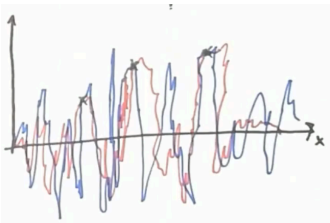
Week 5a

Underfitting, model very certain,
check underfitting: observe residuals all positive -> all negative -> all positive;
observe residual highly correlated

problem similar to overfitting: model too complex (with lots of narrow RBF)
& not enough data
=> posterior distribution very uncertain (similar to prior except at few data points)
e.g.



Model choice: represent belief about which model the features came from
e.g. narrow VS wide distribution (= lots of narrow RBFs); hyperparameters

marginal likelihood **p (Ytrain | Xtrain, M) = p (D | M)**
marginalized out weight w
M = model choice (implicitly always there): contains noise variance, prior mean, prior covariance (RBF position & width), activation functions
x,y: training instance

Use sum & product rule, but can't solve the integral

$$p(\mathbf{y} \mid X, \mathcal{M}) = \int p(\mathbf{y}, \mathbf{w} \mid X, \mathcal{M})\, \mathrm{d}\mathbf{w} = \int p(\mathbf{y} \mid X, \mathbf{w}, \mathcal{M})\, p(\mathbf{w} \mid \mathcal{M})\, \mathrm{d}\mathbf{w},$$

Likelihood. prior

Trick to get p (y | x,M): use posterior weight **p(w | D)** (= LHS) mentioned before:
denominator is marginal likelihood, nominator is standard likelihood & prior
solve marginal likelihood by rearranging terms
could use any w (if w=w could cancel out terms), get the same answer

posterior weights:
$p(\underline{w} \mid \underline{y}, X, M) = \frac{p(\underline{y} \mid \underline{w}, X, M) \cdot p(\underline{w} \mid X, M)}{p(\underline{y} \mid X, M)}$

Apply marginal likelihood (after having it):

apply Bayes rule, get p(D | M)

or apply maximum marginal likelihood:

model with prior

$$p(\mathbf{w} \mid \sigma_w) = \mathcal{N}(\mathbf{w}; \mathbf{0}, \sigma_w^2 \mathbb{I}), \tag{2}$$

and likelihood

$$p(y \mid \mathbf{x}, \mathbf{w}, \sigma_y) = \mathcal{N}(y; \mathbf{w}^\top \mathbf{x}, \sigma_y^2), \tag{3}$$

we can fit the *hyperparameters* σ_w and σ_y, parameters which specify the model, to maximize their marginal likelihood:

Replace M

$$p(\mathbf{y} \mid X, \sigma_w, \sigma_y) = \int p(\mathbf{y}, \mathbf{w} \mid X, \sigma_w, \sigma_y)\, \mathrm{d}\mathbf{w} = \int p(\mathbf{y} \mid X, \mathbf{w}, \sigma_y)\, p(\mathbf{w} \mid \sigma_w)\, \mathrm{d}\mathbf{w}. \tag{4}$$

To choose hyperparameters of M (**on training set only!!!**, since bayesian)

e.g. marginal likelihood of good model >> overfitting model (with narrow basis functions), also >> underfitting model

Since: Like the dice example, one model could only generate smooth curve, overfitting model could generate smooth & overfitted curve, marginal likelihood is much smaller

Week5b

Bayesian optimization: use uncertainty to guide search (do the next experiment in highly uncertain area)

task: e.g. have uncertainty, find greatest y value (acquisition function)

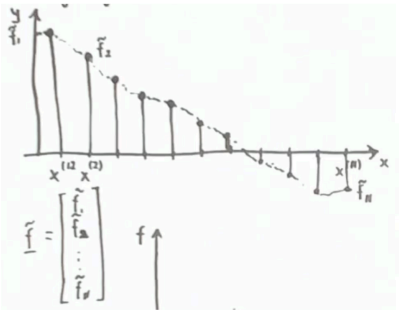
limitation of bay linear regression: too certain if model too simple (**"underfitting"**)

(e.g. when move x to infinity, y of [increasing regression line & 3RBFs])

Gaussian Process prior

Discretized function, represented by huge vector ($\tilde{f}$, many points)

discrete points drawn from normal distribution, defined mainly by (very big) covariance matrix



$$p(\underline{\tilde{f}}) = \mathcal{N}(\underline{\tilde{f}}; \underline{0}, \Sigma)$$

$$\Sigma_{ij} = \mathrm{cov}(\tilde{f}_i, \tilde{f}_j)$$

$$= \mathbb{E}[\tilde{f}_i \tilde{f}_j] - \mathbb{E}[\tilde{f}_i]\mathbb{E}[\tilde{f}_j]$$

set prior entries in covariance: defined by **kernel function** (= relation between 2 points fi fj);

xi xj could be multivariate (for a surface)

kernel: semi-positive definite and symmetric

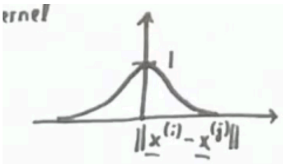
$$\mathrm{cov}[\tilde{f}_i, \tilde{f}_j] = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}).$$

functions for k: **Mercer kernel**, positive semi definite

$$k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2),$$

want smooth curve, so want neighboring points highly correlated, distant points less correlated

kernel function:

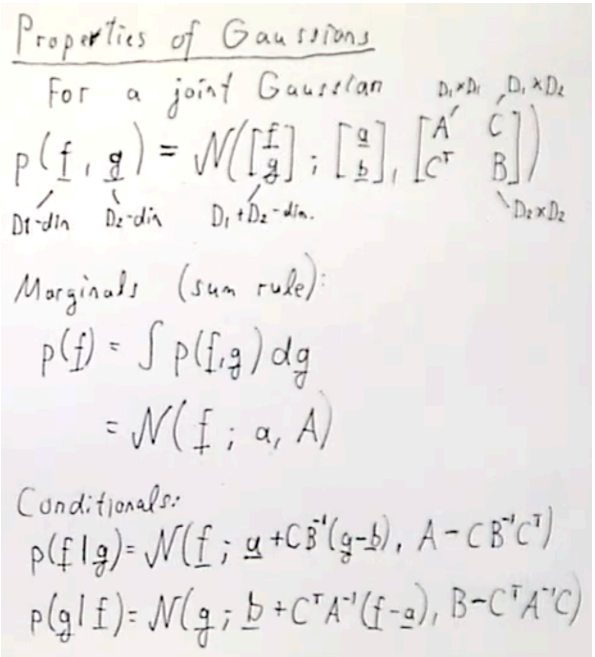


Gaussian Process regression

Gaussian properties:

in covariance, A: cov just consider f; B: cov just consider g;

C and C^T: how f and g related



Terms:

- y: observed training labels
- ~f: real training labels
- ~f*: real test labels
- x, x*: training location x & test location x*

have prior f ~ GP (k),
have noisy observations y (noise around function values ~fn, as equation bellow),
could update our belief (by likelihood)

- for n datapoints, product over n

$$y_n \sim \mathcal{N}(\tilde{f}_n, \sigma_y^2)$$
$$\Rightarrow \text{likelihood } p(y_n \mid \tilde{f}) = \mathcal{N}(y_n; \tilde{f}_n, \sigma_y^2)$$

have unknown true function, joint distribution of ~f and ~f* : p (~f, ~f*)

- D: size of input vector x
- N: # of training locations
- M: # of test locations

$$p(\tilde{f}, \tilde{f}_*) = \mathcal{N}\left(\begin{bmatrix}\tilde{f} \\ \tilde{f}_*\end{bmatrix}; \begin{bmatrix}0 \\ 0\end{bmatrix}, \begin{bmatrix}K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*)\end{bmatrix}\right)$$
$$K(X,Z)_{ij} = k(x^{(i)}, z^{(j)})$$

$K(X,Z)$: $N \times M$; $x^{(i)}$: $M \times D$; $z^{(j)}$: $N \times D$

Prior:
joint distribution of the observations y and ~f* : p(y, ~f*)

(since ~f* only related to ~f; y is just observation noise on top of ~f)

$$p\left(\begin{bmatrix}\mathbf{y} \\ \mathbf{f}_*\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y} \\ \mathbf{f}_*\end{bmatrix}; \mathbf{0}, \begin{bmatrix}K(X,X) + \sigma_y^2\mathbb{I} & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*)\end{bmatrix}\right),$$

= additional observation noise

Do **inference** by GP regression:

- p(~f* | y): same as p(Ytest | Xtest, D)
- just copy values from p(y, ~f*) according to Gaussian properties above
- **Advantage of GP**: just copy values from p(g | f), no calculation needed

Inference

$$p(\tilde{f}_* \mid y) = \mathcal{N}(\tilde{f}_*; \_\_, \_\_)$$

"empty values":

$$\text{mean, } \bar{\mathbf{f}}_* = K(X_*, X)(K(X, X) + \sigma_y^2 \mathbb{I})^{-1}\mathbf{y}$$
$$\text{cov}(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_y^2 \mathbb{I})^{-1}K(X, X_*)$$

cov matrix depends on input location x, but does not depends on training label y
so a surprising training label have little effect

Can we be more uncertain at prediction in response to a surprising training label?

=> change GP kernel function, learn the parameters in kernel (choose hyperparmeter)

Visualizing the posterior