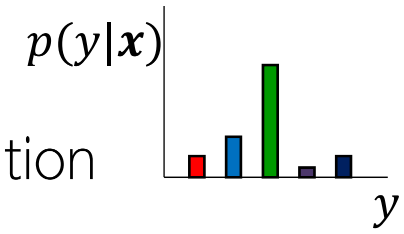
Probabilistic Supervised Learning: Linear Regression

CS772A: Probabilistic Machine Learning

Piyush Rai

Probabilistic Supervised Learning

- Goal: To learn the conditional distribution $p(y|x)$ of output given input
- The form of the distribution $p(y|x)$ depends on output type, e.g.,
  - Real: Model $p(y|\boldsymbol{x})$ using a Gaussian (or some other suitable real-valued distribution)
  - Binary: Model $p(y|\boldsymbol{x})$ using a Bernoulli
  - Categorical/multiclass: Model $p(y|\boldsymbol{x})$ using a multinoulli/categorical distribution
  - Various other types (e.g., count, positive reals, etc) can also be modeled using appropriate distributions (e.g., Poisson for count, gamma for positive reals)



- The distribution $p(y|\boldsymbol{x})$ can be defined directly or indirectly

"Direct" way without modeling the inputs $\boldsymbol{x}_n$

Parameters of this distribution are the outputs of function f

$$p(y|\boldsymbol{x}) = p(y|f(\boldsymbol{x}, \boldsymbol{w}))$$

"Indirect" way by modeling the outputs as well as the inputs

"Indirect" way requires first learning the joint distribution of inputs and outputs

$$p(y|\boldsymbol{x}) = \frac{p(y, \boldsymbol{x})}{p(\boldsymbol{x})}$$

Discriminative vs Generative Sup. Learning

Non-probabilistic supervised learning approaches (e.g., SVM) are usually considered discriminative since $p(\boldsymbol{x})$ is never modeled

- Direct way of sup. learning is discriminative, indirect way is generative

Discriminative Approach

$$p(y|\boldsymbol{x}) = p(y|f(\boldsymbol{x}, \boldsymbol{w}))$$

f can be any function which uses inputs and weights $\boldsymbol{w}$ to defines parameters of distr. $\boldsymbol{p}$

Some examples

$$p(y|\boldsymbol{x}) = \mathcal{N}(y|\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1})$$

$$p(y|\boldsymbol{x}) = \text{Bernoulli}(y|\sigma(\boldsymbol{w}^\top \boldsymbol{x}))$$

Generative Approach

$$p(y|\boldsymbol{x}) = \frac{p(y, \boldsymbol{x})}{p(\boldsymbol{x})}$$

Requires estimating the joint distribution of inputs and outputs to get the conditional $p(y|\boldsymbol{x})$ (unlike the discriminative approach which directly estimates the conditional $p(y|\boldsymbol{x})$ and does not model the distribution of $\boldsymbol{x}$)

- Note: Generative approach can also be used for other settings too, such as unsupervised learning and semi-supervised learning (will see later)

Probabilistic Linear Regression

A discriminative model for regression problems

- Assume training data $\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$, with features $\boldsymbol{x}_n \in \mathbb{R}^D$ and responses $y_n \in \mathbb{R}$

Unknown to be estimated

Each weight assumed real-valued

- Assume y_n generated by a noisy linear model with wts $\boldsymbol{w} = [w_1, \dots, w_D] \in \mathbb{R}^D$

$$y_n = \boldsymbol{w}^\top \boldsymbol{x}_n + \epsilon_n$$

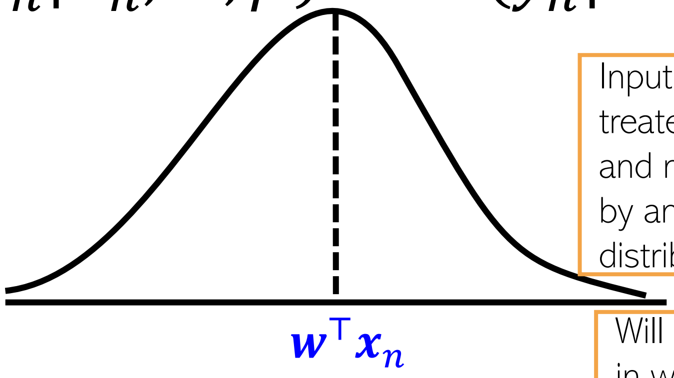
Gaussian noise drawn from $\mathcal{N}(\epsilon_n | 0, \beta^{-1})$

- Notation alert: β is the precision of Gaussian noise (and β^{-1} the variance)

Unknown to be estimated

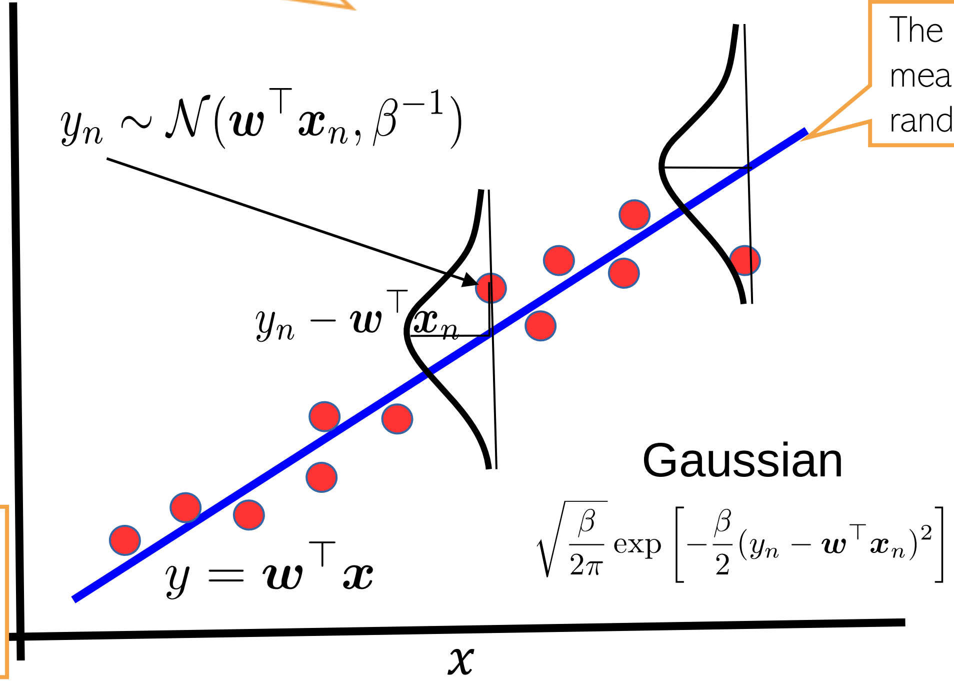
Likelihood model

$$p(y_n | \boldsymbol{x}_n, \boldsymbol{w}, \beta) = \mathcal{N}(y_n | \boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1})$$



Input x_n being treated as given and not modeled by any probability distribution

Will later study models in which both input and output are modeled by distributions



The line represents the mean $\boldsymbol{w}^\top \boldsymbol{x}_n$ of the output random variable y_n

The zero mean Gaussian noise perturbs the output from its mean

Thus NLL is like squared loss

Probabilistic Linear Regression

- For all the training data, we can write the above model in matrix-vector notation

$\boldsymbol{y} = [y_1; y_2; \ldots; y_N]$ is the $N{\times}1$ response vector

$\boldsymbol{X} = [\boldsymbol{x}_1^\top; \boldsymbol{x}_2^\top; \ldots; \boldsymbol{x}_N^\top]$ is the $N \times D$ input matrix

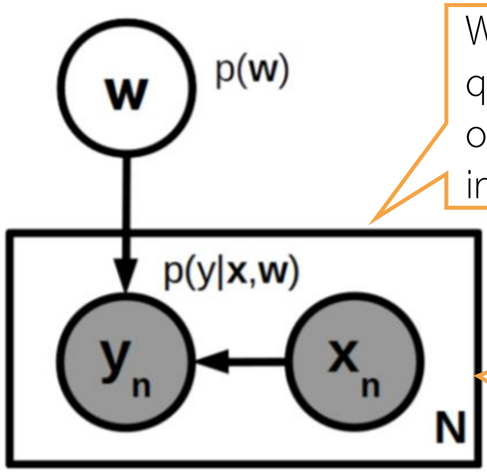
$\boldsymbol{\epsilon} = [\epsilon_1; \epsilon_2; \ldots; \epsilon_N]$ is the $N{\times}1$ noise vector drawn from $\mathcal{N}(\mathbf{0}, \beta^{-1}\boldsymbol{I}_N)$

Same as writing
$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}, \beta) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{w}, \beta^{-1}\boldsymbol{I}_N)$

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon}$$

- This is a linear Gaussian model with $\boldsymbol{w}$ being the unknown Gaussian r.v.
- A simple "plate diagram" for this model would look like this (hyperparameters not shown in the diagram)

Direction of arrow show dependency



White nodes denote unknown quantities, grey nodes denote observed quantities (training input-output pairs)

The plate/box with number N shows that we have N such i.i.d. observations

On compact notations..

- When writing the likelihood (assuming y_n's are i.i.d. given $\boldsymbol{w}$ and $\boldsymbol{x}_n$)

$$p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w},\beta) = \prod_{n=1}^{N} \mathcal{N}(y_n|\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1})$$

$$= \mathcal{N}(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{w}, \beta^{-1}\boldsymbol{I}_N)$$

- Thus a product of $\boldsymbol{N}$ univariate Gaussians here (not always) is equivalent to an $\boldsymbol{N}$-dim Gaussian over the vector $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]$

- We will prefer to use this equivalence at other places too whenever we have multiple i.i.d. random variables, each having a univariate Gaussian distribution

Prior on weights

- Assume a zero-mean Gaussian prior on $\boldsymbol{w}$

May also use a non-zero mean Gaussian prior, e.g., $\mathcal{N}(w_d|\mu, \lambda^{-1})$ if we expect weights to be close to some value μ

This prior assumes that *a priori* each weight has a small value (close to zero)

$$p(\boldsymbol{w}|\lambda) = \prod_{d=1}^{D} p(w_d|\lambda) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda^{-1})$$

λ controls the uncertainty around our prior belief about value of w_d

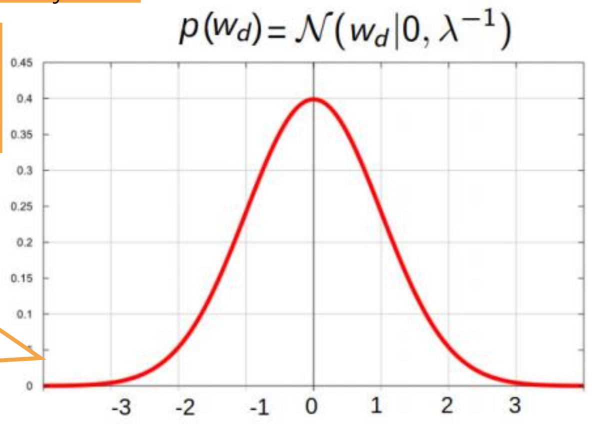
$$= \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \lambda^{-1}\mathbf{I}_D)$$

$$\propto \left(\frac{\lambda}{2\pi}\right)^{\frac{D}{2}} \exp\left[-\frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w}\right]$$

In zero-mean case, λ sort of denotes each feature's importance. Think why?

Large λ means more aggressive push towards zero

The precision λ controls how aggressively the prior pushes w_d towards mean (0)



Can also use a full covariance matrix Λ^{-1} for the prior to impose a priori correlations among different weights

Prior's hyperparameters $(\lambda/\Lambda/\mu)$ etc can be learned as well using point estimation (e.g., MLE-II) or fully Bayesian inference

- Zero-mean Gaussian prior corresponds to ℓ_2 regularizer

Reason: The negative log prior $-\log p(\boldsymbol{w}) \propto \frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w}$

The Posterior

MLE/MAP left as an exercise

- The posterior over $\boldsymbol{w}$ (for now, assume hyperparams $\boldsymbol{\beta}$ and $\boldsymbol{\lambda}$ to be known)

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}, \beta, \lambda) = \frac{p(\mathbf{w}|\lambda)p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \beta)}{p(\mathbf{y}|\mathbf{X}, \beta, \lambda)} \propto p(\mathbf{w}|\lambda)p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \beta)$$

Marginal likelihood for this regression model. Note that it is conditioned on $\mathbf{X}$ too which is assumed given and not being modeled

Must be a Gaussian due to conjugacy

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}, \beta, \lambda) \propto \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda^{-1}\mathbf{I}_D) \times \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \beta^{-1}\mathbf{I}_N)$$

- Using the "completing the squares" trick (or linear Gaussian model results)

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}, \beta, \lambda) = \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$$

Note that $\boldsymbol{\lambda}$ and $\boldsymbol{\beta}$ can be learned under the probabilistic set-up (though assumed fixed as of now)

$$\text{where} \quad \boldsymbol{\Sigma}_N = (\beta \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top + \lambda \mathbf{I}_D)^{-1} = (\beta \mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_D)^{-1} \quad \text{(posterior's covariance matrix)}$$

The form is also similar to the solution to ridge regression $\text{argmin}_w ||y - Xw||^2 + \lambda w^\top w = (X^\top X + \lambda I)^{-1} X^\top y$

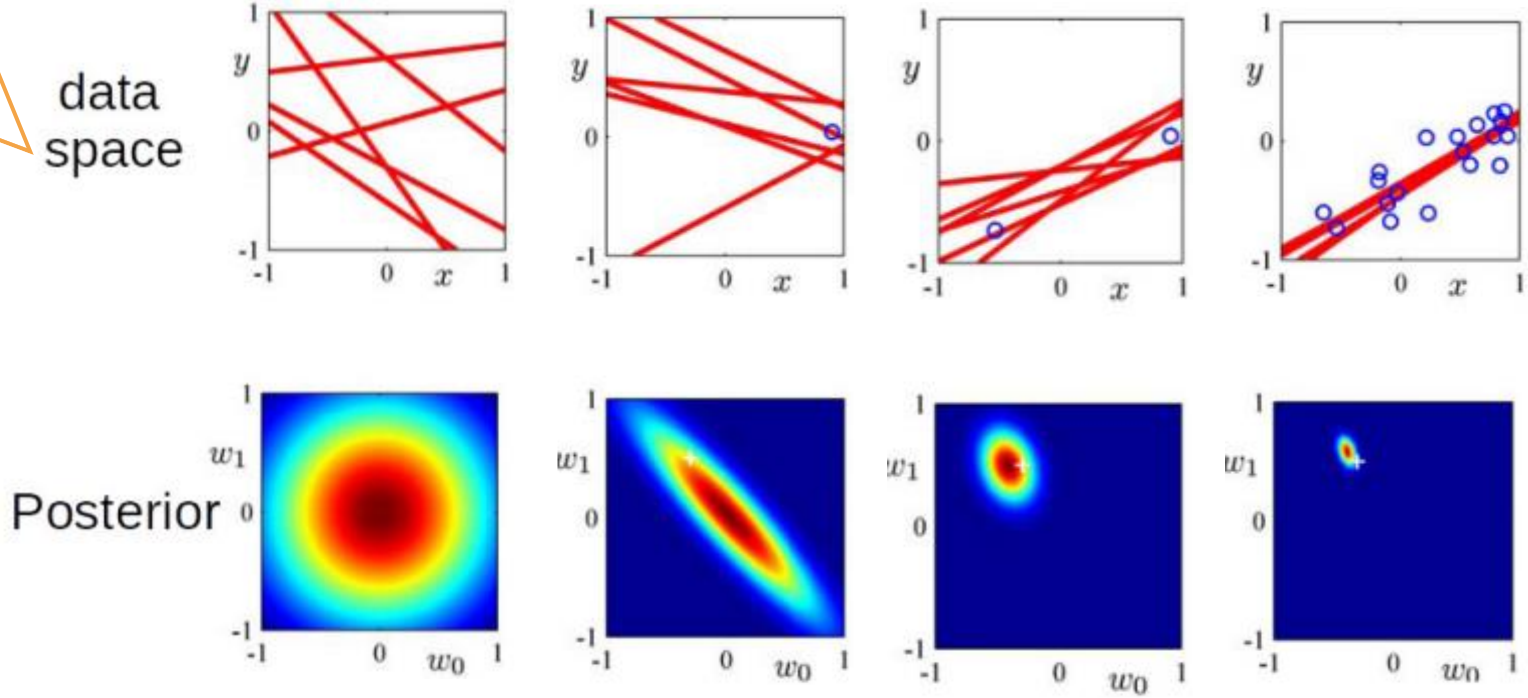
MAP solution turns out to be exactly the same (reason: Gaussian's mean and mode are the same)

$$\boldsymbol{\mu}_N = \boldsymbol{\Sigma}_N \left[\beta \sum_{n=1}^{N} y_n \mathbf{x}_n\right] = \boldsymbol{\Sigma}_N \left[\beta \mathbf{X}^\top \mathbf{y}\right] = (\mathbf{X}^\top \mathbf{X} + \frac{\lambda}{\beta} \mathbf{I}_D)^{-1} \mathbf{X}^\top \mathbf{y} \quad \text{(posterior's mean)}$$

The Posterior: A Visualization

- Assume a lin. reg. problem with true $\boldsymbol{w} = [w_0, w_1], w_0 = -0.3, w_1 = 0.5$
- Assume data generated by a linear regression model $y = w_0 + w_1 x + \text{"noise"}$
  - Note: It's actually 1-D regression (w_0 is just a bias term), or 2-D reg. with feature $[1, x]$
- Figures below show the "data space" and posterior of $\boldsymbol{w}$ for different number of observations (note: with no observations, the posterior = prior)

Each red line represents the "data" generated for a randomly drawn $\boldsymbol{w}$ from the current posterior



Posterior Predictive Distribution

- To get the prediction $\boldsymbol{y}_*$ for a new input $\boldsymbol{x}_*$, we can compute its PPD

$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}, \beta, \lambda) = \int \underbrace{p(y_*|\mathbf{x}_*, \mathbf{w}, \beta)}_{\mathcal{N}(y_*|\boldsymbol{w}^\top \boldsymbol{x}_*, \beta^{-1})} \underbrace{p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \beta, \lambda)}_{\mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)} d\mathbf{w}$$

Only $\boldsymbol{w}$ is unknown with a posterior distribution so only $\boldsymbol{w}$ has to be integrated out

- The above is the marginalization of $\boldsymbol{w}$ from $\mathcal{N}(y_*|\boldsymbol{w}^\top \boldsymbol{x}_*, \beta^{-1})$. Using LGM results

$$\boxed{p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}, \beta, \lambda) = \mathcal{N}(\boldsymbol{\mu}_N^\top \mathbf{x}_*, \beta^{-1} + \mathbf{x}_*^\top \boldsymbol{\Sigma}_N \mathbf{x}_*)}$$

Can also derive it by writing $y_* = \boldsymbol{w}^\top \boldsymbol{x}_* + \epsilon$ where $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$ and $\epsilon \sim \mathcal{N}(0, \beta^{-1})$

- So we have a predictive mean $\boldsymbol{\mu}_N^\top \boldsymbol{x}_*$ as well as an input-specific predictive variance
- In contrast, MLE and MAP make "plug-in" predictions (using the point estimate of $\boldsymbol{w}$)

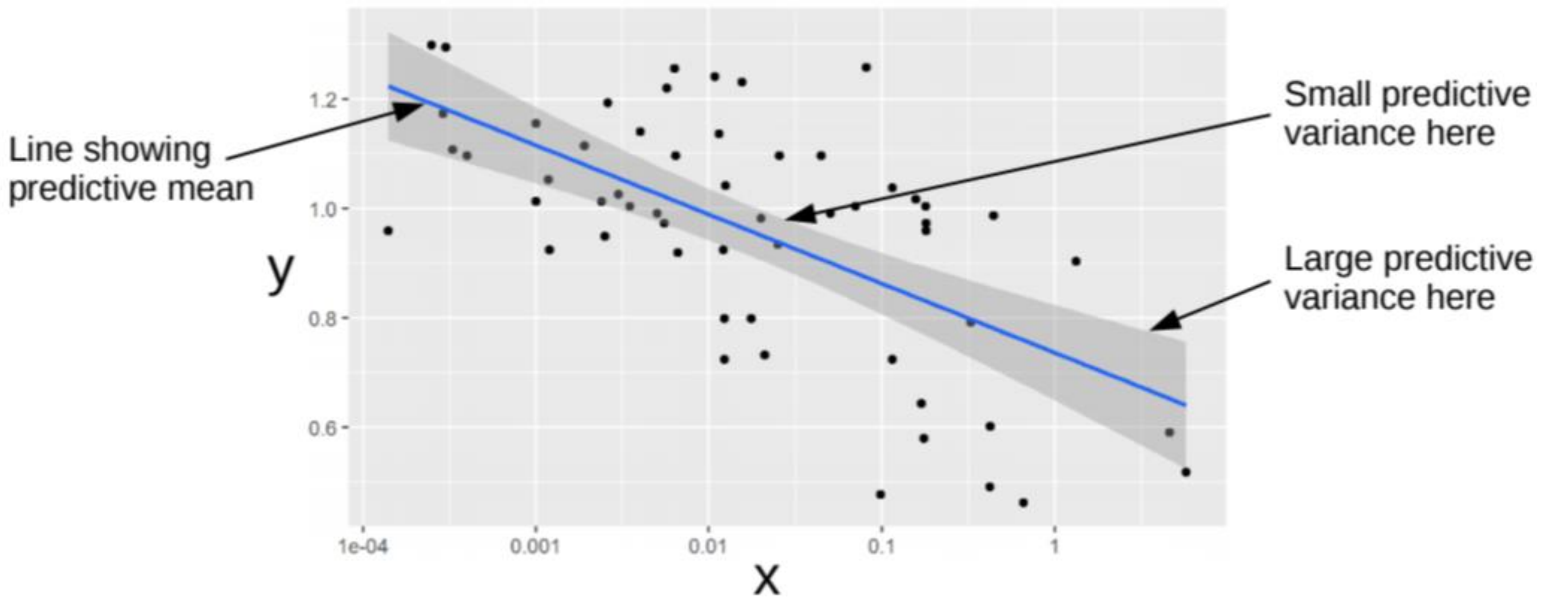
$$p(y_*|\mathbf{x}_*, \mathbf{w}_{MLE}) = \mathcal{N}(\mathbf{w}_{MLE}^\top \mathbf{x}_*, \beta^{-1}) \quad \text{- MLE prediction}$$
$$p(y_*|\mathbf{x}_*, \mathbf{w}_{MAP}) = \mathcal{N}(\mathbf{w}_{MAP}^\top \mathbf{x}_*, \beta^{-1}) \quad \text{- MAP prediction}$$

Since PPD also takes into account the uncertainty in $\boldsymbol{w}$, the predictive variance is larger

- Unlike MLE/MAP, variance of $\boldsymbol{y}_*$ also depends on the input $\boldsymbol{x}_*$ (this, as we will see later, will be very useful in sequential decision-making problems such as active learning)

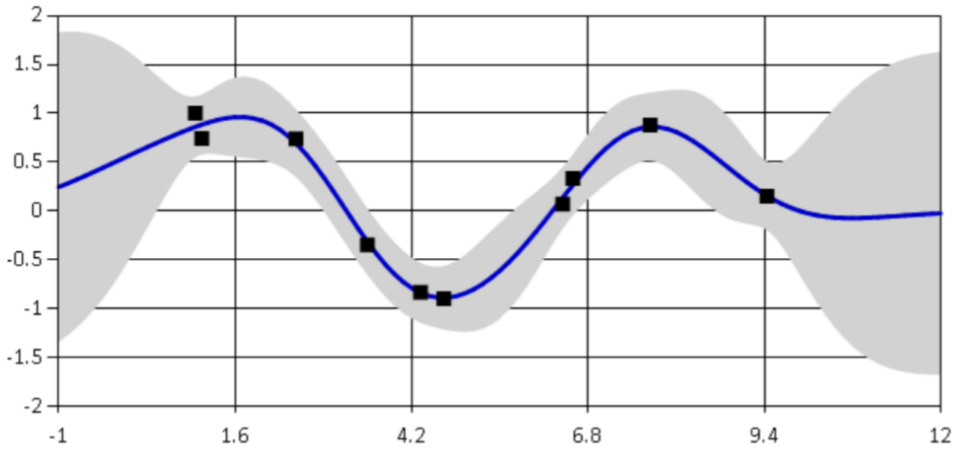
Posterior Predictive Distribution: An Illustration

- Black dots are training examples



- Width of the shaded region at any x denotes the predictive uncertainty at that x (+/- one std-dev)
- Regions with more training examples have smaller predictive variance

Nonlinear Regression



- Can extend the linear regression model to handle nonlinear regression problems

- One way is to replace the feature vectors $\boldsymbol{x}$ by a nonlinear mapping $\boldsymbol{\phi(x)}$

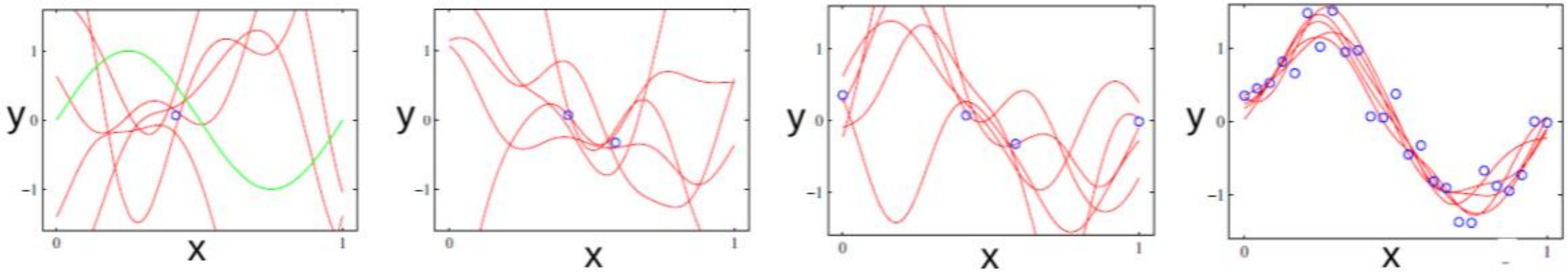
$$p(y|\boldsymbol{x}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}^\top \phi(\boldsymbol{x}), \beta^{-1})$$

Can be pre-defined (e.g., replace a scalar x by polynomial mapping $[1, x, x^2]$) or extracted by a pretrained deep neural net

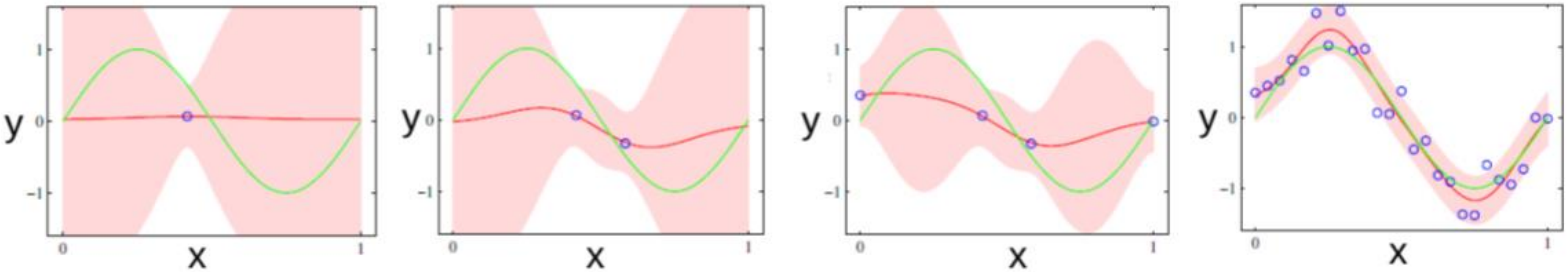
- Alternatively, a kernel function can be used to implicitly define the nonlinear mapping
- More on nonlinear regression when we discuss Gaussian Processes

More on Visualization of Uncertainty

- Figures below: Green curve is the true function and blue circles are observations
- Posterior of the nonlinear regression model: Some curves drawn from the posterior



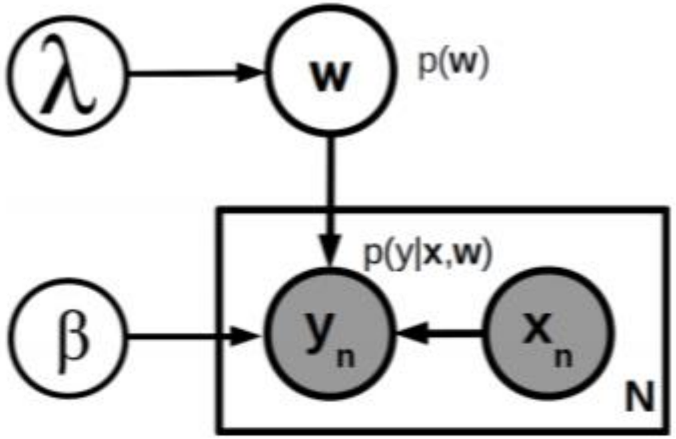
- PPD: Red curve is predictive mean, shaded region denotes predictive uncertainty



Estimating Hyperparameters via MLE-II

- The probabilistic linear reg. model we saw had two hyperparams $(\boldsymbol{\beta}, \boldsymbol{\lambda})$
  - Thus total three unknowns $(\boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{\lambda})$

Need posterior over all the 3 unknowns



$$p(\mathbf{w}, \beta, \lambda | \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta, \lambda)p(\mathbf{w}, \lambda, \beta)}{p(\mathbf{y}|\mathbf{X})}$$

$$= \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta, \lambda)p(\mathbf{w}|\lambda)p(\beta)p(\lambda)}{\int p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta)p(\mathbf{w}|\lambda)p(\beta)p(\lambda)\, d\mathbf{w}\, d\lambda d\beta}$$

PPD would require integrating out all 3 unknowns

$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int p(y_*|\mathbf{x}_*, \mathbf{w}, \beta)p(\mathbf{w}, \beta, \lambda|\mathbf{X}, \mathbf{y})\, d\mathbf{w}\, d\beta\, d\lambda$$

- Posterior and PPD computation is intractable.

Called "MLE-II" because we are maximizing marginal likelihood, not the likelihood

- If we just want point estimates for $(\boldsymbol{\beta}, \boldsymbol{\lambda})$ then MLE-II is an option

Will see various other methods like EM, variational inference, MCMC, etc later

And then compute $p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}, \hat{\beta}, \hat{\lambda})$ treating $\hat{\beta}, \hat{\lambda}$ as given

$$(\hat{\beta}, \hat{\lambda}) = \operatorname{argmax}_{\beta,\lambda} \log p(\boldsymbol{y}|\boldsymbol{X}, \beta, \lambda)$$

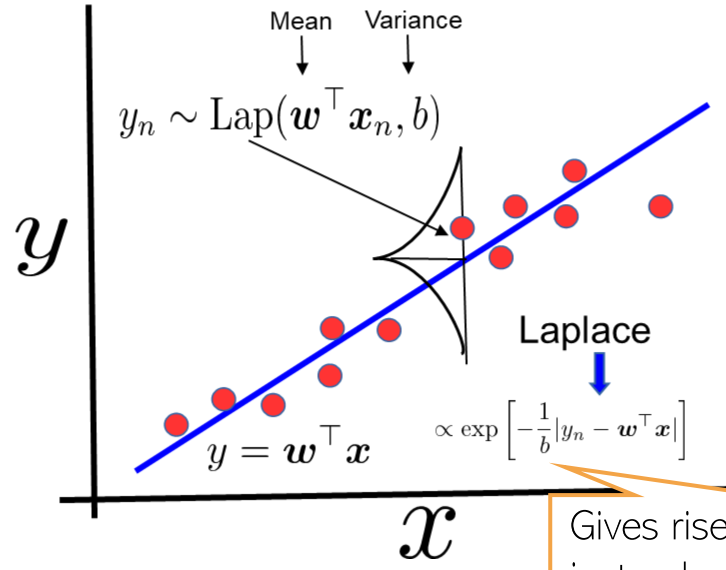
For regression with Gaussian likelihood and Gaussian prior on $\boldsymbol{w}$, the marginal likelihood has an exact expression

Prob. Linear Regression: Some Other Variations

- Can use other likelihoods $p(y_n|\boldsymbol{x}_n, \boldsymbol{w})$ and/or prior distribution $p(\boldsymbol{w})$

- Laplace distribution for the likelihood

$$p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \text{Lap}(y_n|\boldsymbol{w}^\top \boldsymbol{x}_n, b)$$



More robust to outliers than squared loss

Gives rise to "absolute loss" instead of squared loss

- Heteroskedastic noise in the likelihood, e.g.,

$$p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \mathcal{N}(y_n|\boldsymbol{w}^\top \boldsymbol{x}_n, \beta_n^{-1})$$

Can even assume β_n to depend on input $\boldsymbol{x}_n$

Different noise distribution $\mathcal{N}(0, \beta_n^{-1})$ for each y_n

- Feature-specific variances in the prior for $\boldsymbol{w}$

$$p(\boldsymbol{w}) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda_d^{-1}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \boldsymbol{\Lambda}^{-1})$$

Diagonal precision/covariance matrix with λ_d's along the columns of Λ

This has the effect of having feature-specific regularization

Since we can also learn these precisions (e.g., using MLE-II), using such a prior, we can learn the importance of different features (feature selection) which isn't possible with a $\mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \lambda^{-1}\mathbf{I})$ prior with spherical covariance