, \boldsymbol{w}, \beta)p(\boldsymbol{w})}{p(t_{\mathcal{D}})} \\
&\propto p(\boldsymbol{w})p(t_{\mathcal{D}}|x_{\mathcal{D}}, \boldsymbol{w}, \beta) \\
&= p(\boldsymbol{w}) \prod_{i=1}^{N} p(t_i|\boldsymbol{x}_i, \boldsymbol{w}, \beta)
\end{aligned}
$$

- $p(\boldsymbol{w})$: knowledge about $\boldsymbol{w}$ before observing any data
- $p(t_{\mathcal{D}}|\boldsymbol{w})$: how likely the observed data is for a particular parameter value
- $p(\boldsymbol{w}|t_{\mathcal{D}})$: knowledge about $\boldsymbol{w}$ from the observed data and the model

Maximum a posteriori learning

MAP criterion for $\boldsymbol{w}$:

$$
\begin{aligned}
\boldsymbol{w}_{\text{MAP}} &= \arg\max_{\boldsymbol{w}} \; p(\boldsymbol{w}|t_{\mathcal{D}}, x_{\mathcal{D}}, \beta) \\
&= \arg\max_{\boldsymbol{w}} \; p(\boldsymbol{w}) \prod_{i=1}^{N} p(t_i|\boldsymbol{x}_i, \boldsymbol{w}, \beta) \\
&= \arg\min_{\boldsymbol{w}} \; -\sum_{i=1}^{N} \ln p(t_i|\boldsymbol{x}_i, \boldsymbol{w}, \beta) - \ln p(\boldsymbol{w}) \\
&= \arg\min_{\boldsymbol{w}} \; L_{\mathcal{D}}(\boldsymbol{w}) + \frac{\lambda}{N}\|\boldsymbol{w}\|^2
\end{aligned}
$$

with $\lambda = \alpha/\beta$

Maximum a posteriori learning

$$\boldsymbol{w}_{\text{MAP}} = \arg\min_{\boldsymbol{w}} \; L_{\mathcal{D}}(\boldsymbol{w}) + \frac{\lambda}{N}\|\boldsymbol{w}\|^2$$

Observations:

- As $N \longrightarrow \infty$ MAP estimate tends to ML estimate
- MAP criterion (ridge regression), modifies ML criterion by adding the quadratic (or Tikhonov) regularization function

$$R(\boldsymbol{w}) = \|\boldsymbol{w}\|^2$$

Solution:

$$\boldsymbol{w}_{\text{MAP}} = \boldsymbol{m}_N = \left(\lambda \boldsymbol{I} + \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{t}_{\mathcal{D}}$$

Regularization: Decreases model capacity $\longrightarrow$ Can control overfitting.

Example: Bayesian linear regression (Bishop Figure 3.7)

Problem: fitting straight line to noisy measurements generated from

$$f(x, \boldsymbol{a}) = a_0 + a_1 x, \qquad a_0 = -0.3,\ a_1 = 0.5$$

by adding Gaussian noise $\mathcal{N}(0, 0.04)$.

Model:

$$t(x, \boldsymbol{w}) = \underbrace{w_0 + w_1 x}_{\boldsymbol{w}^\mathsf{T}\boldsymbol{x}} + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, 0.04)$$

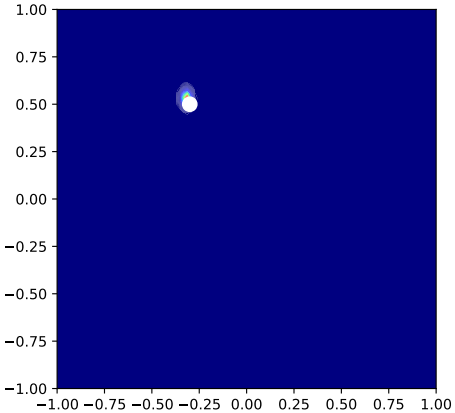
with prior

$$p(\boldsymbol{w}) = \mathcal{N}\left(\boldsymbol{w} \,\middle|\, \begin{pmatrix} 0 & 0 \end{pmatrix}^\mathsf{T}, \alpha^{-1}\boldsymbol{I}_2\right), \qquad \alpha = 2$$

Example: Bayesian linear regression

Plot after two hundred measurements

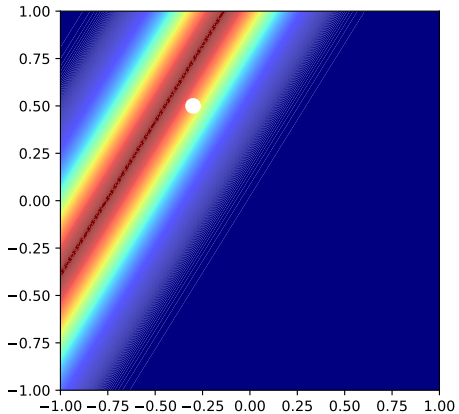
prior



w_1

w_0

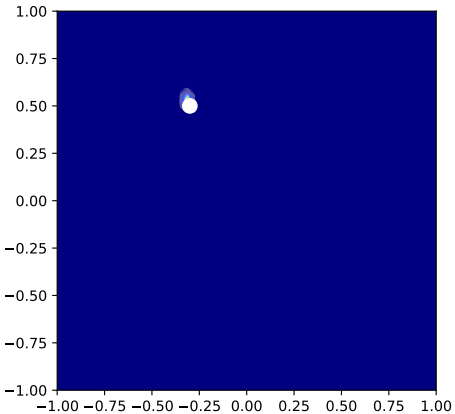
likelihood



w_1

w_0

posterior



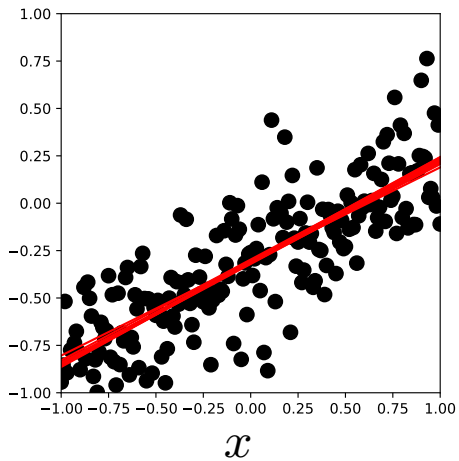
w_1

w_0

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_{199}, \boldsymbol{S}_{199})$$

$$p(t_{200}|\boldsymbol{w}) = \mathcal{N}(t_{200}|w_0 + w_1 x_{200}, \beta^{-1})$$

$$p(\boldsymbol{w}|y_1^{200}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_{200}, \boldsymbol{S}_{200})$$



t

x

Bayesian linear regression

Frequentist approach: Aims at identifying a specific value for $\boldsymbol{\theta}$ of a probabilistic model to derive a predictor

$$\hat{t}^*(x) = \arg\min_{\hat{t}} \mathbb{E}_{\mathbf{t}|\mathbf{x}}[\ell(\mathbf{t}, \hat{t})|\boldsymbol{x}]$$

- ML: Chooses $\boldsymbol{\theta}$ that maximizes probability of training data,
- MAP: Includes also prior information about parameter vector

Bayesian approach: Assumes $\boldsymbol{\theta}$ jointly distributed with data $\longrightarrow$ Does not commit to a single value of $\boldsymbol{\theta}$ but considers explanations provided by all possible values of $\boldsymbol{\theta}$, each weighted according to a data-dependent belief

Bayesian linear regression

Bayesian linear regression model:

$$t(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}) + \varepsilon = \mu(\boldsymbol{x}, \boldsymbol{w}) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \beta^{-1})$$
$$\boldsymbol{w} \sim p(\boldsymbol{w})$$

Probabilistic model:

$$p(t_{\mathcal{D}} | x_{\mathcal{D}}, \boldsymbol{w}, \beta) = \prod_{i=1}^{N} \mathcal{N}\left(t_i | \mu(\boldsymbol{x}_i, \boldsymbol{w}), \beta^{-1}\right) \qquad \text{likelihood}$$
$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} | \boldsymbol{m}_0, \boldsymbol{S}_0) \qquad \text{prior}$$

End goal: Making predictions of t for new values of $\boldsymbol{x} \longrightarrow$ Directly evaluate the posterior distribution $p(t | t_{\mathcal{D}}, x_{\mathcal{D}}, \boldsymbol{x}, \beta)$

A fully Bayesian solution returns the entire posterior $p(t | \mathcal{D}, \boldsymbol{x}, \beta) \longrightarrow$ provides more information about unobserved label t.

Predictive distribution

We obtain

$$p(t|\mathcal{D}, \boldsymbol{x}, \beta) = \int \underbrace{p(\boldsymbol{w}|\mathcal{D}, \beta)}_{\text{posterior dist. of } \boldsymbol{w}} \; p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) d\boldsymbol{w}$$

Bayesian approach: $p(t|\boldsymbol{x}, \boldsymbol{w}, \beta)$ associated with each value of $\boldsymbol{w}$ weighted by the posterior belief

$$p(\boldsymbol{w}|\mathcal{D}, \beta) = \frac{p(\boldsymbol{w}) p(t_{\mathcal{D}}|x_{\mathcal{D}}, \boldsymbol{w}, \beta)}{p(t_{\mathcal{D}}|x_{\mathcal{D}}, \beta)}$$

Predictive distribution

Computing $p(\boldsymbol{w}|\mathcal{D}, \beta)$ and $p(t|\mathcal{D}, \boldsymbol{x}, \beta)$ difficult!

For our Bayesian linear regression problem with $\boldsymbol{w} \sim \mathcal{N}(0, \alpha^{-1}\boldsymbol{I})$:

$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}\left(t|\mu(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1}\right)$$
$$p(\boldsymbol{w}|\mathcal{D}, \beta) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_N, \boldsymbol{S}_N)$$

with

$$\boldsymbol{m}_N = \beta \boldsymbol{S}_N \boldsymbol{\Phi}^\mathsf{T} \boldsymbol{t}, \quad \boldsymbol{S}_N^{-1} = \alpha \boldsymbol{I} + \beta \boldsymbol{\Phi}^\mathsf{T} \boldsymbol{\Phi}$$

We obtain:

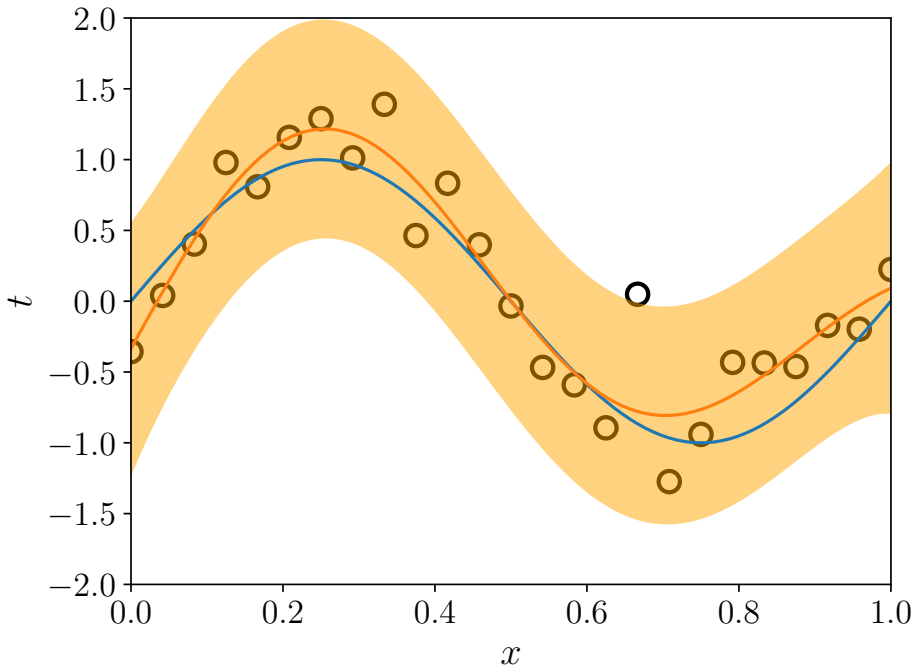
$$p(t|\boldsymbol{x}, \mathcal{D}, \beta) = \mathcal{N}(t|\mu(\boldsymbol{x}, \boldsymbol{w}_{\mathsf{MAP}}), \sigma_N^2(\boldsymbol{x})),$$

with

$$\mu(\boldsymbol{x}, \boldsymbol{w}_{\mathsf{MAP}}) = \boldsymbol{w}_{\mathsf{MAP}}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{m}_N^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x})$$
$$\sigma_N^2(\boldsymbol{x}) = \frac{1}{\beta} + \frac{1}{\beta}\left(\phi(\boldsymbol{x})^\mathsf{T}\left(\lambda \boldsymbol{I} + \boldsymbol{\Phi}^\mathsf{T}\boldsymbol{\Phi}\right)^{-1}\phi(\boldsymbol{x})\right)$$

Predictive distribution (illustration of $p(t|\boldsymbol{x}, \mathcal{D}, \beta)$)

- t|x $\sim \mathcal{N}(\sin(2\pi x), \beta^{-1})$
- Model: $\mu(x, \boldsymbol{w}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}(x)$, with Gaussian basis functions $\phi_j(x)$
- $\alpha = 10^{-3}$, $\beta = 2$



blue: true model, $\sin(2\pi x)$
orange: mean of predictive distribution, $\mu(x, \boldsymbol{w}_{\text{MAP}})$
shaded orange: $\mu(x, \boldsymbol{w}_{\text{MAP}}) \pm \sigma_N(x)$

Reading

"Pattern recognition and machine learning,"

Chapter 1 (1.2.4–1.2.6), Chapter 3 (Intro, 3.1, 3.2, 3.3 (until 3.3.2))