Linear Models of Regression

K Sri Rama Murty

IIT Hyderabad

ksrm@ee.iith.ac.in

Regression

- Predict target variable(s) $t \in \mathbb{R}$ given D-dimensional input vector $\mathbf{x}$
- E.g. Weight estimation, Share market prediction, 3D image from 2D
- Target can be estimated as a linear combination of inputs

$$\hat{t} = y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots w_D x_D = \mathbf{w}^{\mathsf{T}} \mathbf{x}$$

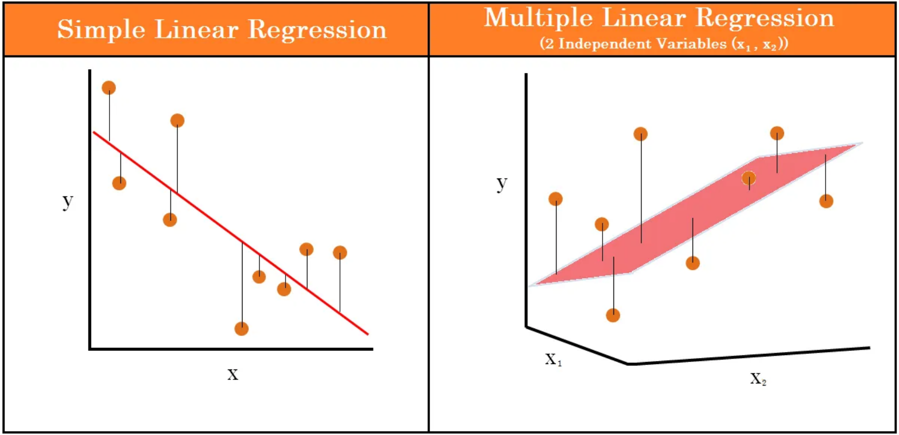
$$\mathbf{x} = [1 \ x_1 \ x_2 \ \cdots \ x_D]^{\mathsf{T}} \qquad \mathbf{w} = [w_0 \ w_1 \ w_2 \ \cdots w_D]^{\mathsf{T}}$$

- Determine the model parameters $\mathbf{w}$ to minimize error on labeled training data

$$\mathcal{S} = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots (\mathbf{x}_N, t_N)\}$$

- Need to define a loss function for optimizing model parameters $\mathbf{w}$

Illustration of Linear Regression



Least Squares Criterion to Determine $\mathbf{w}$

- Estimated target for n^{th} sample in the dataset $\mathcal{S}$

$$\hat{t}_n = y(\mathbf{x}_n, \mathbf{w}) = \mathbf{w}^\mathsf{T}\mathbf{x}_n \qquad n = 1, 2, \cdots, N$$

- Given the ground truth target t_n, the error in estimation

$$e_n = t_n - y_n \quad n = 1, 2, \cdots N$$

- For an arbitrary choice of parameters $\mathbf{w}$, overall error on training set

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} e_n^2$$

- Estimate $\mathbf{w}$ to minimize loss function $J(\mathbf{w})$ on the training dataset $\mathcal{S}$

$$\mathbf{w}_* = \arg\min_{w} J(\mathbf{w})$$

Estimating Optimal $\mathbf{w}$

- Let the input data be organized in the form of a $N \times (D+1)$ matrix

$$\mathbf{X} = [\mathbf{x}_1 \; \mathbf{x}_2 \; \cdots \; \mathbf{x}_N]^{\mathsf{T}}$$

- Estimated output vector $\mathbf{y}$ for all data points is given by

$$\mathbf{y}_{N\times 1} = \mathbf{X}_{N\times D+1}\mathbf{w}_{D+1x1}$$

- Loss function: Trace of the outer product of error vector $\mathbf{e} = \mathbf{t} - \mathbf{y}$

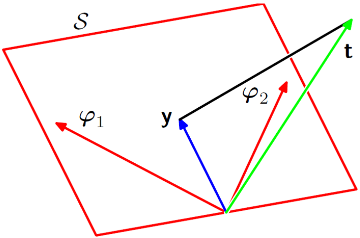
$$J(\mathbf{w}) = \frac{1}{2}\,\mathrm{Tr}[\mathbf{e}\;\mathbf{e}^{\mathsf{T}}]$$

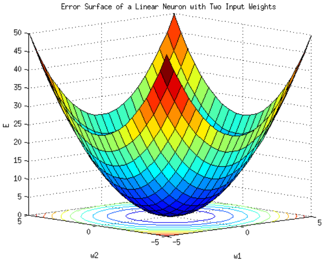
- Equating derivative of loss function w.r.t $\mathbf{w}$ to $\mathbf{0}$

$$\begin{aligned} \nabla_{\mathbf{w}} J(\mathbf{w}) &= \nabla_{\mathbf{w}}\mathbf{y}\;\nabla_{\mathbf{y}}\mathbf{e}\;\nabla_{\mathbf{e}} J(\mathbf{w}) \\ &= -\mathbf{X}^{\mathsf{T}}(\mathbf{t} - \mathbf{X}\mathbf{w}) = \mathbf{0} \end{aligned}$$

$$\boxed{\mathbf{w} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{t}}$$

Geometric Interpretation of Least Squares





- Given N examples, the target vector $\mathbf{t} \in \mathbb{R}^N$ and columns of $\mathbf{X} \in \mathbb{R}^N$
- Let $\mathcal{S}$ denote a subspace spanned by columns of X in N-dim space
- $\mathbf{y} = \mathbf{Xw} \in \mathcal{S}$, being a linear combination of columns of $\mathbf{X}$
- For the LS optimality criterion
  - $\mathbf{y}$ is orthogonal projection of $\mathbf{t}$ on $\mathcal{S}$
  - Error surface $J(\mathbf{w})$ is convex
  - Sim. to Wiener filter: $\mathbf{w} = \mathbf{R}_{xx}^{-1} \mathrm{r}_{xt}$
  - Also referred to as pseudo-inverse sol.

Homework

- Prove the following matrix derivatives

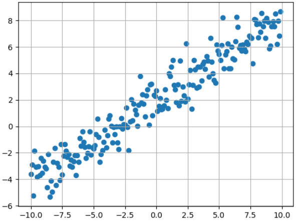
$$\nabla_{\mathbf{X}} \operatorname{Tr}(\mathbf{XA}) = \mathbf{A}^{\mathsf{T}} \qquad \nabla_{\mathbf{X}} \operatorname{Tr}(\mathbf{XA}) = \mathbf{A}^{\mathsf{T}} \qquad \nabla_{\mathbf{X}} \operatorname{Tr}(\mathbf{AX}^{\mathsf{T}}) = \mathbf{A}$$

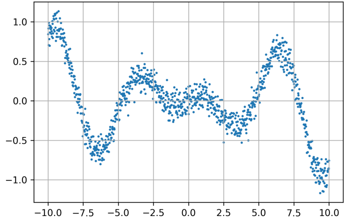
$$\nabla_{\mathbf{X}} \operatorname{Tr}(\mathbf{AXB}) = \mathbf{A}^{\mathsf{T}}\mathbf{B}^{\mathsf{T}} \qquad \nabla_{\mathbf{X}} \operatorname{Tr}(\mathbf{AX}^{\mathsf{T}}\mathbf{B}) = \mathbf{BA}$$

$$\nabla_{\mathbf{X}} \operatorname{Tr}(\mathbf{XAX}^{\mathsf{T}}) = \mathbf{X}(\mathbf{A} + \mathbf{A}^{\mathsf{T}})$$

- True or False: Two nonzero, noncollinear vectors span $\mathbb{R}^2$. If your answer is true express an arbitrary vector $\mathbf{a}$ in terms of two noncollinear vectors $\mathbf{v}_1$ and $\mathbf{v}_2$.
- Prove that there cannot be any other w_0 that gives a lower error than w_* to least squares formulation.
- Extend the linear regression concept to estimate multivariate target vectors $\mathbf{t}_n \in \mathbb{R}^K$

Nonlinear Input-Output Relations





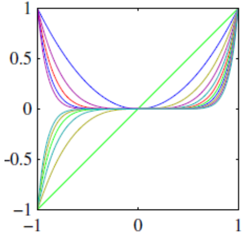
- Polynomial curve fitting can be used to model nonlinear i/o relation

$$\begin{aligned}\hat{t} = y(x, \mathbf{w}) &= w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M \\ &= \mathbf{w}^\mathsf{T} \phi(x) \qquad \text{(Model is linear in } \mathbf{w}\text{)}\end{aligned}$$

- $\phi(.) : \mathbb{R}^1 \to \mathbb{R}^M$ - nonlinear transformation to higher dim. space

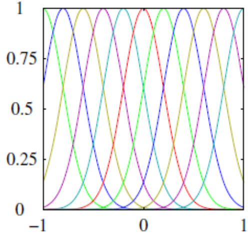
Kernel Examples

Polynomial



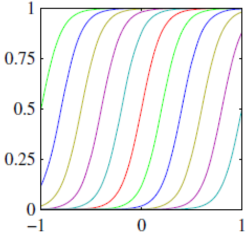
$\phi_j(x) = x^j$

Gaussian



$\phi_j(x) = \exp\left(-\frac{||x-\mu_j||^2}{2\sigma^2}\right)$

Logistic



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

- Global vs Local kernels
  - Changes in one region of input space affect all other regions
  - Local kernels are preferable for functions with varying characteristics
- Explicit vs Implicit kernels
  - Explicit representation for $\phi(\mathbf{x})$ is available or not

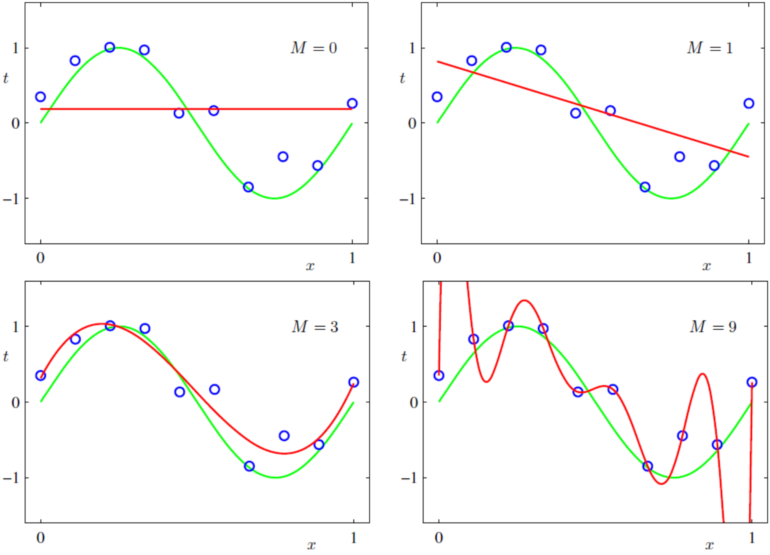
Least Squares Regression in Kernel Space

- If t_n is nonlinearly related to $\mathbf{x}_n$, perform regression in kernel space.
- Let $\phi : \mathbb{R}^D \to \mathbb{R}^{M+1}, \quad M > D$ is a nonlinear kernel mapping
- $\mathbf{x}_n = [x_{n1} \ x_{n2}]^\mathsf{T} \in \mathbb{R}^2$ can be mapped using 2nd order polynomial kernel as $\phi(\mathbf{x}_n) = [1 \ x_{n1} \ x_{n2} \ x_{n1}^2 \ x_{n2}^2 \ x_{n1}x_{n2}]^\mathsf{T} \in \mathbb{R}^6$
- The target t_n is regressed from the kernel representation $\phi(\mathbf{x}_n)$ as

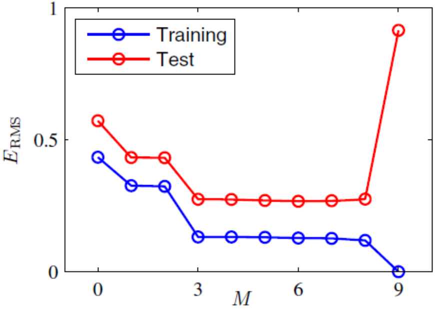
$$\hat{t}_n = \mathbf{w}^\mathsf{T}\phi(\mathbf{x}_n) \qquad \mathbf{w} \in \mathbb{R}^{M+1}$$

- The regression coefficients are given by $\mathbf{w}_* = (\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi})^{-1}\mathbf{\Phi}^\mathsf{T}\mathbf{t}$
  - $\mathbf{\Phi} \in \mathbb{R}^{N \times M+1}$ denotes nonlinearly transformed data matrix
- DNNs can be used to learn data-dependent nonlinear transf. $\boldsymbol{\phi(\mathbf{x}_n)}$
- The last layer of DNNs typically performs linear regression on $\phi(\mathbf{x}_n)$

Effect of Model Order M: $t = \sin(\pi x) + \epsilon$



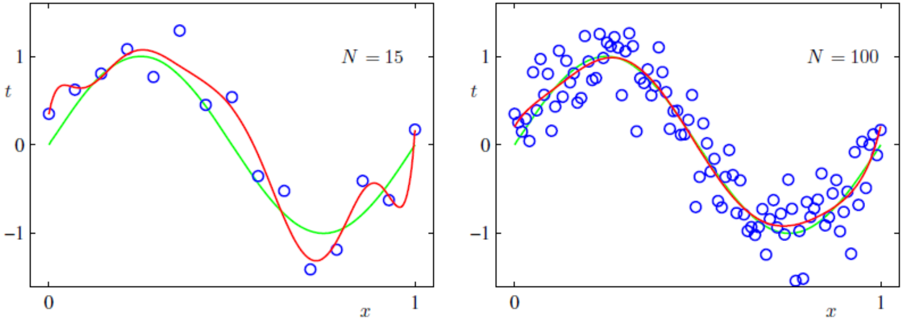
Model Validation



	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

- Training & test error diverge for higher model orders
- Model 'overfits' to the noise in the training data
- Large amplitude weights with alternating polarity.
- $(\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi})$ may be ill conditioned

Amount of Training Data ($M = 9$)



- Overfitting is less severe with increased amount of data.
- Model order cannot be limited by the amount of data available!
- Model order should be based on complexity of task/pattern!
- A way forward: arrest the growth of the model weights

Regularized Least Squares

- Add a penalty term to the error term to discourage weight growth

$$J(\mathbf{w}) = \underbrace{E_D(\mathbf{w})}_{\text{Data Term}} + \underbrace{\lambda E_W(\mathbf{w})}_{\text{Regularization Term}}$$

- λ controls the relative importance of the data and reg. terms
  - Smaller λ results in high variance, higher λ results in high bias
- Sum of squares error function with a quadratic regularizer

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \mathbf{w}^\mathsf{T}\phi(\mathbf{x}_n)\right)^2 + \frac{\lambda}{2}\mathbf{w}^\mathsf{T}\mathbf{w}$$

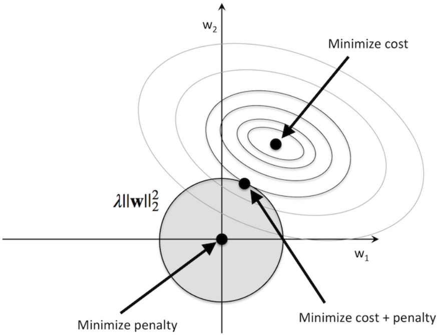
- Equating $\nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbf{0} \implies -\mathbf{\Phi}^\mathsf{T}(\mathbf{t} - \mathbf{\Phi}\mathbf{w}) + \lambda\mathbf{w} = \mathbf{0}$

$$\mathbf{w}_* = (\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^\mathsf{T}\mathbf{t}$$

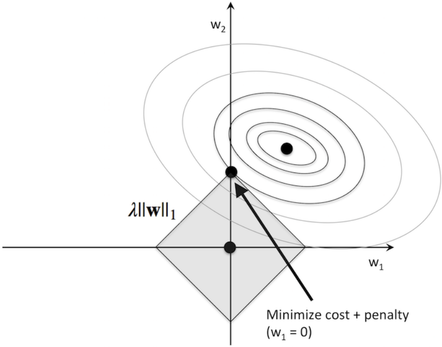
- Regularization term conditions the autocorrelation matrix!

Modified Error Surface

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \mathbf{w}^\mathsf{T}\phi(\mathbf{x}_n)\right)^2 + \lambda\|\mathbf{w}\|_p$$

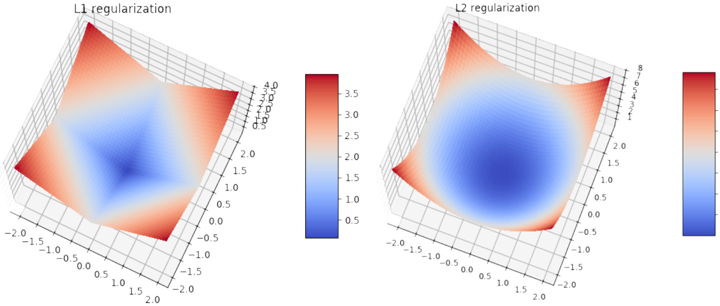


L_2 Regularizer



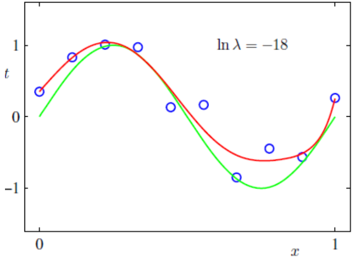
L_1 Regularizer

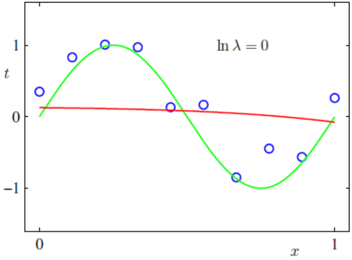
L_1 vs L_2



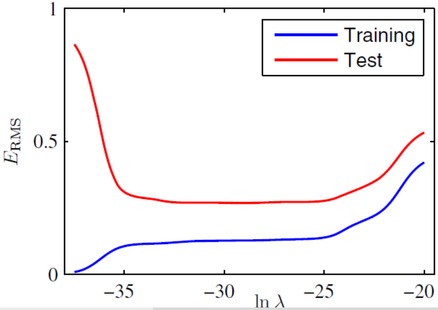
- L_2 regularization - also referred to as ridge regression
  - Promotes smaller and more evenly distributed weights
  - Equivalent to imposing Gaussian priors on parameters $\mathbf{w}$
- L_1 regularization, also referred to as LASSO
  - Promotes sparse models, which improves interpretability
  - Equivalent to imposing Laplacian priors on parameters $\mathbf{w}$

Effect of Regularization ($N = 10$, $M = 9$)

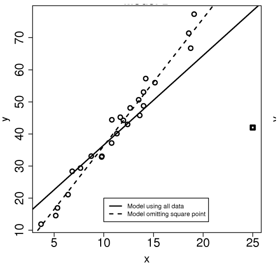




	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01



Issues with Least Squares Error



- Least squares error is highly sensitive to outliers
  - Squaring of errors amplifies the impact of large deviations
- LS assumes that errors are normally distributed with constant variance
- Computational inefficiency and potential numerical instability
  - For large datasets, pseudo-inverse can be computationally expensive
  - If features are highly correlated, then $\mathbf{X}^T\mathbf{X}$ becomes ill-conditioned

Regression - Loss Functions

- Least squares

$$J(\mathbf{w}) = \sum_{n=1}^{N} (t_n - y_n)^2$$

- Least absolute deviations

$$J(\mathbf{w}) = \sum_{n=1}^{N} |t_n - y_n|$$

- Huber loss

$$J(\mathbf{w}) = \sum_{n=1}^{N} \begin{cases} \frac{1}{2}(t_n - y_n)^2 & \text{if } |t_n - y_n| \le \delta, \\ \delta|t_n - y_n| - \frac{1}{2}\delta^2 & \text{if } |t_n - y_n| > \delta. \end{cases}$$

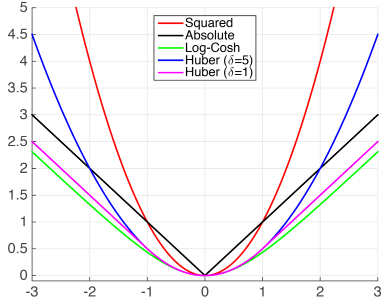
- Log-Cosh loss

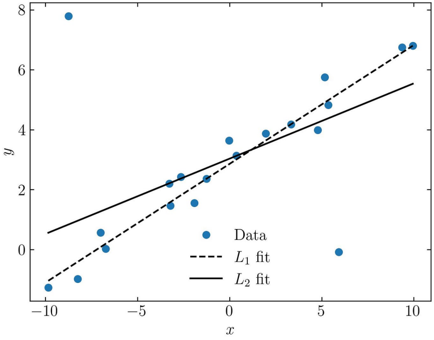
$$J(\mathbf{w}) = \sum_{n=1}^{N} \log(\cosh(t_n - y_n))$$

- KL Divergence

$$J(\mathbf{w}) = \sum_{n=1}^{N} t_n \log \frac{t_n}{y_n}$$

Illustration of Loss Functions



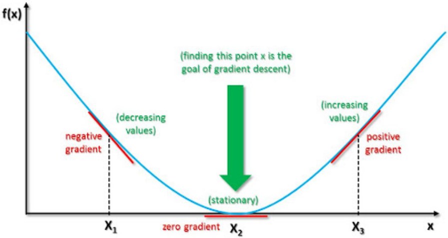


- LAD (L_1) is robust to outliers, but it is not differentiable around zero
- Log-Cosh and Huber follow L_2 at lower error and L_1 at higher errors
- For losses other than LS, we cannot get closed-form solutions

Sequential Learning

- The loss functions may not offer a closed-form solution
- In several applications, we may get data sequentially
- In such cases, we need to search for a solution on the error surface
- Iteratively update $\mathbf{w}^{(\tau+1)}$ by adding a correction factor

$$\mathbf{w}_{\tau+1} = \mathbf{w}_{\tau} + \Delta\mathbf{w}_{\tau}$$



Gradient Descent Algorithm

- Apply correction factor in the negative direction of gradient of $J(\mathbf{w}_\tau)$

$$\mathbf{w}_{\tau+1} = \mathbf{w}_\tau - \eta \nabla J(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_\tau}$$

- η is the step-size parameter - controls the rate of convergence
- For least squares loss, the weight updates are given by

$$J(\mathbf{w}) = \frac{1}{2}\sum_n \left(t_n - \mathbf{w}^\mathsf{T}\phi(\mathbf{x}_n)\right)^2 \quad \nabla J(\mathbf{w}) = -\sum_n \left(t_n - \mathbf{w}^\mathsf{T}\phi(\mathbf{x}_n)\right)\phi(\mathbf{x}_n)$$

$$\mathbf{w}_{\tau+1} = \mathbf{w}_\tau + \eta \sum_n \left(t_n - \mathbf{w}_\tau^\mathsf{T}\phi(\mathbf{x}_n)\right)\phi(\mathbf{x}_n) \quad \tau = 0, 1, \cdots$$

- For least absolute deviations, the weight updates are

$$J(\mathbf{w}) = \sum_n \left|t_n - \mathbf{w}^\mathsf{T}\phi(\mathbf{x}_n)\right| \quad \nabla J(\mathbf{w}) = -\sum_n \text{sign}\left(t_n - \mathbf{w}^\mathsf{T}\phi(\mathbf{x}_n)\right)\phi(\mathbf{x}_n)$$

$$\mathbf{w}_{\tau+1} = \mathbf{w}_\tau + \eta \sum_n \text{sign}\left(t_n - \mathbf{w}^\mathsf{T}\phi(\mathbf{x}_n)\right)\phi(\mathbf{x}_n)$$

Effect of Step-Size η

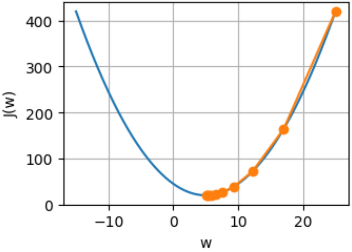
$$J(w) = (w-5)^2 + 20 \quad \nabla J(w) = 2(w-5)$$

$$w_{\tau+1} = w_\tau - 2\eta(w_\tau - 5)$$

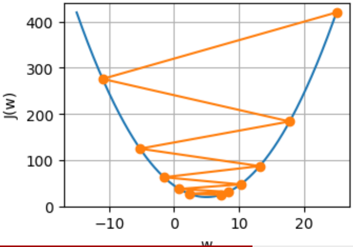
$$w_0 = 25 \quad \& \quad \eta = 0.2$$

Evaluate w_τ for $\tau = 1, 2, 3$

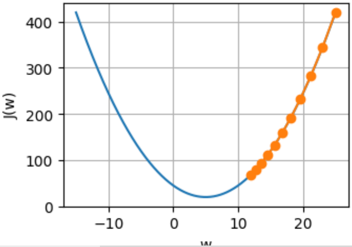
$w_0 = 25 \quad \eta = 0.2$



$\eta = 0.9$



$\eta = 0.05$



Steepest Descent

- Full dataset is used to compute the gradient of the loss function
- The *true gradient* for the full data set is computed as

$$\nabla J(\mathbf{w}) = \sum_{n=1}^{N} \nabla J_n(\mathbf{w})$$

  where $J_n(w)$ denotes loss on the n^{th} data point
- Error surface is consistent across the iterations
- Steepest descent converges smoothly for convex error surfaces

$$\lim_{\tau \to \infty} \mathbf{w}_\tau \to (\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi})^{-1}\mathbf{\Phi}^\mathsf{T}\mathbf{t} \qquad \text{for LS}$$

- Steepest descent gets stuck in local minima on nonconvex surfaces
- Gradient computation for all data at each iteration is expensive

Stochastic Gradient Descent (SGD)

- A random batch of points is used to compute the gradient
- Error surface slightly differs across iterations depending on batch
- *Noisy gradient* for a batch of points ($\mathcal{B}$) is computed as

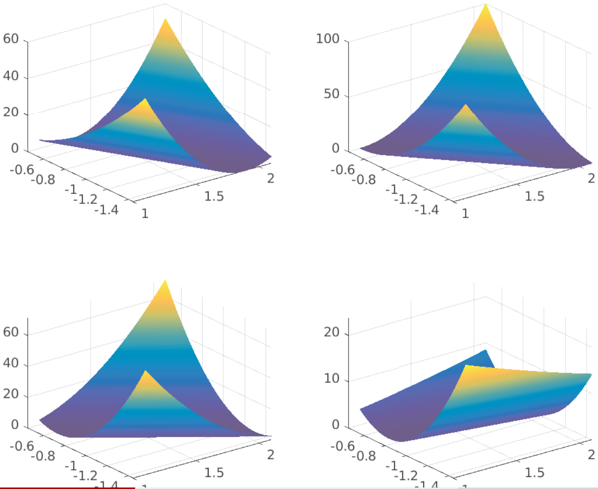
$$\nabla J(\mathbf{w}) = \sum_{n \in \mathcal{B}} \nabla J_n(\mathbf{w})$$

  where $|\mathcal{B}|$ is referred to as size of the batch. ($|\mathcal{B}| = 1$: LMS)

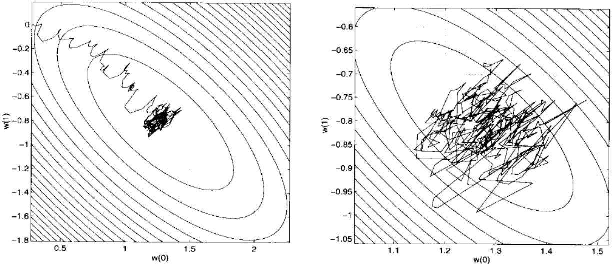
- The noisy gradients can help escape local minima and saddle points
- Suitable for very large datasets as full data is not processed at once
- Loss reduces quickly in the initial iterations but oscillates near minima

SGD Error Dynamics

$$t_n = w_1 x_{n1} + w_2 x_{n2} + \epsilon \qquad w_1 = 1.6, w_2 = -0.5$$



Convergence of SGD

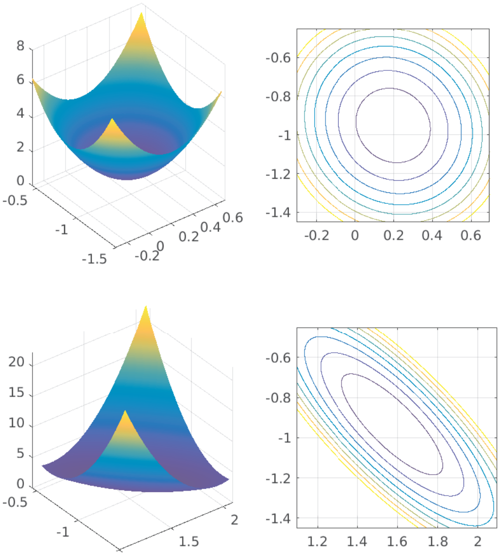


- SGD algorithm converges in mean:

$$\lim_{\tau \to \infty} \mathbb{E}[\mathbf{w}_\tau] \to (\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi})^{-1}\mathbf{\Phi}^\mathsf{T}\mathbf{t} \qquad \eta \text{ is small enough}$$

- Expectation over multiple runs (k) converges to true solution for convex error surfaces, provided η is sufficiently small

Geometry of Error Surface vs Convergence Rate



- Gradient magnitude depends on direction!
- η has to be fixed based on steepest direction.
- Convergence along flatter dimension is too slow!
- Normalization

$$\hat{\phi}_j(\mathbf{x}) = \frac{\phi_j(\mathbf{x}) - \mu_j}{\sigma_j}$$

Newtons Method

- The weights of the model are updated as

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \Delta\mathbf{w}$$

- Expanding the objective function using Taylor series

$$J(\mathbf{w}_{n+1}) = J(\mathbf{w}_n + \Delta\mathbf{w}) = J(\mathbf{w}_n) + \Delta\mathbf{w}^{\mathsf{T}} \nabla J(\mathbf{w}_n) + \frac{1}{2}\Delta\mathbf{w}^{\mathsf{T}} \nabla^2 J(\mathbf{w}_n)\Delta\mathbf{w}$$

- Estimate $\Delta\mathbf{w}$ s.t $J(\mathbf{w}_n + \Delta\mathbf{w})$ is minimized

$$\frac{\partial}{\partial \Delta\mathbf{w}}\left(J(\mathbf{w}_n) + \Delta\mathbf{w}^{\mathsf{T}} \nabla J(\mathbf{w}_n) + \frac{1}{2}\Delta\mathbf{w}^{\mathsf{T}} \nabla^2 J(\mathbf{w}_n)\Delta\mathbf{w}\right) = 0$$

- Optimal update is given by $\Delta\mathbf{w} = -\frac{\nabla J(\mathbf{w}_n)}{\nabla^2 J(\mathbf{w}_n)}$

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mathbf{H}^{-1}(\mathbf{w}_n)\nabla J(\mathbf{w}_n) \qquad \mathbf{H}(\mathbf{w}_n) = \nabla^2 J(\mathbf{w}_n)$$

Homework - 1

- Apply Newtons method to steepest-descent algorithm to the optimal step size η, and check how many iterations are required for convergence.

$$\mathbf{w}^{new} = \mathbf{w}^{old} + \eta\ \mathbf{X}^{\mathsf{T}}(\mathbf{t} - \mathbf{X}\mathbf{w})\Big|_{\mathbf{w}=\mathbf{w}^{old}}$$

Homework - 2

- Suppose you are experimenting with L_1 and L_2 regularization. Further, imagine that you are running gradient descent and at some iteration your weight vector is $w = [1, \epsilon] \in \mathbb{R}^2$ where $\epsilon > 0$ is very small. With the help of this example explain why L_2 norm does not encourage sparsity i.e., it will not try to drive ϵ to 0 to produce a sparse weight vector. Give mathematical explanation.

Homework - 3

- Till now we have been considering a scalar target t from a vector of input observations $\mathbf{x}$. How do you extend this approach for regressing a vector of targets $\mathbf{t} = (t_1, t_2, \cdots t_P)$. Derive the closed form solutions and write sequential update equations using SGD.

Probabilistic Approach to Regression

- Predict target variable(s) $t \in \mathbb{R}$ given the observation vector $\mathbf{x} \in \mathbb{R}^D$
- Target is estimated as a deterministic function $y(\mathbf{x}_n, \mathbf{w})$ with error e_n.

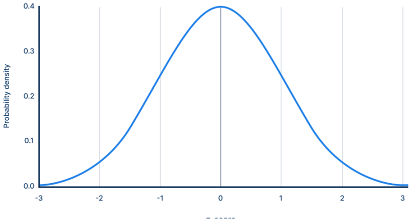
$$t_n = y(\mathbf{x}_n, \mathbf{w}) + e_n$$

- Assuming that the error is Gaussian distributed: $e_n \sim \mathcal{N}(0, \beta^{-1})$
  - The conditional distribution of target t_n is given by

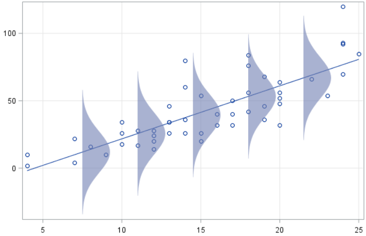
$$p(t_n/\mathbf{x}_n, \mathbf{w}, \beta) \sim \mathcal{N}\left(y(\mathbf{x}_n, \mathbf{w}), \beta^{-1}\right)$$

  - β is referred to as the precision parameter
- We need to estimate $\mathbf{w}$ and β to maximize $p(t_n/\mathbf{x}_n, \mathbf{w}, \beta) \quad \forall n$

Conditional Density of Target



Error Density



Target Conditional

- Gaussian error $e_n \implies$ Gaussian conditional density on targets t_n
- In this case, the mean of the target density is conditioned on input.
  - The observed samples are drawn from this conditional density
  - Given the observations, estimate the conditioning parameters

Maximum Likelihood (ML) Formulation

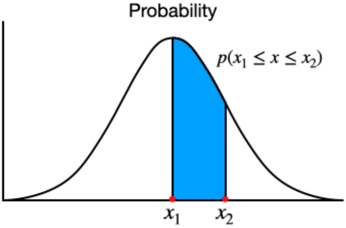
- Training data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2) \cdots (\mathbf{x}_n, t_n) \cdots (\mathbf{x}_N, t_N)\}$
- Let the target be estimated as $\hat{t}_n = \mathbf{y}(\mathbf{x}_n, \mathbf{w}) = \mathbf{w}^{\mathsf{T}} \phi(\mathbf{x}_n)$
- Assuming Gaussian errors: $p(t_n/\mathbf{x}_n, \mathbf{w}, \beta) = \mathcal{N}(\mathbf{w}^{\mathsf{T}} \phi(\mathbf{x}_n), \beta^{-1})$
- Assuming the data points are drawn independently and identically

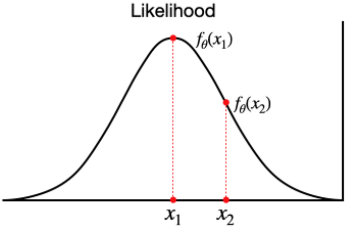
$$p(t_1, t_2, \cdots t_N/\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_N, \mathbf{w}, \beta) = \prod_{n=1}^{N} p(t_n/\mathbf{x}_n, \mathbf{w}, \beta)$$

$$\begin{aligned}
\log p(\mathbf{t}/\mathbf{X}, \mathbf{w}, \beta) &= \sum_{n=1}^{N} \log \mathcal{N}(\mathbf{w}^{\mathsf{T}} \phi(\mathbf{x}_n), \beta^{-1}) \\
&= \frac{N}{2} \log \beta - \frac{N}{2} \log(2\pi) - \frac{\beta}{2} \sum_{n=1}^{N} (t_n - \mathbf{w}^{\mathsf{T}} \phi(x_n))^2
\end{aligned}$$

- $\mathbf{w}$ and β have to be estimated to maximize likelihood $p(\mathbf{t}/\mathbf{X}, \mathbf{w}, \beta)$

Probability vs Likelihood





What is the probability that a sample drawn from a standard Gaussian lies between 1 & 2?

$$P[x_1 \leq X \leq x_2] = \int_{x_1}^{x_2} p_X(x)dx$$

Given that $x = 0.5$ is observed, is it likely to come from $p_{X_1}(x) = \mathcal{N}(0, 1)$ or $p_{X_2}(x) = \mathcal{N}(10, 1)$?

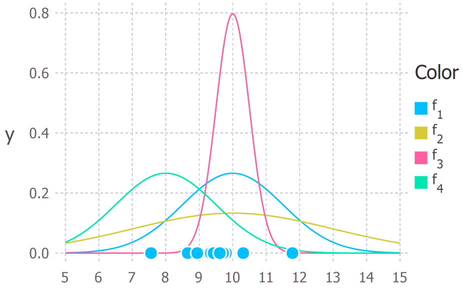
The value of the density function itself is considered the likelihood.

$$\mathcal{L}(\theta_1/x = 0.5) = p(x = 0.5/\theta_1)$$

Understanding Likelihood

- **Probability** of an event happening given the model (parameters)
  - Given the **model**, guess outcome
- **Likelihood** of a model (parameters) given the data
  - Given the **data**, assess how likely different models are.
- Given data D and parameters of a model θ, likelihood of model is

$$\mathcal{L}(\theta/D) = p(D/\theta)$$



ML $\iff$ Least Squares

- ML with Gaussian conditional density assumption is same as LS

$$\mathbf{w}_{ML} = (\mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top \mathbf{t} \qquad \frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n=1}^{N} \Big(t_n - \mathbf{w}_{ML}^\top \phi(\mathbf{x}_n) \Big)^2$$

- ML approach assigns a probability density to the estimated target

$$p(t/\mathbf{x}, \mathbf{w}_{ML}, \beta_{ML}) = \mathcal{N}(y(\mathbf{x}, \mathbf{w}_{ML}), \beta_{ML}^{-1})$$

$$\mathbb{E}[t/\mathbf{x}] = \int t p(t/\mathbf{x}) dt = y(\mathbf{x}, \mathbf{w}_{ML})$$

- ML with Laplacian conditional density assumption is same as LAD
- ML & LS rely on point estimates of model parameters $\mathbf{w}$
- Point estimates cannot be exact with a finite number of samples
- Instead, estimate the distribution of $\mathbf{w}$

Maximum A Posteriori (MAP) Estimate

- Given a set of N datapoints, the posterior distribution of $\mathbf{w}$ is

$$p(\mathbf{w}/\mathbf{t}, \mathbf{X}) \propto p(\mathbf{w})p(\mathbf{t}/\mathbf{w}, \mathbf{X})$$

- Let the prior distribution of $\mathbf{w}$ be Gaussian: $p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$
- Let the conditional distribution of target be Gaussian

$$p(t_n/\mathbf{x}_n, \mathbf{w}, \beta) \sim \mathcal{N}(\mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}_n), \beta^{-1})$$

- The posterior distribution of $\mathbf{w}$ is given by

$$p(\mathbf{w}/\mathbf{t}, \mathbf{X}, \alpha, \beta) \propto \mathcal{N}(\mathbf{w}/\mathbf{0}, \alpha^{-1}\mathbf{I}) \prod_{n=1}^{N} \mathcal{N}(\mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}_n), \beta^{-1})$$

$$\log p(\mathbf{w}/\mathbf{t}) = -\frac{\beta}{2}\sum_{n=1}^{N}\Big(t_n - \mathbf{w}^{\mathsf{T}}\phi(x_n)\Big)^2 - \frac{\alpha}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} + \text{const}$$

- Estimate $\mathbf{w}$ to maximize $\log p(\mathbf{w}/\mathbf{t})$

MAP $\iff$ Regularized Least Squares

- MAP estimation is equivalent to RLS with $\lambda = \frac{\alpha}{\beta}$
- MAP estimate of $\mathbf{w}$ is given by

$$\mathbf{w}_{MAP} = (\mathbf{\Phi}^{\mathsf{T}}\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^{\mathsf{T}}\mathbf{t}$$

- Gaussain priors $\iff$ L_2 regularizer
- Laplacian priors $\iff$ L_1 regularizer

Evaluating Posterior Density $p(\mathbf{w}/\mathbf{t})$

- Let the prior distribution of $\mathbf{w}$ be $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}/\mathbf{m}_0, \mathbf{\Sigma}_0)$
- Assuming linear model with Gaussian errors, the likelihood is given by

$$p(\mathbf{t}/\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n/\mathbf{w}^\mathsf{T}\phi(\mathbf{x}_n), \beta^{-1}) = \mathcal{N}(\mathbf{t}/\mathbf{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I})$$

- The posterior density after observing 'N' samples is given by

$$p(\mathbf{w}/\mathbf{t}) \propto \mathcal{N}(\mathbf{w}/\mathbf{m}_0, \mathbf{\Sigma}_0)\ \mathcal{N}(\mathbf{t}/\mathbf{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I}) = \mathcal{N}(\mathbf{w}/\mathbf{m}_N, \mathbf{\Sigma}_N)$$

- $\mathbf{m}_N$ and $\mathbf{\Sigma}_N$ can be evaluated by completing quadratic term of exp()

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

Bayesian Sequential Estimates

- Let the posterior distribution of $\mathbf{w}$ after observing n samples be

$$p(\mathbf{w}/\mathbf{t}_{1:n}) \sim \mathcal{N}(\mathbf{w}/\mathbf{m}_n, \mathbf{\Sigma}_n)$$

- In sequential update, $p(\mathbf{w}/\mathbf{t}_{1:n})$ is used as prior for $(n+1)^{th}$ sample
- The posterior stats can be updated after observing $(\mathbf{x}_{n+1}, t_{n+1})$ as

$$\mathbf{m}_{n+1} = \mathbf{\Sigma}_{n+1} \left(\mathbf{\Sigma}_n^{-1} \mathbf{m}_n + \beta \phi(\mathbf{x}_{n+1}) t_{n+1} \right)$$
$$\mathbf{\Sigma}_{n+1}^{-1} = \mathbf{\Sigma}_n^{-1} + \beta \phi(\mathbf{x}_{n+1}) \phi^{\mathsf{T}}(\mathbf{x}_{n+1})$$

Bayes Updates Illustration: $t = a_0 + a_1 x + \epsilon$

- Actual targets are generated as

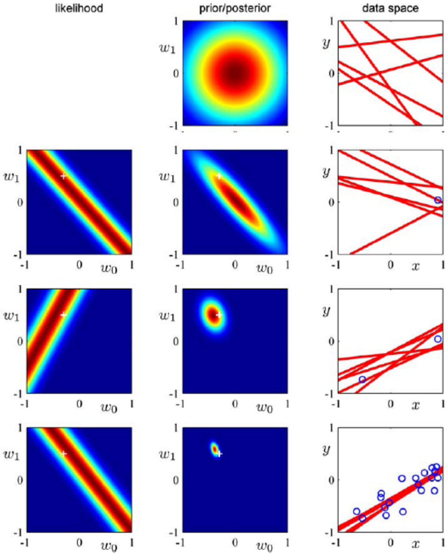
$$t = 0.5x - 0.3 + \epsilon$$

$$x \in \mathcal{U}[-1\ 1] \qquad \epsilon \in \mathcal{N}(0, 0.2^2)$$

- Assume: $y(x, \mathbf{w}) = w_1 x + w_0$
- Assume noise variance is known

$$\beta = \frac{1}{0.2^2} \qquad \alpha = 2.0$$

- Seq. update posterior $p(\mathbf{w}/\mathbf{t})$
- Draw random samples from $p(\mathbf{w}/\mathbf{t})$ and plot y= $w_1 x + w_0$
- Lines converge as data increase



Homework

- Given a Gaussian marginal distribution for $\mathbf{x}$ and a Gaussian conditional distribution for $\mathbf{y}$ in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$$
$$p(\mathbf{y}/\mathbf{x}) = \mathcal{N}(\mathbf{y}/\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1})$$

show that the marginal distribution of $\mathbf{y}$ and conditional distribution of $\mathbf{x}$ are given by

$$p(\mathbf{y}) = \mathcal{N}\left(\mathbf{y}/\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathsf{T}}\right)$$
$$p(\mathbf{x}/\mathbf{y}) = \mathcal{N}\left(\mathbf{x}/\boldsymbol{\Sigma}\left(\mathbf{A}^{\mathsf{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\right), \boldsymbol{\Sigma}\right)$$
$$\text{where } \boldsymbol{\Sigma} = \left(\boldsymbol{\Lambda} + \mathbf{A}^{\mathsf{T}}\mathbf{L}\mathbf{A}\right)^{-1}$$

Predictive Distributions

- Given a training set of N points $(\mathbf{x}_{1:N}, t_{1:N})$, predict target distribution for a new input $\mathbf{x}_0$

$$p(t_0/\mathbf{x}_0, \mathbf{X}, \mathbf{t}, \alpha, \beta) = \int p(t_0, \mathbf{w}/\mathbf{x}_0, \mathbf{X}, \mathbf{t}, \alpha, \beta) d\mathbf{w}$$
$$= \int p(t_0/\mathbf{w}, \mathbf{x}_0, \beta) p(\mathbf{w}/\mathbf{X}, \mathbf{t}, \alpha, \beta) d\mathbf{w}$$

- The predictive distribution is Gaussian and is given by

$$p(t_0/\mathbf{x}_0, \mathbf{X}, \mathbf{t}, \alpha, \beta) = \mathcal{N}\left(t_0/\mathbf{m}_N^T \phi(\mathbf{x}_0), \sigma_N^2(\mathbf{x}_0)\right)$$
$$\sigma_N^2(\mathbf{x}_0) = \frac{1}{\beta} + \boldsymbol{\phi}^{\mathsf{T}}(\mathbf{x}_0) \boldsymbol{\Sigma}_N \phi(\mathbf{x}_0)$$

- What happens to predictive distribution if we have one more data point?

Relation Between σ^2_{N+1} and σ^2_N

- The predictive distribution after observing $N+1$ points is

$$\sigma^2_{N+1}(\mathbf{x}_0) = \frac{1}{\beta} + \phi^{\mathsf{T}}(\mathbf{x}_0)\mathbf{\Sigma}_{N+1}\phi(\mathbf{x}_0)$$

- From Bayesian sequential estimates, $\mathbf{\Sigma}_{N+1}$ is related to $\mathbf{\Sigma}_N$ as

$$\mathbf{\Sigma}^{-1}_{N+1} = \mathbf{\Sigma}^{-1}_N + \beta\phi(\mathbf{x}_{N+1})\phi^{\mathsf{T}}(\mathbf{x}_{N+1})$$

- Woodbury matrix inversion lemma

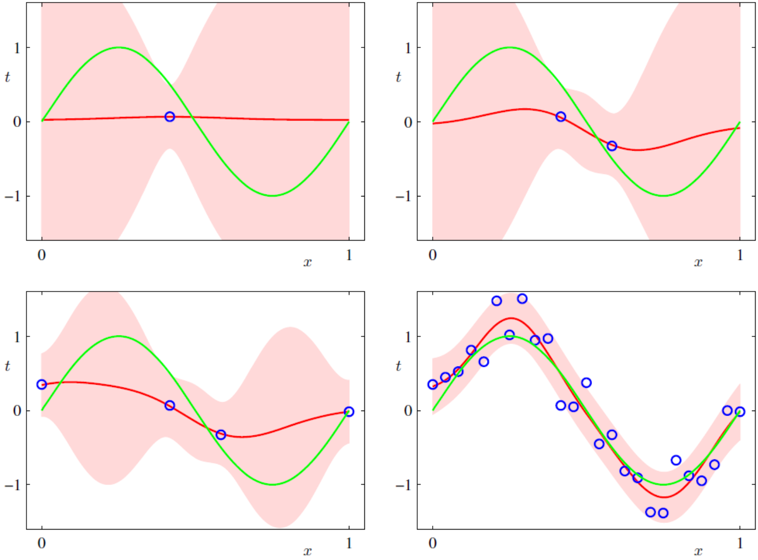
$$\left(\mathbf{M} + \mathbf{v}\mathbf{v}^{\mathsf{T}}\right)^{-1} = \mathbf{M}^{-1} - \frac{\mathbf{M}^{-1}\mathbf{v}\mathbf{v}^{\mathsf{T}}\mathbf{M}^{-1}}{1 + \mathbf{v}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{v}}$$

- Predictive distribution gets narrower with additional training points

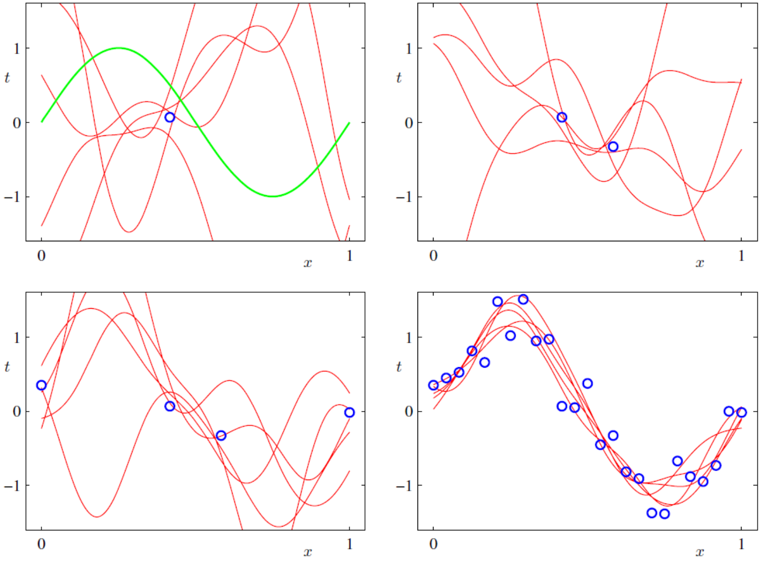
$$\sigma^2_{N+1}(\mathbf{x}_0) \leq \sigma^2_N(\mathbf{x}_0) \qquad \lim_{N\to\infty} \sigma^2_N(\mathbf{x}_0) \to \frac{1}{\beta}$$

- Predictive uncertainty reduces with more data

Predictive Distribution: $t = \sin(2\pi x) + \epsilon$



Curves $y(x, \mathbf{w})$ Sampled from Posterior $p(\mathbf{w}/\mathbf{t})$



Summary of Linear Models of Regression

- Linear in model parameters $\mathbf{w}$.
  - If $\mathbf{x}$ and t are linearly related $\hat{t} = \mathbf{w}^{\mathsf{T}}\mathbf{x}$
  - If relationship is not linear: $\hat{t} = \mathbf{w}^{\mathsf{T}}\phi(\mathbf{x})$
  - LS criterion leads to pseudo-inverse solution: $\mathbf{w}_* = (\mathbf{\Phi}^{\mathsf{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathsf{T}}\mathbf{t}$
  - Regularize $\mathbf{w}$ to avoid over-fitting: $\mathbf{w}_* = (\mathbf{\Phi}^{\mathsf{T}}\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^{\mathsf{T}}\mathbf{t}$
  - Gradient descent algorithms can be used for sequential learning
- Probabilistic interpretation to regression
  - Point estimate of target does not hold for one-to-many maps
  - ML estimation assigns a distribution to the target $p(t_n/y(\mathbf{w}, \mathbf{x}_n), \beta^{-1})$
  - Parameters $\mathbf{w}$ depend on training set - point estimate not enough
  - MAP estimation assigns a distribution to $\mathbf{w}$: $p(\mathbf{w}/\mathbf{t}, \mathbf{X}, \alpha, \beta)$
  - Predict target distribution for a test-point x_0: $p(t_0/x_0, \mathbf{t}, \mathbf{X}, \alpha, \beta)$
  - Predictive uncertainty depends on x_0 and is smallest in the neighborhood of train data points.

Homework

- For Gaussian likelihood and Gaussian posterior, prove that the he predictive distribution is Gaussian and is given by

$$p(t_0/\mathbf{x}_0, \mathbf{X}, \mathbf{t}, \alpha, \beta) = \mathcal{N}\left(t_0/\mathbf{m}_N^T\phi(\mathbf{x}_0), \sigma_N^2(\mathbf{x}_0)\right)$$

$$\sigma_N^2(\mathbf{x}_0) = \frac{1}{\beta} + \boldsymbol{\phi}^\mathsf{T}(\mathbf{x}_0)\boldsymbol{\Sigma}_N\phi(\mathbf{x}_0)$$

- Prove that the predictive uncertainty deceases with increase in training data, i.e., predictive distribution gets narrower with additional training points

$$\sigma_{N+1}^2(\mathbf{x}_0) \leq \sigma_N^2(\mathbf{x}_0)$$

Thank You!