CS 559: Gaussian Process & Supportive Vector Machine

Lecture 6

Outline

- Exam
- Model Selection
- Gaussian Process
- Supportive Vector Machine
- Exam Review

Exam:

- Date: 3/23/2021 11:59 PM – 3/26/2021 11:59 PM
- Time: 1.5 hours
- Format: Online via Canvas Lockdown Browser.
- Question Types: Multiple choice, True/False, Matching, etc…
- Topics: From Lecture 1 to Lecture 6

Empirical Risk Minimization

$$\min_{w} \frac{1}{n} \sum_{i=1}^{N} \underbrace{l(h_w(x_i), y_i)}_{\text{Loss}} + \underbrace{\lambda r(w)}_{\text{Regularizer}} \qquad (1)$$

- The loss function – a continuous function penalizing training error
- The regularizer – a continuous function penalizing classifier complexity

Loss Function: Classifications

Hinge-Loss:

$$\max[1 - h_w(x_i)y_i, 0]^p$$

- Usage: SVM
- When $p = 1$, Standard SVM: the loss function denotes the size of the margin between linear separator and its closest points in either class.
- When $p = 2$, Squared Hingeless SVM: Only differentiable everywhere with $p = 2$.

Log-Loss:

$$\log\left(1 + e^{-h_w(x_i)y_i}\right)$$

- Usage: Logistic Regression
- Its outputs are well-calibrated probabilities.
- Value range from 0 to 1.
- Goal: to minimize its value.

Zero-One Loss:

$$\delta\left(sign\left(h_w(x_i)\right) \neq y_i\right)$$

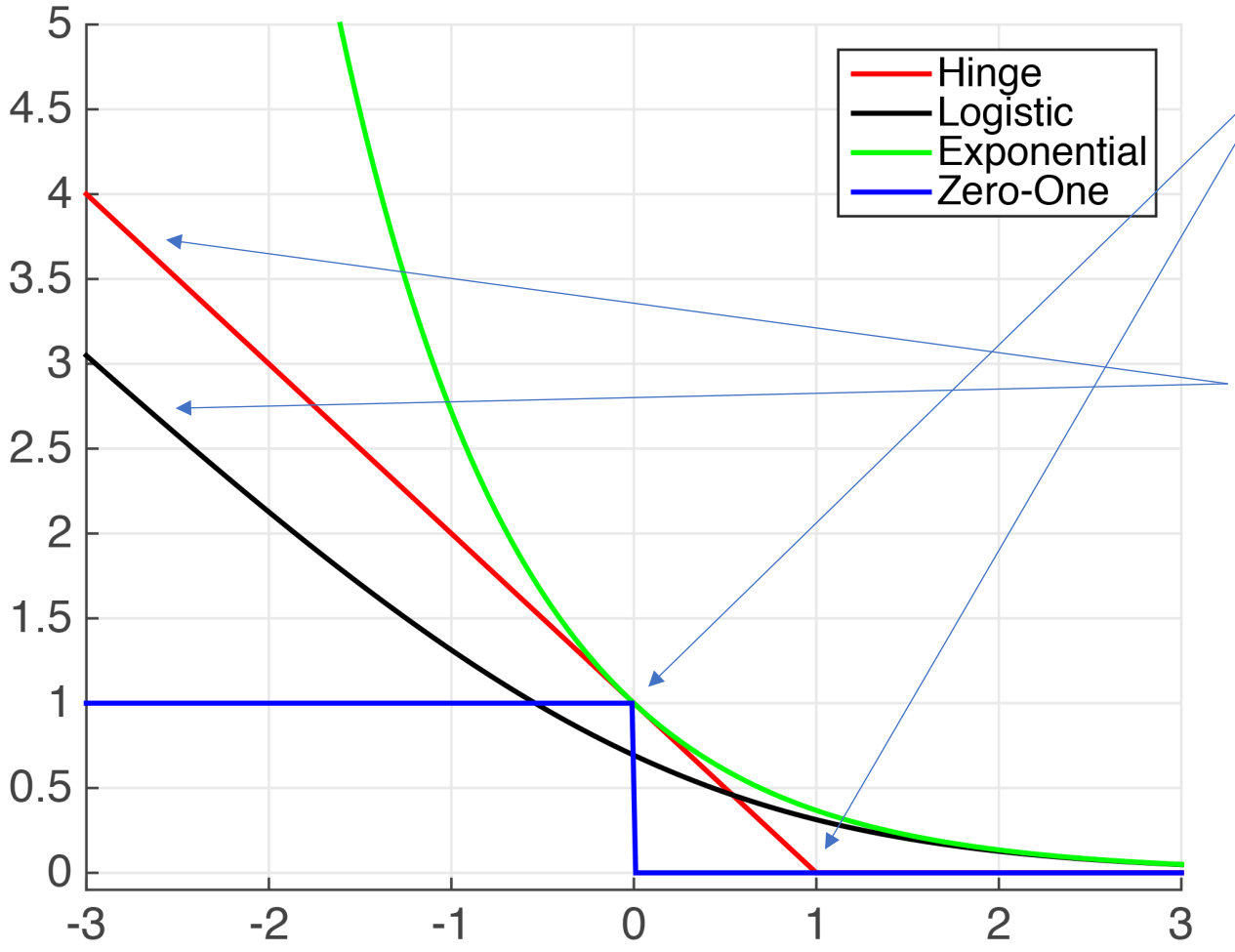
- Usage: Actual Classification Loss
- Characteristic: Not convex nor continuous.

Exponential Loss:

$$\exp(-h_w(x_i)y_i)$$

- Usage: AdaBoost
- Characteristic: Very aggressive
- Misprediction increases exponentially with the value of $-h_w(x_i)y_i$.
- May cause problems with noisy data

Classification Loss Functions Summary



Which functions are strict upper bounds on the 0/1-loss?

What can you say about the hinge-loss and the log-loss as $z \to -\infty$?

Loss Functions: Regressions

Squared Loss:

$$(h(x_i) - y_i)^2$$

- Most popular regression loss function.
- Estimates Mean.
- It is differentiable everywhere.
- Sensitive to outliers.

Absolute Loss:

$$|h(x_i) - y_i|$$

- Another Most popular regression loss function.
- Estimates Median.
- Less sensitive than squared loss function.
- Not differentiable at 0.

Huber Loss:

$$\begin{cases} \frac{1}{2}(h(x_i) - y_i)^2, & if\ |h(x_i) - y_i| < \delta \\ \delta(|h(x_i) - y_i| - \delta/2), & Otherwise \end{cases}$$

- Also known as Smooth Absolute Loss
- "Best of Both Worlds"
- Once-differentiable.
- When the loss is large, use absolute loss.
- When the loss is small, use squared loss.

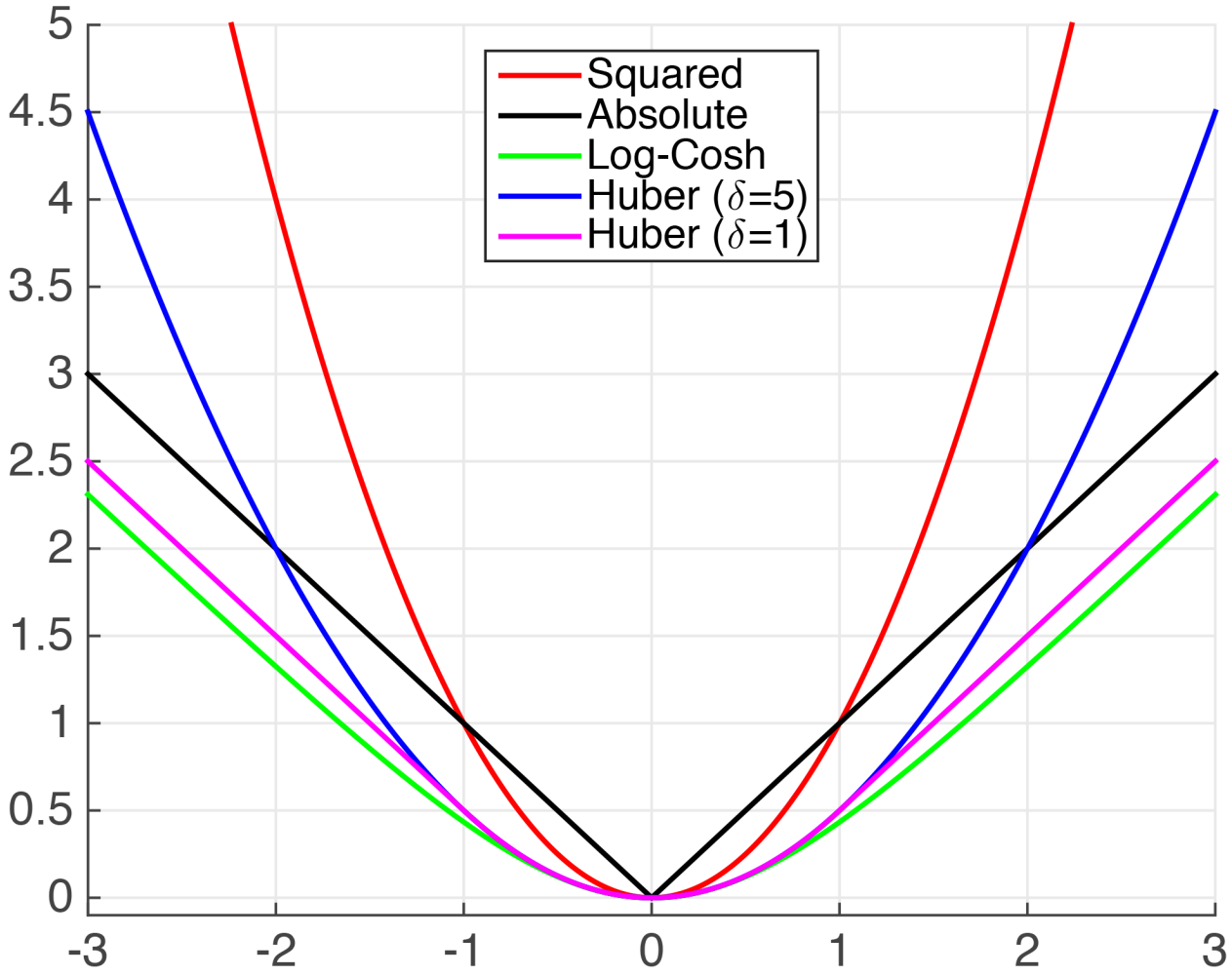
Log-Cosh Loss:

$$\log(\cosh(h(x_i) - y_i))$$

$$\cosh\left(\frac{e^x + e^{-x}}{2}\right)$$

- Similar to Huber Loss.
- Differentiable everywhere.

Regression: Loss Functions Summary



See how sensitive functions are!

Regularizers

We can write Equation (1) into different formation of optimization problem.

$$\min_{w,b} \sum_{i=1}^{N} l(h_w(x_i), y_i) + \lambda r(w) \tag{2}$$

$$\Leftrightarrow \min_{w,b} \sum_{i=1}^{N} l(h_w(x_i), y_i) \;\; s.t. r(w) \leq B$$

Equation (2) indicates that for each $\lambda \geq 0$, the regularizer r(w) is less than or equal to a constant number B

Regularizers

l_2 Regularization:

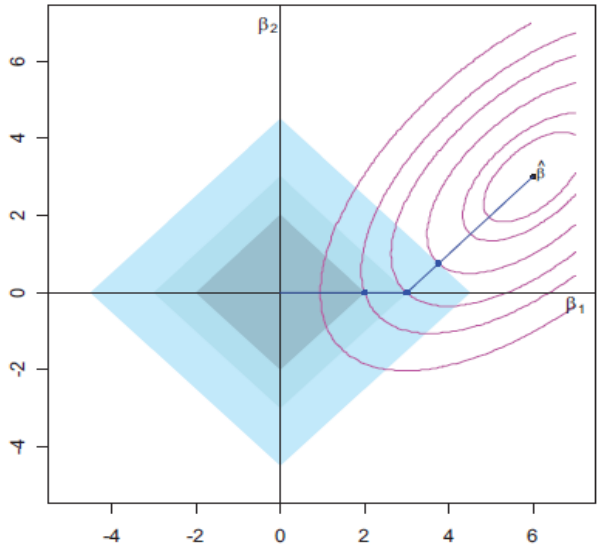
$$r(w) = w^T w = ||w||^2$$

- Strictly convex and differentiable
- Uses weights on all features – dense solutions

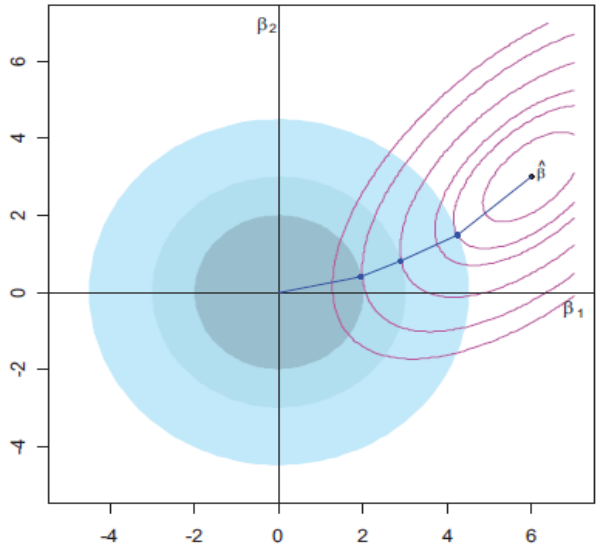
l_1 Regularization:

$$r(w) = ||w||$$

- Convex but not strictly
- Not differentiable at 0
- Sparse Solutions



l_1



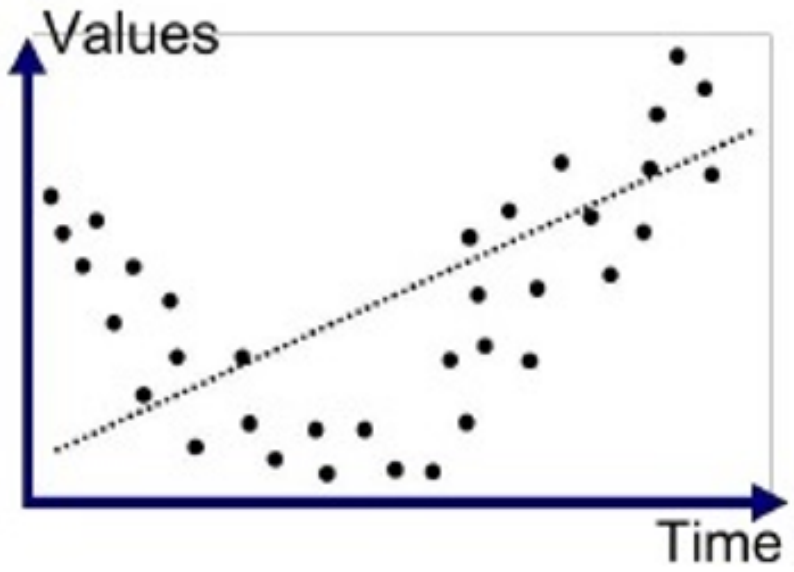
l_2

Elastic Net Regularization

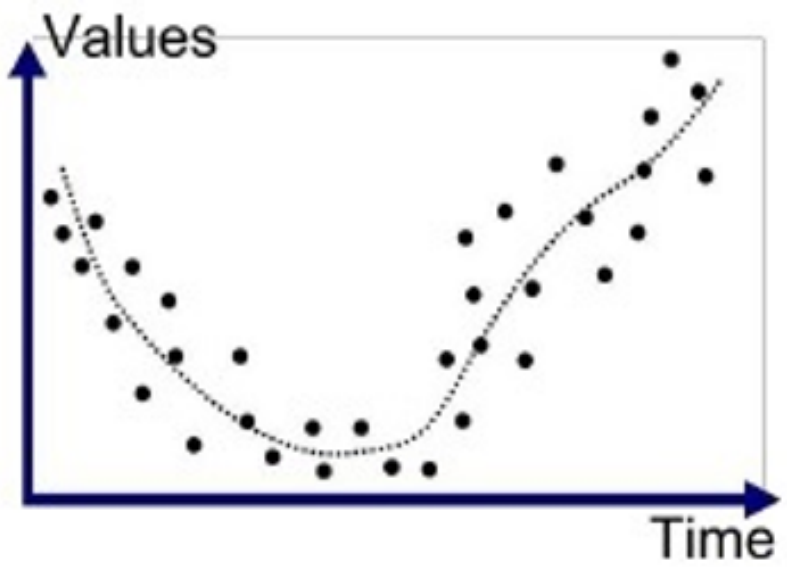
$$\lambda||w|| + (1-\lambda)||w||^2$$

Loss and Regularizer	Comments
Least Squares $\frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2$	• Square Loss • No Regularization • Closed form solution: $w = (XX^T)^{-1}Xy^T$ • $X = [x_1, \dots, x_n], y = [y_1, \dots, y_n]$
Ridge Regression $\min_w \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \lambda\|\|w\|\|^2$	• Squared Loss • l_2 regularization • $w = (XX^T + \lambda I)^{-1}Xy^T$
Lasso Regression $\min_w \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \lambda\|\|w\|\|$	• Sparsity inducing – good for feature selection • Convex but not strict (no unique solution) • Not differentiable at 0 • Solve with sub-gradient descent
Elastic Net $\min_w \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \lambda\|\|w\|\| + (1-\lambda)\|\|w\|\|^2$	• Unique solution and sparsity inducing • Dual of squared-loss SVM • Non-differentiable
Logistic Regression $\min_{w,b} \frac{1}{n}\sum_{i=1}^{n}\log\left(1 - e^{-y_i(w^T x_i + b)}\right)$	• Solve with gradient descent • Probability estimation
Linear SVM $\min_w C\sum_{i=1}^{N}\max[1 - y_i(w^T x_i + b), 0] + \lambda\|\|w\|\|^2$	• Typically l_2 regularized sometimes l_1 • Quadratic program • Kernalize – Solve very efficiently

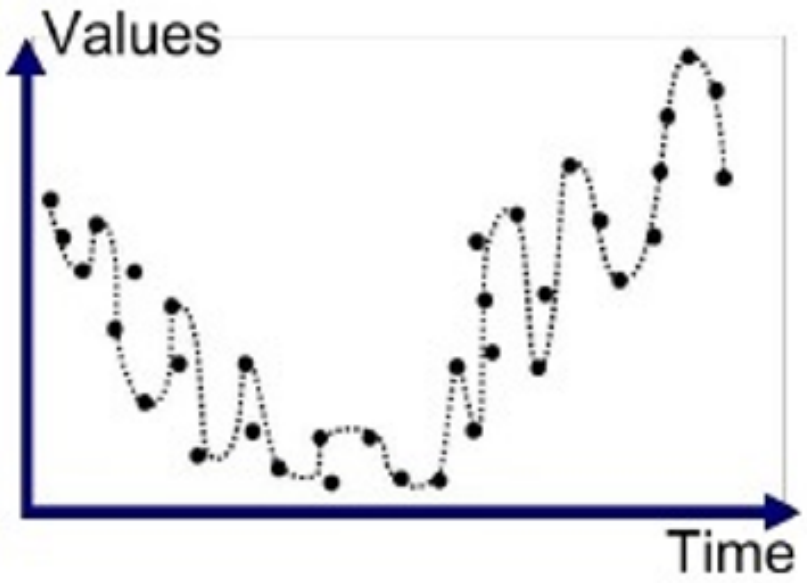
Overfitting vs. Underfitting



Underfitted



Good Fit/Robust



Overfitted

Overfitting vs. Underfitting

$$\min_w \frac{1}{n}\sum_{i=1}^{n} l(h_w(x_i), y_i) + \lambda r(w)$$

Loss Regularizer

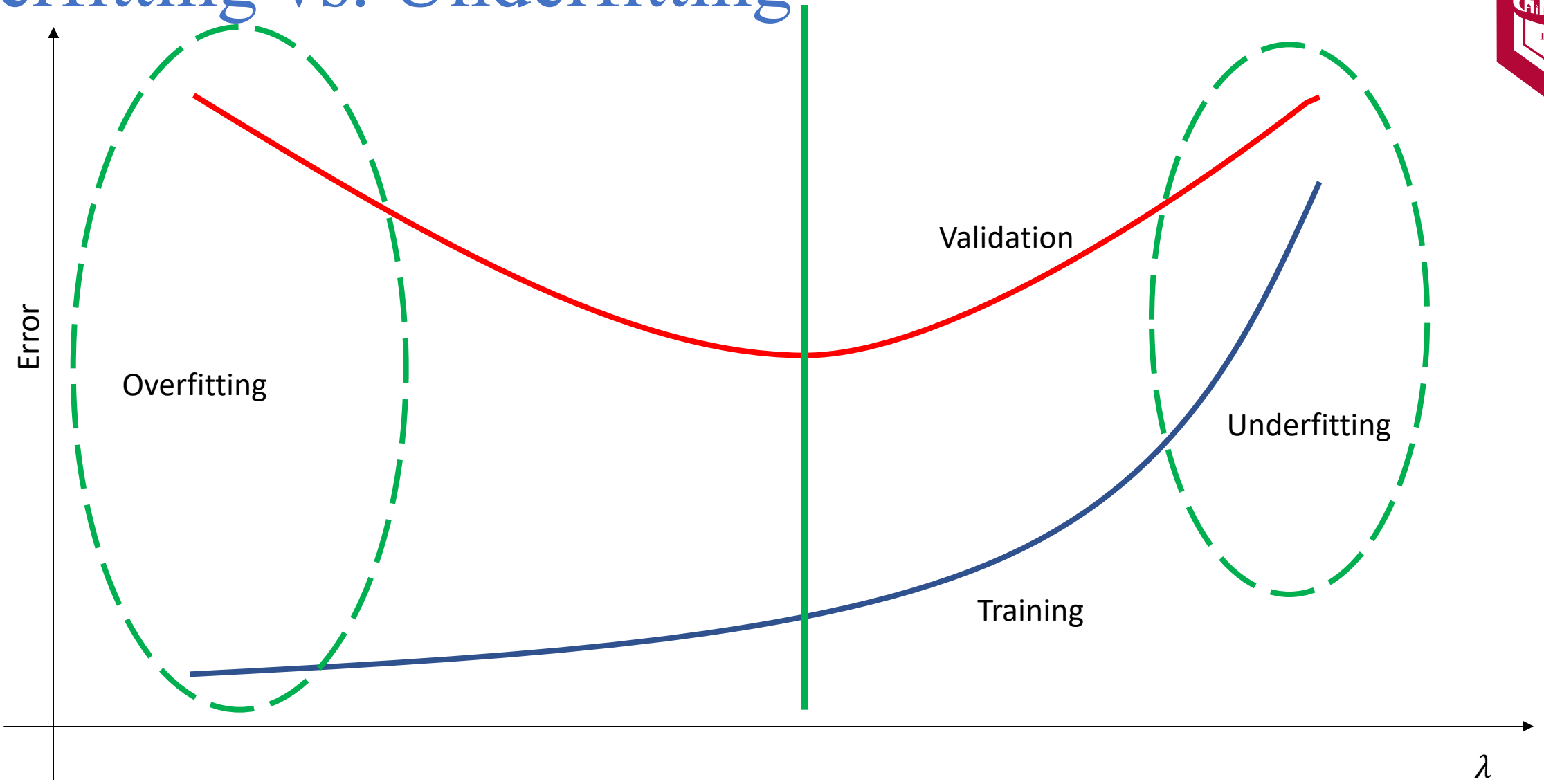
A question rise is on λ – how should we handle it?

Leads to two possible scenarios: **Underfitting** or **Overfitting**

Underfitting – The model does not learn enough through the training process. Both errors on training and test will be high. (The model is relatively simpler than what it supposed to be.)

Overfitting – The model learns too much from training data set. The test error may rise because the model will reflects on the exact patterns of training data set in the test set. (The model is relatively complex than what it should be.)

Overfitting vs. Underfitting



Overfitting vs. Underfitting

The big question is…. **HOW do we find the exact point to stop?**

Typical range: $10^{-5}, 10^{-4}, 10^{-3}, \cdots, 10^{1}, 10^{2}, 10^{3}, \cdots$

K-fold Cross Validation

- Divide the training data k portion equally.
- Train on k-1 of them and leave k^{th} one for the validation.
- Do this k times and find the best λ that gives the smallest error on the validation set.
  - When the data is too small, you leave only one observation for the validation.

For a specific search on λ

- Do with $\lambda = \{\cdots, 10^{-3}, 10^{-2}, \cdots, 10^{1}, 10^{2}, \cdots\}$ to find the best magnitude.
- Then do for the specific value of λ value in the best chosen magnitude
- Example: Let's say you learned that $\lambda = 10^{-2}$ is the best magnitude on the 1^{st} try. Then do with $\lambda = \{0.01, 0.015, 0.02, \cdots\}$ on the 2^{nd} try.

Bias-Variance Tradeoff

$$E_{x,y,D}[(h_D(x)-y)^2] = E_{x,y,D}\left[\left[\left(h_D(x)-\bar{h}(x)\right)+\left(\bar{h}(x)-y\right)\right]^2\right]$$

$$= E_{x,D}\left[\left(h_D(x)-\bar{h}(x)\right)^2\right] + 2E_{x,y,D}\left[\left(h_D(x)-\bar{h}(x)\right)\left(\bar{h}(x)-y\right)\right] + E_{x,y}\left[\left(\bar{h}(x)-y\right)^2\right] \quad (1)$$

2nd Term

$$E_{x,y,D}\left[\left(h_D(x)-\bar{h}(x)\right)\left(\bar{h}(x)-y\right)\right] = E_{x,y}\left[E_D\left[h_D(x)-\bar{h}(x)\right]\left(\bar{h}(x)-y\right)\right]$$
$$= E_{x,y}\left[\left(E_D[h_D(x)]-\bar{h}(x)\right)\left(\bar{h}(x)-y\right)\right]$$
$$= E_{x,y}[0]$$

$$E_D[h_D(x)] = \bar{h}(s)$$

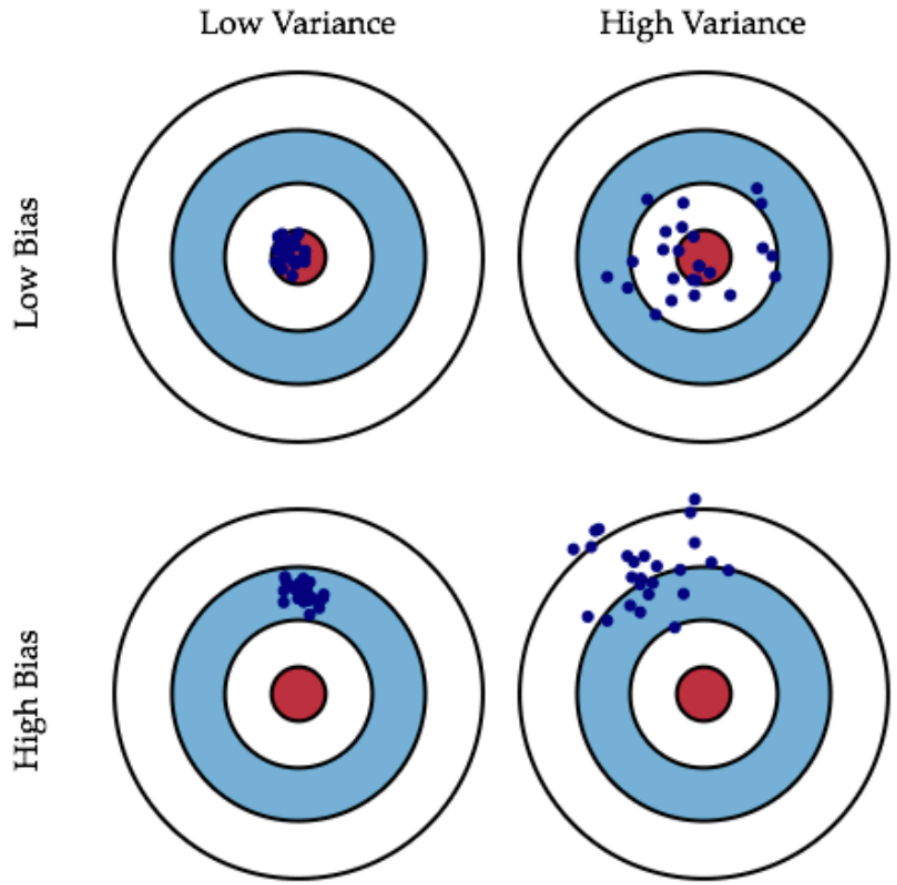
3rd Term

$$E\left[\left(\bar{h}(x)-y\right)^2\right] = E\left[\left(\left(\bar{h}(x)-\bar{y}(x)\right)+\left(\bar{y}(x)-y\right)\right)^2\right]$$
$$= E[(\bar{y}(x)-y)^2] + E\left[\left(\bar{h}(x)-\bar{y}(x)\right)^2\right] + 2E\left[\left(\bar{h}(x)-\bar{y}(x)\right)\left(\bar{y}(x)-y\right)\right]$$

$$E_{x,y,D}[(h_D(x)-y)^2] = E_{x,D}\left[\left(h_D(x)-\bar{h}(x)\right)^2\right] + E_{x,y}\left[\left(\bar{h}(x)-\bar{y}(x)\right)^2\right] + E_x[(\bar{y}(x)-y)^2] \quad (2)$$

$Var(x)$ =Variance Bias² ϵ =Noise

Bias-Variance Tradeoff



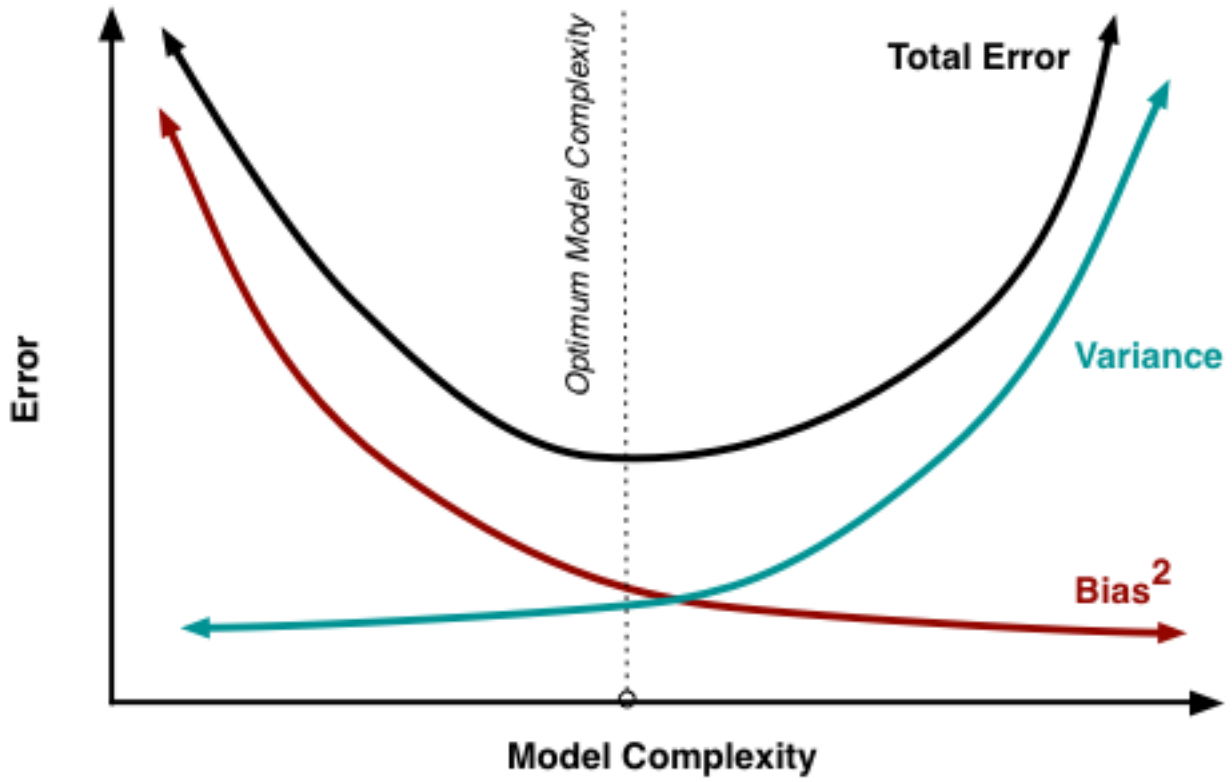


Figure: Graphical illustration of bias vs. variance

Kernel Methods

Kernel Function

- Given: There are a large set of fixed linear basis functions.
- Problem: It is all depends on how complex the basis functions are.
- Solution: Use "dual trick" that is depends on the amount data than the function.

- We have a set of basis functions $\phi(\boldsymbol{x})$ that map inputs x to a feature space.
- This feature space appears in the dot product $\phi(x)^T\phi(x)$ of input pairs, x, x'.
- The kernel function $k(x, x') = \phi(x)^T\phi(x')$ in feature space.
  - We are interested in kernel function not the set of basis functions.

Dual Representations

The linear regression objective is the error.

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}[w^T\phi(x_n) - y_n]^2 + \frac{1}{2}\lambda\mathbf{w}^\mathbf{T}\mathbf{w} \tag{30}$$

We set the gradient to 0.

$$E(\mathbf{w}) = \sum_{n}(\mathbf{w}^T\phi(\boldsymbol{x}_n) - y_n)\phi(\boldsymbol{x}_n) + \lambda\mathbf{w} = 0 \tag{31}$$

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n}(\mathbf{w}^T\phi(\boldsymbol{x}_n) - y_n)\phi(\boldsymbol{x}_n) \tag{32}$$

w is linear combination of inputs in feature space

$$\{\phi(x_n)|1 \leq n \leq N\}$$

Dual Representations

Substitute $\mathbf{w} = \mathbf{\Phi a}$

Where $\Phi = [\phi(x_1), \phi(x_2), \ldots, \phi(x_N)]$

$$a = \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} \text{ and } a_n = -\frac{1}{\lambda}(\mathbf{w}^T \phi(\boldsymbol{x}_n) - y_n)$$

Dual objective: minimize E with respect to a

$$E(a) = \frac{1}{2} a^T \Phi^T \Phi \Phi^T \Phi a - a^T \Phi^T \Phi y + \frac{y^T y}{2} + \frac{\lambda}{2} a^T \Phi^T \Phi a \quad (33)$$

Gram Matrix

The Gram Matrix: $K = \Phi^T \Phi$

Substitution K into (31)

$$E(a) = \frac{1}{2} a^T K K a - a^T K y + \frac{y^T y}{2} + \frac{\lambda}{2} a^T K a \quad (34)$$

$$\nabla E(a) = K K a - K y + \lambda K a$$

$$K y = K(K + \lambda I) a$$

$$a = (K - \lambda I)^{-1} y$$

Prediction $y_* = \phi(x_*)^T w = \phi(x_*)^T \Phi a = k(x_*, X)(K + \lambda I)^{-1} y \quad (35)$

where (X,y) is the training set and (x_*, y_*) is a test instance

- Primal Solution: depends on # of basis functions
- Dual solution: depends on amount of data
  - Advantage: can use very large # of basis functions

    Just need to know k

Constructing Kernels

Possibilities:

- Find mapping Φ to feature space and let $K = \phi^T \phi$
- Directly specify K

A valid kernel must be positive semi-definite:

- k must factor into the product of a transpose matrix by itself
- or all eigenvalues ≥ 0
- problem? It is not always easy to verify.

Rules to construct Kernels

Let $k_1(x, x')$ and $k_2(x, x')$ be valid kernels.

The following kernels are also valid:

$= ck_1(x, x')$ c is the constant > 0

$= f(x)k_1(x, x')f(x')$ f is the function

$= q\big(k_1(x, x')\big)$ q is the polynomial

$= \exp\big(k_1(x, x')\big)$

$k(x, x') = k_1(x, x') + k_2(x, x')$

$= k_3\big(\phi(x), \phi(x')\big)$

$x^T A x'$ A is symmetric positive semi-definite

$= k_a(x_a, x'_a) + k_b(x_b, x'_b)$ k_a & k_b are valid kernel

$= k_a(x_a, x'_a)k_b(x_b, x'_b)$

Other Application? sets, strings, graphs, etc…

Example -

Lodhi, Sunders, Shawe-Tylor, Christianini, Watkins,

Text Classification Using Sting Kernels, JMLR, p. 419-444, 2002

Kernel Construction Example

Consider a 2-D input space $\boldsymbol{x} = (x_1, x_2)$ and suppose we are going to map into a feature space in the degree of 2 using

$$k(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^T \boldsymbol{z})^2$$

$$\begin{aligned} k(x, z) = (x^T z)^2 &= (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= \left(x_1^2, \sqrt{2} x_1 x_2, x_2^2\right)\left(z_1^2, \sqrt{2} z_1 z_2, z_2^2\right) \\ &= \phi(x)^T \phi(z) \end{aligned}$$

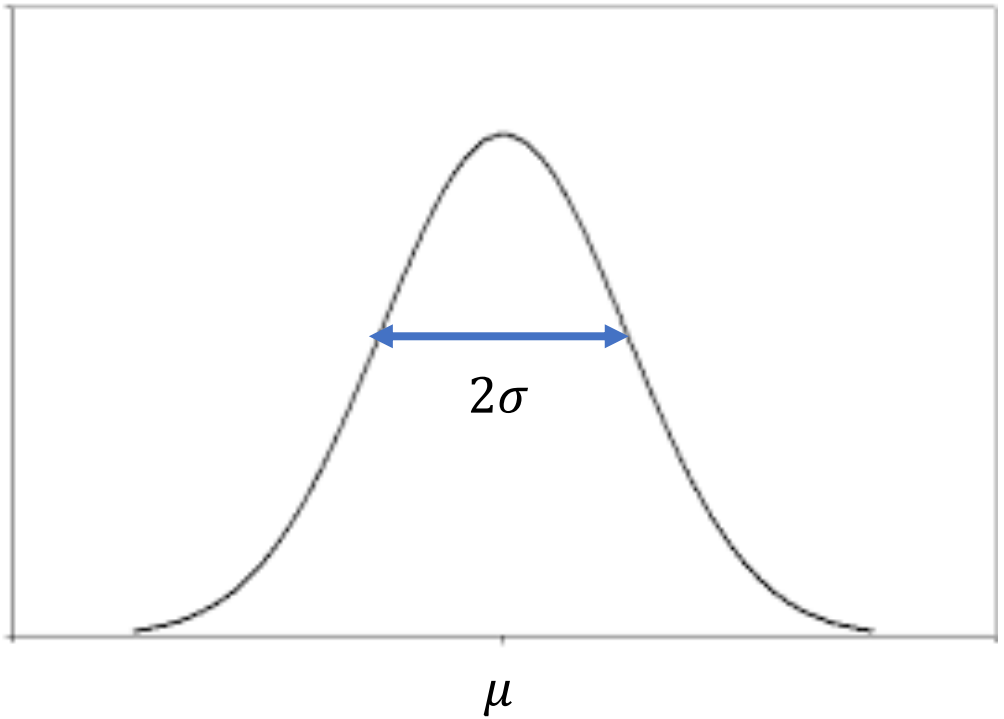
The feature mapping comprises all possible second order terms.

Which kernel functions to use? We can always use the functions whose validities are already confirmed.

Gaussian Processes

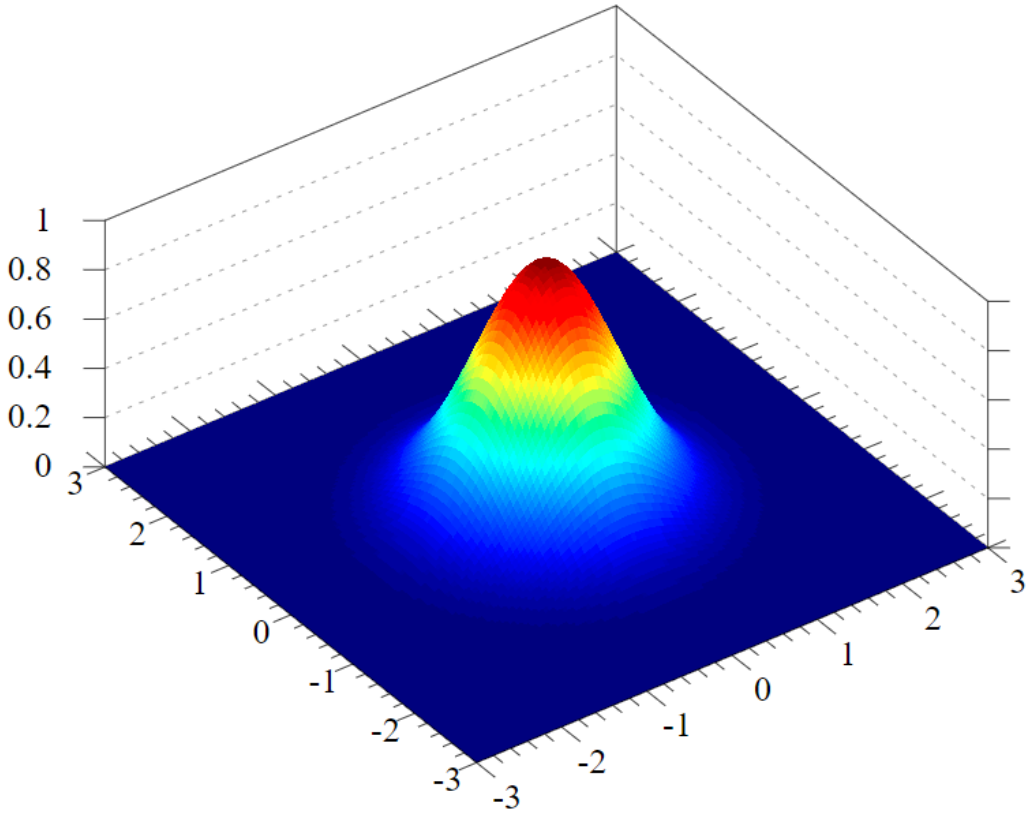
Gaussian Distributions

1-D Gaussian



$$p(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Multivariate Gaussian



$$p(x|\mu,\Sigma) = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}|}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}$$

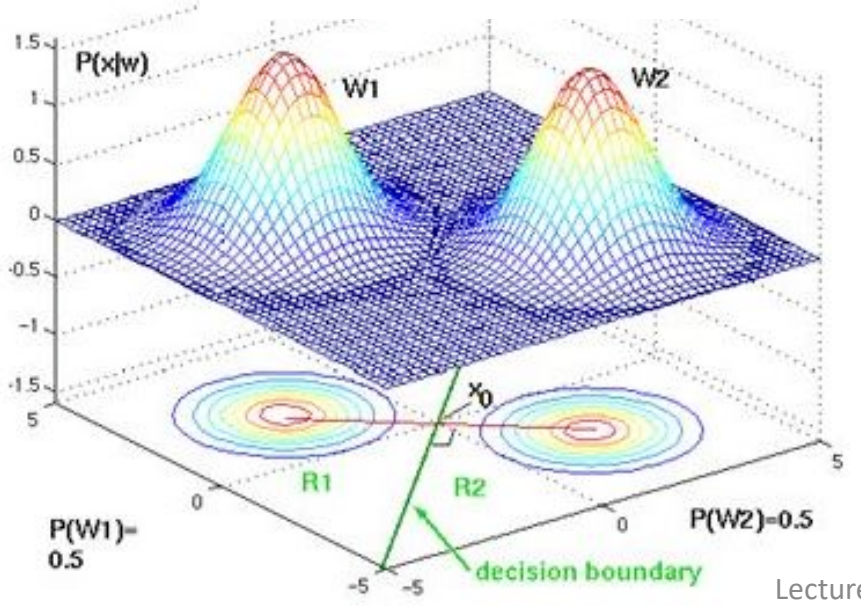
Gaussian Distributions

Multivariate Gaussian Example: A 2-D Gaussian

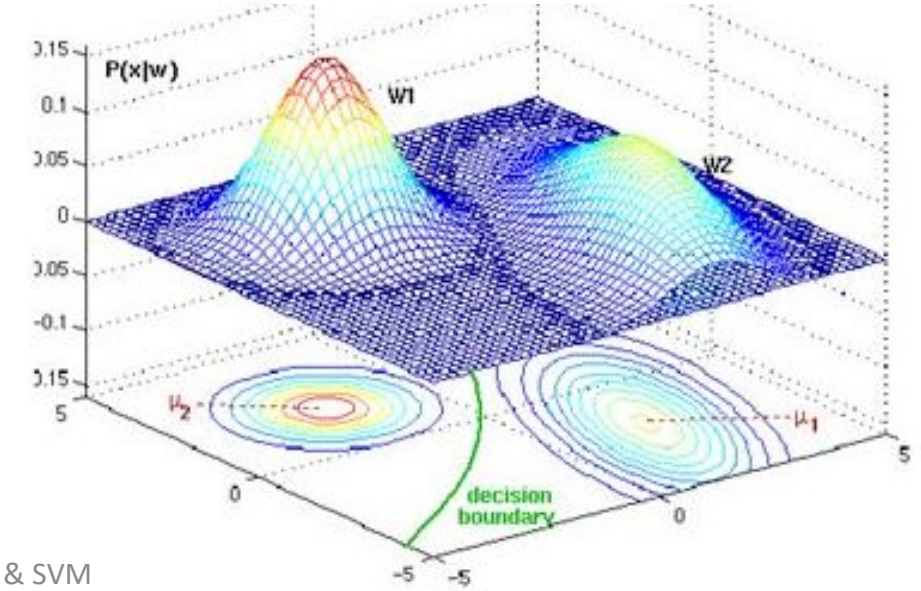
- A mean vector: $\mu = [\mu_1, \mu_2]$
- A covariance matrix: $\Sigma = \begin{bmatrix} \sigma_{11}^2 & \sigma_{21}^2 \\ \sigma_{12}^2 & \sigma_{22}^2 \end{bmatrix}$

where $\sigma_{ij}^2 = E\left[(x_i - \mu_i)(x_j - \mu_j)\right]$ is covariance and Σ is symmetric "positive semi-definite".

$\Sigma = \sigma^2 I$



$\Sigma =$ arbitrary

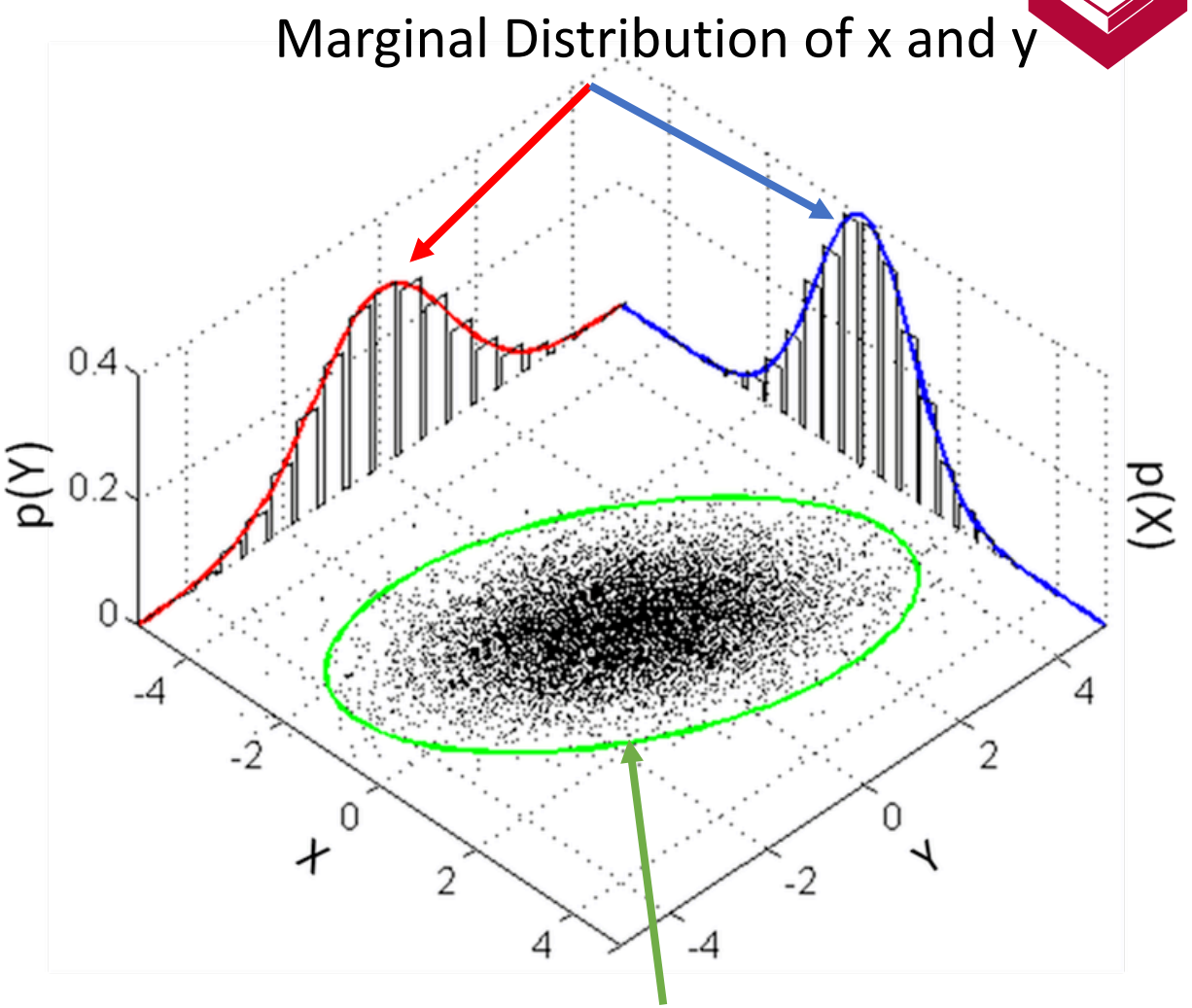


Gaussian Distributions

Marginal Distributions of Gaussians

- Given: $\boldsymbol{x} = (x_a, x_b)$, $\boldsymbol{\mu} = (\mu_a, \mu_b)$,
- $\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$
- Marginal distributions of Gaussians are Gaussian
- Marginal distribution - probability of variable w/o reference to other variables.

$$p(x_a) = \mathcal{N}(x_a | \mu_a, \Sigma_{aa})$$



Gaussian Distributions

Block Matrix Inversion

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} (A - BD^{-1}C)^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix} = \begin{bmatrix} S_D^{-1} & -A^{-1}BS_A^{-1} \\ -D^{-1}CS_D^{-1} & S_A^{-1} \end{bmatrix}$$

Schur complements:

$$S_A = D - CA^{-1}B$$
$$S_D = A - BD^{-1}C$$

Conditional Distribution

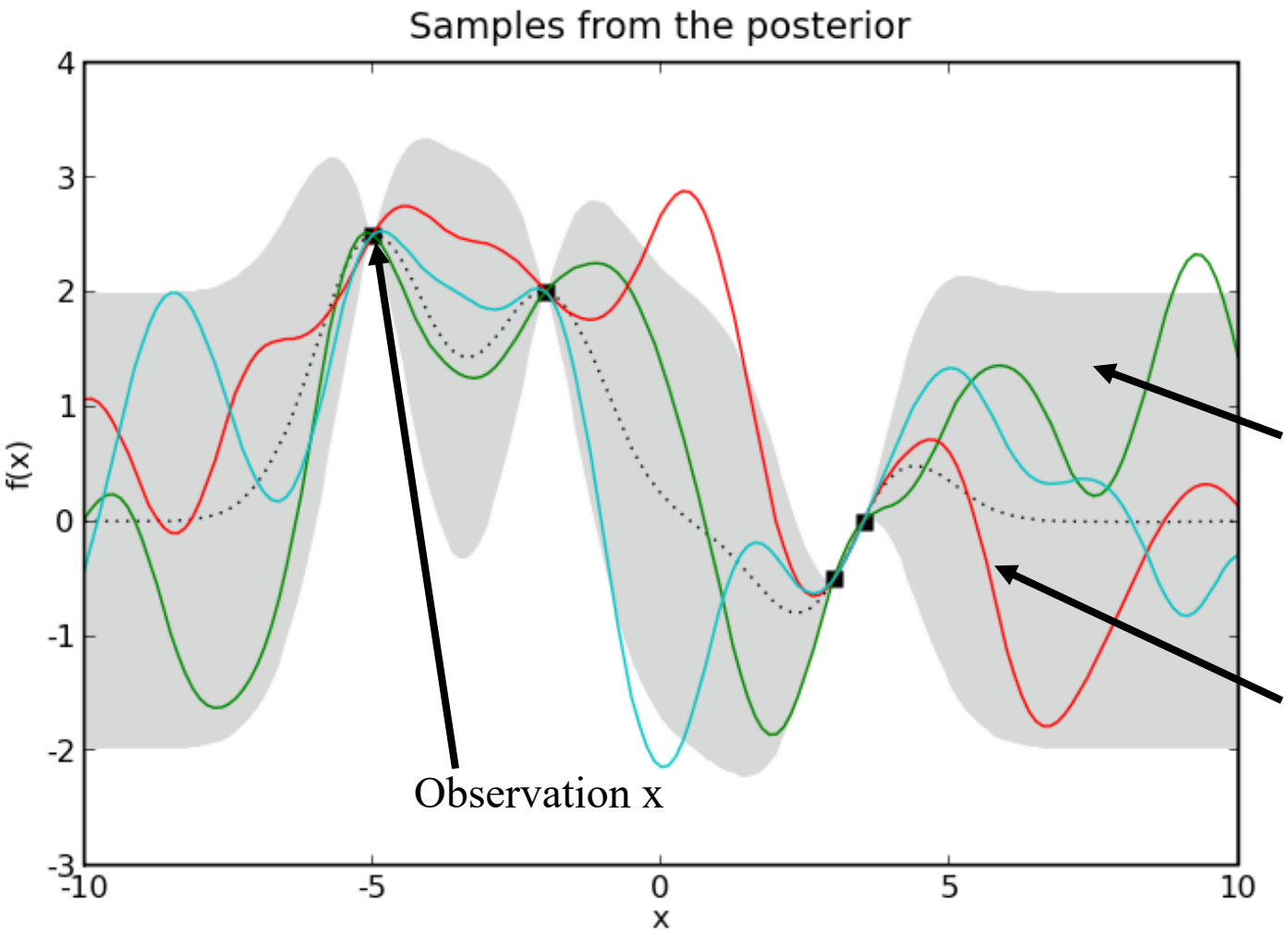
Notation: $\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$ $\Lambda = \Sigma^{-1} = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}$

Distribution: $p(x_a|x_b) = \mathcal{N}\left(x_a \middle| \mu_{(a|b)}, \Lambda_{aa}^{-1}\right)$

where $\mu_{(a|b)} = \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_a) = \mu_a - \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_a)$

and Schur complement $\Lambda_{aa}^{-1} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}$

Regression



Linear Regression:

$$y = w^T x + \epsilon$$

$$y = w^T \phi(x) + \epsilon$$

Linear regression gives a predictive model for one particular parameter w.

How can we do regression?

- We would like a method that could provide confidence during the estimation.
- How about kernelize linear regression?

Gaussian Process

Definition:

- Probability distribution indexed by an arbitrary set.
- Each element gets a Gaussian distribution over the reals with mean μ
- Distributions are dependent/correlated as defined by k(x,z)
- Any finite subset of indices defines a multivariate Gaussian distribution
- $f \sim GP(\mu, k)$

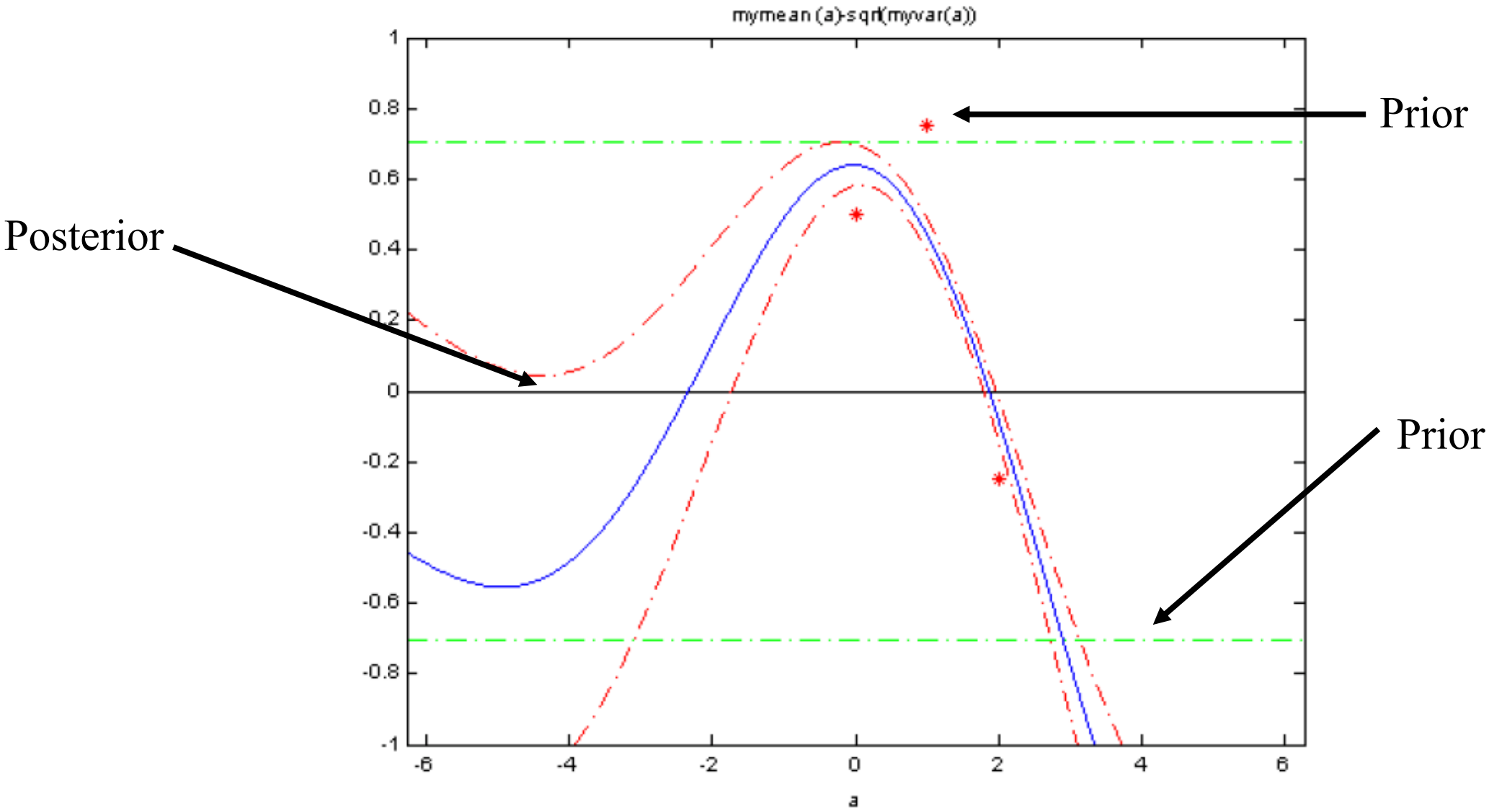
What we get…

- Distribution over functions and functions for any domain index sets.

General Procedure:

- Start with GP prior
- Get some data
- Compute a posterior

Gaussian Process



GP – Weight Space View

GP = Bayesian ridge regression in feature space + kernel trick to carry out computations

Training Set: $D = \{(x_i, y_i) | i = 1, \dots, n\}$

- x_i is the input vector of D-dimension
- y_i is a scalar output or target

Training Data: X and Y

Design matrix $X = [x_1 | \cdots | x_n] \in \mathbb{R}^{D \times n}$

Target vector: $Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n$

$f(x) = x^T w \in \mathbb{R}, x, w \in \mathbb{R}^D$

$y = f(x) + \epsilon = x^T w + \epsilon \in \mathbb{R}$ where $\epsilon \sim \mathcal{N}(0, \sigma^2) \in \mathbb{R}$

GP – Weight Space View LR w/ Gaussian Noi

Calculate the Likelihood and put w prior over parameters w, $w \sim \mathcal{N}(0, \Sigma)$:

$$p(Y|X,w) = \prod_{i=1}^{n} p\left(y_i \middle| x_i^T w\right) \tag{1}$$

$$= \prod_{i=1}^{n} \mathcal{N}\left(x_i^T w, \sigma^2\right) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{\left(y_i - x_i^T w\right)^2}{2\sigma^2}\right]$$

$$= \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \exp\left[-\frac{1}{2\sigma^2}\left|\left|Y - X^T w\right|\right|^2\right]$$

$$= \mathcal{N}(X^T w, \sigma^2 I) \tag{2}$$

GP – Weight Space View LR w/ Gaussian Noi

The prior: $w \sim \mathcal{N}(0, \Sigma)$

The posterior:

$$p(w|X,Y) = \frac{p(Y|X,w)p(w)}{p(Y|X)} = \frac{p(Y|X,w)p(w)}{\int p(Y|X,w)dw} \tag{3}$$

$$= \frac{\mathcal{N}(X^T w, \sigma^2)\mathcal{N}(0,\Sigma)}{\int \mathcal{N}(X^T w, \sigma^2)\mathcal{N}(0,\Sigma)dw}$$

$$\sim \mathcal{N}(X^T w, \sigma^2)\mathcal{N}(0,\Sigma) \tag{4}$$

GP – Weight Space View LR w/ Gaussian Noi

Ridge Regression

$$
\begin{aligned}
p(w|X,Y) &\sim \mathcal{N}(X^T w, \sigma^2)\mathcal{N}(0,\Sigma) \\
&\sim \exp\left[-\frac{1}{2\sigma^2}(Y - X^T w)^T (Y - X^T w)\right] \exp\left[-\frac{1}{2} w^T \Sigma^{-1} w\right] \qquad (5) \\
&\sim \exp[-\frac{1}{2}(w - \bar{w})^T \underbrace{\left(\frac{1}{\sigma^2} XX^T + \Sigma^{-1}\right)}_{=A} (w - \bar{w}) \\
&\sim \mathcal{N}(\bar{w}, A^{-1}) \qquad (6)
\end{aligned}
$$

After "completing the square"

where $\bar{w} = \sigma^{-2}\underbrace{(\sigma^{-2}XX^T + \Sigma^{-1})^{-1}}_{A^{-1} \in \mathbb{R}^{D\times D}} XY \in \mathbb{R}^D$ and $A = (\sigma^{-2}XX^T + \Sigma^{-1}) \in \mathbb{R}^{D\times D}$

GP – Weight Space View LR w/ Gaussian Noi

Use the posterior to predict f in a test point x_*:

$f(x) = x^T w \in \mathbb{R}, x, w \in \mathbb{R}^D$

$y = f(x) + \epsilon = x^T w + \epsilon \in \mathbb{R}$ where $\epsilon \sim \mathcal{N}(0, \sigma^2) \in \mathbb{R}$

$$p(f_*|x_*, X, Y) = \int \underbrace{p(f_*|x_*, w)}_{\delta(f_*, x_*^T, w)} \underbrace{p(w|Y, X)}_{\mathcal{N}(\overline{w}, A^{-1})} dw \quad (7)$$

$$= N(x_*^T \overline{w}, x_*^T A^{-1} x_*) \quad (8)$$

GP – Explicit Feature Space

$$\phi(x) = [x_1, x_1 x_2^2, \dots]^T \in \mathbb{R}^N$$
$$\Phi(X) = [\phi(x_1)|\phi(x_2)|\cdots|\phi(x_n)] \in \mathbb{R}^{N \times n}$$

$\Phi(X)$ is the aggregation of columns $\phi(x)$ for all cases in the training set.

The model is $f(x) = \phi(x)^T w$ and $\phi(x), w \in \mathbb{R}^N$

$$y = f(x) + \epsilon = \phi(x)^T w + \epsilon \in \mathbb{R}$$

GP – Explicit Feature Space

The predictive distribution after feature map:

$$p(f_*|x_*, X, Y) = \mathcal{N}\left(\phi(x_*)^T \bar{w}, \phi(x_*)^T A^{-1} \phi(x_*)\right) \quad (9)$$

where $\bar{w} = \sigma^{-2}(\sigma^{-2}\Phi(X)\Phi(X)^T + \Sigma^{-1})\Phi(X)Y \in \mathbb{R}^N$ and

$A^{-1} \in R^{N\times N}$, $\in R^{N\times n}$, $\in R^{n\times 1}$

$$A = (\sigma^{-2}\Phi(X)\Phi(X)^T + \Sigma^{-1}) \in \mathbb{R}^{N\times N}$$

A problem is we need $N\times N$ matrix inversion…

GP – Explicit Feature Space

We can shorten this equation using

$$\phi_* = \phi(x) \in \mathbb{R}^N, \Phi = \Phi(X) \in \mathbb{R}^{N\times n},$$
$$\bar{w} = \sigma^{-2}(\sigma^{-2}\Phi\Phi^T + \Sigma^{-1})\Phi Y \in \mathbb{R}^N,$$
$$A = (\sigma^{-2}\Phi\Phi^T + \Sigma^{-1}) \in \mathbb{R}^{N\times N}$$

We only need to work with ***n*×*n* matrices**, not $N\times N$!

$$p(f_*|x_*, X, Y) = \mathcal{N}(\phi_*^T\bar{w}, \phi_*^T A^{-1}\phi) \tag{10}$$
$$= \mathcal{N}(\sigma^{-2}\phi_*^T[\sigma^{-2}\Phi\Phi^T + \Sigma^{-1}]^{-1}\Phi Y, \phi_*^T[\sigma^{-2}\Phi\Phi^T + \Sigma^{-1}]^{-1}\phi_*)$$

$\mathbb{R}^{N\times N}$ over $\phi_*^T[\sigma^{-2}\Phi\Phi^T + \Sigma^{-1}]^{-1}\phi_*$

$\mathbb{R}$ over $\sigma^{-2}\phi_*^T[\sigma^{-2}\Phi\Phi^T + \Sigma^{-1}]^{-1}\Phi Y$

Let $K = \Phi(X)^T\Sigma\Phi(X)$,

$$p(f_*|x_*, X, Y)$$
$$= \mathcal{N}\left(\phi_*^T\Sigma\Phi(K + \sigma^2 I)^{-1}Y, \phi_*^T\Sigma\phi_* - \phi_*^T\Sigma\Phi(K + \sigma^2 I)^{-1}\Phi^T\Sigma\phi_*\right) \tag{11}$$

$\mathbb{R}^{1\times n}$ $\mathbb{R}^{n\times n}$ $\mathbb{R}^{n\times 1}$ $\mathbb{R}^{1\times 1}$ $\mathbb{R}^{1\times n}$ $\mathbb{R}^{n\times n}$ $\mathbb{R}^{n\times 1}$

GP – Explicit Feature Space

$$p(f_*|x_*, X, Y) = \mathcal{N}\left(\phi_*^T \Sigma \Phi (K + \sigma^2 I)^{-1} Y, \phi_*^T \Sigma \phi_* - \phi_*^T \Sigma \Phi (K + \sigma^2 I)^{-1} \Phi^T \Sigma \phi_*\right) \quad (11)$$

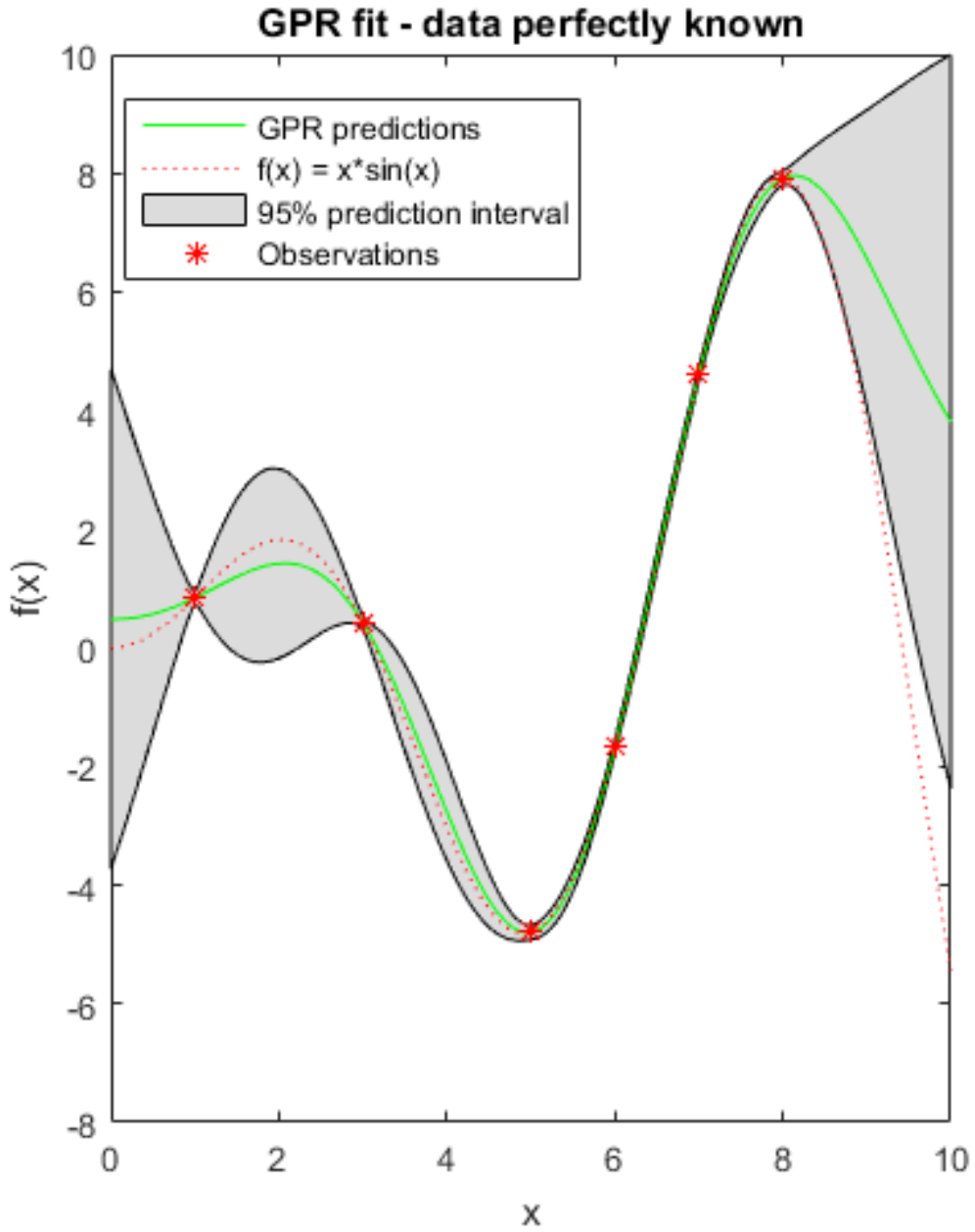
The feature space **always** enters in the form of

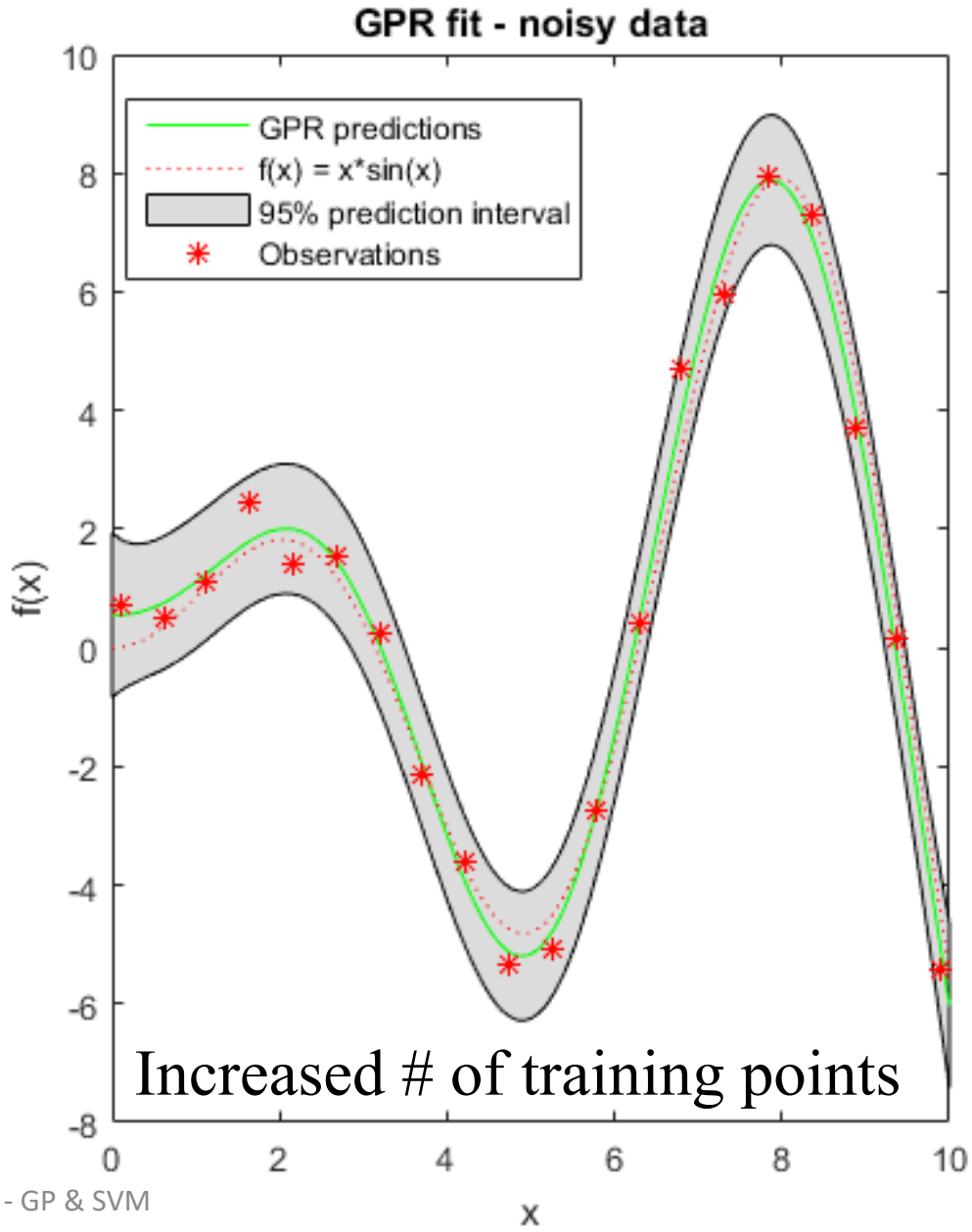
$$\phi_*^T \Sigma \phi_*, \phi_*^T \Sigma \phi, \phi^T \Sigma \phi \in \mathbb{R}^{n \times n}$$

Let form of $k = \phi(x)^T \Sigma \phi(x')$ - a covariance function or kernel.

Inner product in the feature space: $\psi(x) = \Sigma^{1/2} \phi(x)$

GP – Results


GPR fit - data perfectly known
GPR predictions
f(x) = x*sin(x)
95% prediction interval
Observations
f(x)
x


GPR fit - noisy data
GPR predictions
f(x) = x*sin(x)
95% prediction interval
Observations
f(x)
x
Increased # of training points

GP – Function Space View

Definition: GP is a collection of random variables, any finite number of which have a joint Gaussian distribution.

$$m(x) = E[f(x)] \quad \text{(mean function)}$$

$$k(x, x') = E\left[\left(f(x) - m(x)\right)\left(f(x') - m(x')\right)^T\right] \in \mathbb{R} \quad \text{(covariance func.)}$$

GP is completely specified by its mean function $m(x)$ and covariance function $k(x, x')$.

$$f(x) \sim GP\left(m(x), k(x, x')\right) \in \mathbb{R}, x \in \mathbb{R}^D \tag{12}$$

GP – Function Space View

For each $x \in \mathbb{R}^D$ we associate a Gaussian variable $f(x)$ such that $f(x) \in \mathcal{N}\big(m(x), k(x,x)\big)$ and its correlation with other $f(x')$ variables is $k(x,x')$.

$$\mathbb{R} \ni f(x) \sim \mathcal{N}\big(m(x), f(x,x)\big)$$

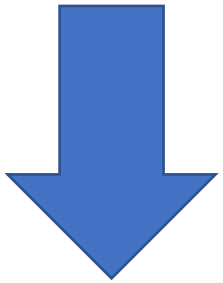
$$\begin{bmatrix} f(x) \\ f(x') \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(x) \\ m(x') \end{bmatrix}, \begin{bmatrix} k(x,x) & k(x',x) \\ k(x,x') & k(x',x') \end{bmatrix}\right) \tag{13}$$

GP – noise free observations

Training set: $D = \{(x_i, f_i) | i = 1, \dots, n\}$

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \in \mathbb{R}^{n \times D}, n \text{ training inputs.} \qquad X_* = \begin{bmatrix} x_{*1}^T \\ \vdots \\ x_{*m}^T \end{bmatrix} \in \mathbb{R}^{m \times D}, m \text{ test inputs}$$

$$f = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \in \mathbb{R}^{n \times D}, n \text{ training inputs} \qquad f_* = \begin{bmatrix} f_{*1} \\ \vdots \\ f_{*m} \end{bmatrix} \in \mathbb{R}^{m \times D}, m \text{ test inputs}$$



$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0_n \\ 0_m \end{bmatrix}, \begin{bmatrix} k(X, X) & k(X, X_*) \\ k(X_*, X) & k(X_*, X_*) \end{bmatrix} \right) \tag{14}$$

GP – noise free observations

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0_n \\ 0_m \end{bmatrix}, \begin{bmatrix} k(X,X) & k(X,X_*) \\ k(X_*,X) & k(X_*,X_*) \end{bmatrix}\right)$$

$$\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} \Lambda = \Sigma^{-1} = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}$$

$$p(x_a|x_b) = \mathcal{N}\left(x_a \middle| \mu_{a|b}, \Lambda_{aa}^{-1}\right)$$

$$\mu_{a|b} = \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_a) = \mu_a - \Sigma_{aa}\Sigma_{bb}^{-1}(x_b - \mu_a)$$

$$\Lambda_{aa}^{-1} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}$$

$$p(f_*|X_*, X, f) = \mathcal{N}\left(k(X_*,X)k(X,X)^{-1}f, k(X_*,X_*) - k(X_*,X)k(X,X)^{-1}k(X,X_*)\right) \quad (15)$$

If $X_* = X \Longrightarrow f_* = f$ and the covariance is 0.

$$p(f_*|X_*, X, f) = \mathcal{N}\left(\phi_*^T\Sigma\Phi(K + \sigma^2 I)^{-1}f, \phi_*^T\Sigma\phi_* - \phi_*^T\Sigma\Phi(K + \sigma^2 I)^{-1}\Phi^{\mathrm{T}}\Sigma\phi_*\right) \quad (16)$$

GP – noise observations

$$y = f(x) + \epsilon \in \mathbb{R}$$
$$\epsilon \sim \mathcal{N}(0, \sigma^2) \in \mathbb{R}$$

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0_n \\ 0_m \end{bmatrix}, \begin{bmatrix} k(X,X) & k(X,X_*) \\ k(X_*,X) & k(X_*,X_*) \end{bmatrix}\right)$$

$$\Rightarrow cov(y_p, y_q) = k(x_p, x_q) + \sigma^2 \delta_{p,q}$$ ← (Dirac Delta Function)

$$\Rightarrow cov([y_1, \dots, y_n]) = k(X,X) + \sigma^2 I \in \mathbb{R}^{n \times n}$$

$$\Rightarrow \begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0_n \\ 0_m \end{bmatrix}, \begin{bmatrix} k(X,X) + \sigma^2 I & k(X,X_*) \\ k(X_*,X) & k(X_*,X_*) \end{bmatrix}\right)$$ ← (New joint distribution)

(17)

The posterior for the noisy observations:

$$p(f_*|X,Y,X_*) = \mathcal{N}\left(\bar{f}_*, cov(f_*)\right)$$ (18)

GP – noise observations

$$\Rightarrow \begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0_n \\ 0_m \end{bmatrix}, \begin{bmatrix} k(X,X)+\sigma^2 I & k(X,X_*) \\ k(X_*,X) & k(X_*,X_*) \end{bmatrix}\right)$$

$$\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} \Lambda = \Sigma^{-1} = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}$$

$$p(x_a|x_b) = \mathcal{N}\left(x_a \middle| \mu_{a|b}, \Lambda_{aa}^{-1}\right)$$

$$\mu_{a|b} = \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_a) = \mu_a - \Sigma_{aa}\Sigma_{bb}^{-1}(x_b - \mu_a)$$

$$\Lambda_{aa}^{-1} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}$$

$$p(f_*|X,Y,X_*) = \mathcal{N}\left(\bar{f}_*, cov(f_*)\right)$$

where

$$\bar{f}_* = E[f_*|X,Y,X_*] = k(X_*,X)[k(X,X)+\sigma^2 I]^{-1}Y \in \mathbb{R}^m \quad (19)$$

$$cov(f_*) = k(X_*,X_*) - k(X_*,X)[k(X,X)+\sigma^2 I]^{-1}K(X,X_*) \in \mathbb{R}^{m\times m} \quad (20)$$

$$\mathcal{N}(\phi_*^T\Sigma\Phi(K+\sigma^2 I)^{-1}Y, \phi_*^T\Sigma\phi_* - \phi_*^T\Sigma\Phi(K+\sigma^2 I)^{-1}\Phi\Sigma\phi_*) \leftarrow \text{(Weight Space View)}$$

If $k = \phi(x)^T\Sigma\phi(x')$

GP – noise observations

Short Notation for a single test point x_*:

$$K = k(X, X) \in \mathbb{R}^{n \times n}$$
$$K_* = k(X, X_*) \in \mathbb{R}^{n \times m}$$

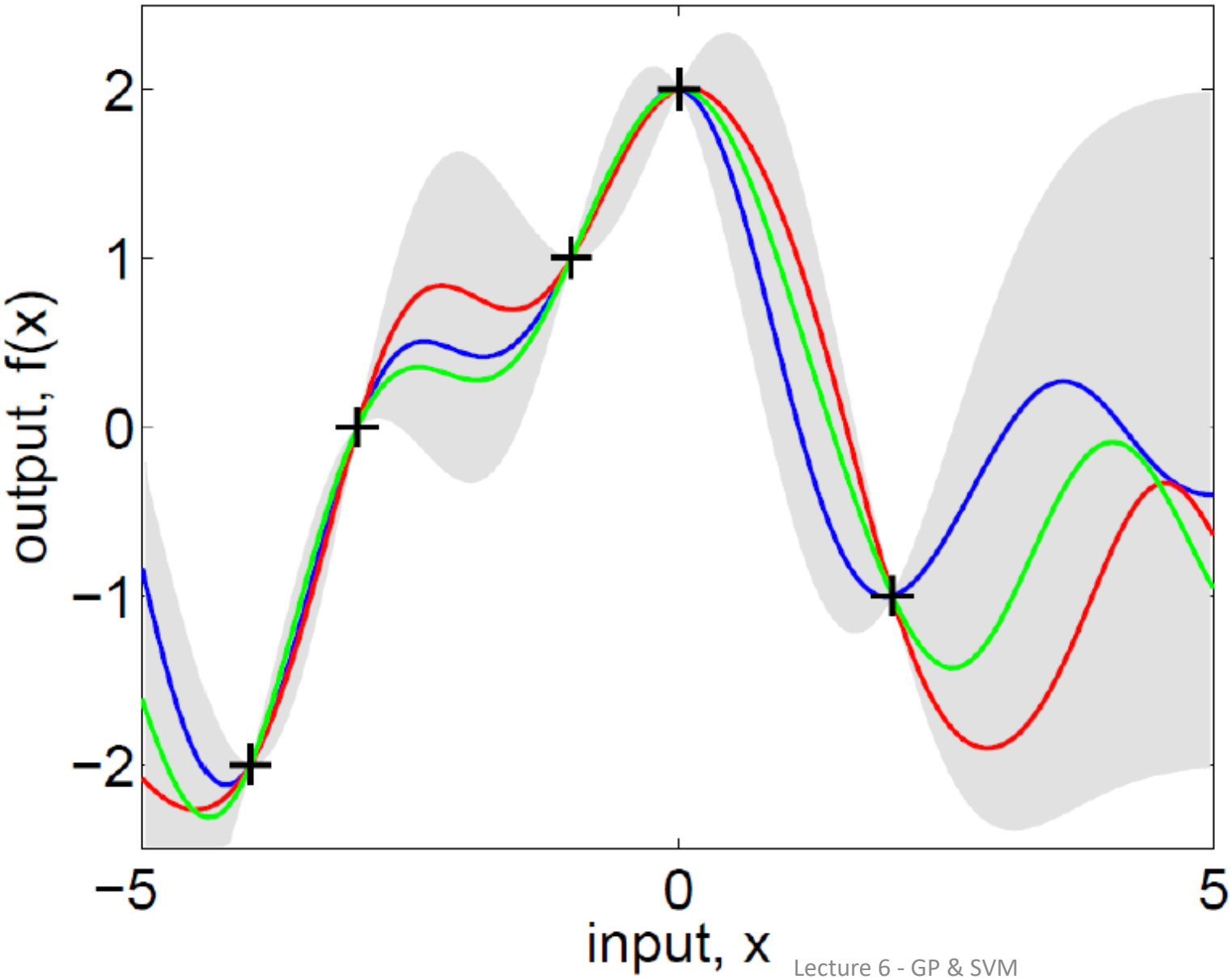
$$k(x_*) = k_* = k(X, x_*) = \begin{bmatrix} k(x_1, x_*) \\ \vdots \\ k(x_n, x_*) \end{bmatrix} \in \mathbb{R}^n$$

$$\bar{f}_* = k_*^T [K + \sigma^2 I]^{-1} Y \in \mathbb{R} \tag{21}$$

$\mathbb{R}^{1 \times n}$ $\mathbb{R}^{n \times n}$ $\mathbb{R}^n$

$$cov(f_*) = k(x_*, x_*) - k_*^T [K + \sigma^2 I]^{-1} k_* \in \mathbb{R} \tag{22}$$

$\mathbb{R}$ $\mathbb{R}^{1 \times n}$ $\mathbb{R}^{n \times n}$ $\mathbb{R}^n$


output, f(x)
input, x
2
1
0
−1
−2
−5
0
5

GP – noise observations

Linear combination of observation Y - linear predictor

Let $\beta^T = k_*^T[K + \sigma^2 I]^{-1} \in \mathbb{R}^{1\times n}$

$$\bar{f}_* = \sum_{i=1}^{N} \beta_i^T y_i$$

$$\bar{f}_* = \underbrace{k_*^T[K + \sigma^2 I]^{-1}}_{\mathbb{R}^{\mathbf{1}\times \boldsymbol{n}}} \underbrace{Y}_{\mathbb{R}^n} \in \mathbb{R}$$

A linear combination of n kernel function - representer theorem

Let $\alpha = [K + \sigma^2 I]^{-1} Y$

$$\bar{f}_* = \sum_{i=1}^{n} \alpha_i^T k(x_i, x_*)$$

$$\bar{f}_* = \underbrace{\{[K + \sigma^2 I]^{-1} Y\}^T}_{\mathbb{R}^{\boldsymbol{n}\times \boldsymbol{n}}} \underbrace{k_*}_{\mathbb{R}^{n\times 1}} \in \mathbb{R}$$

Kernel Construction Example

Consider a 2-D input space $\boldsymbol{x} = (x_1, x_2)$ and suppose we are going to map into a feature space in the degree of 2 using

$$k(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^T \boldsymbol{z})^2$$

$$\begin{aligned} k(x, z) = (x^T z)^2 &= (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= \left(x_1^2, \sqrt{2} x_1 x_2, x_2^2\right)\left(z_1^2, \sqrt{2} z_1 z_2, z_2^2\right) \\ &= \phi(x)^T \phi(z) \end{aligned}$$

The feature mapping comprises all possible second order terms.

Which kernel functions to use? We can always use the functions whose validities are already confirmed.

Supportive Vector Machine

- Maximum Margin Classifiers
- Overlapping class distributions
- Classification
- Regression
- Kernel SVM - Regression

SVM

$$y(x) = w^T \phi(x) + b$$

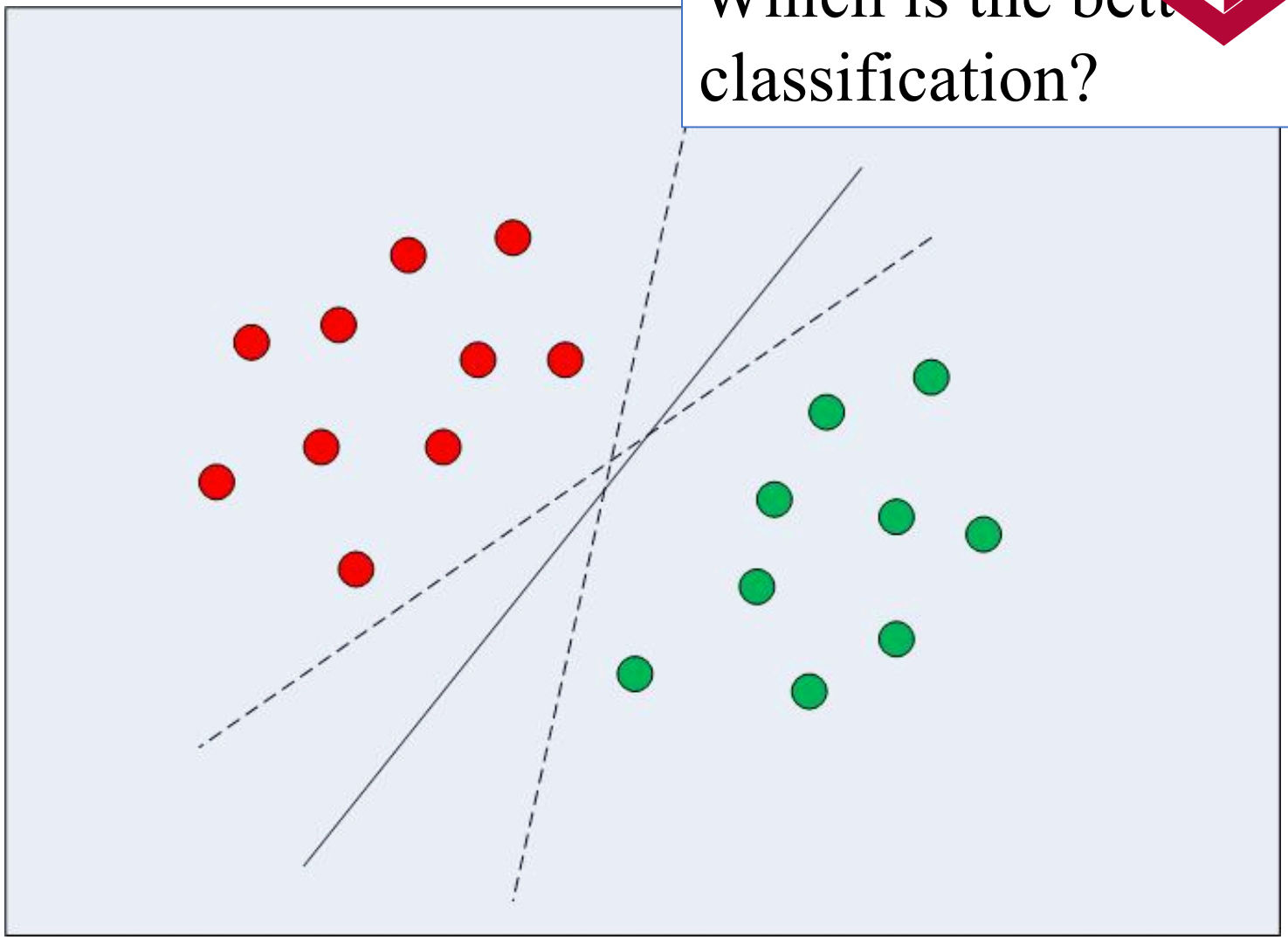
- $\phi(x)$ a fixed feature-space transformation
- b the bias parameter
- training set - linearly separable in feature space
- Data: $\{(x_i, t_i)|n = 1, \ldots, n\}$
- $t_i \in \{-1,1\}$

SVM vs. Perceptron Algorithm

Perceptron Algorithm

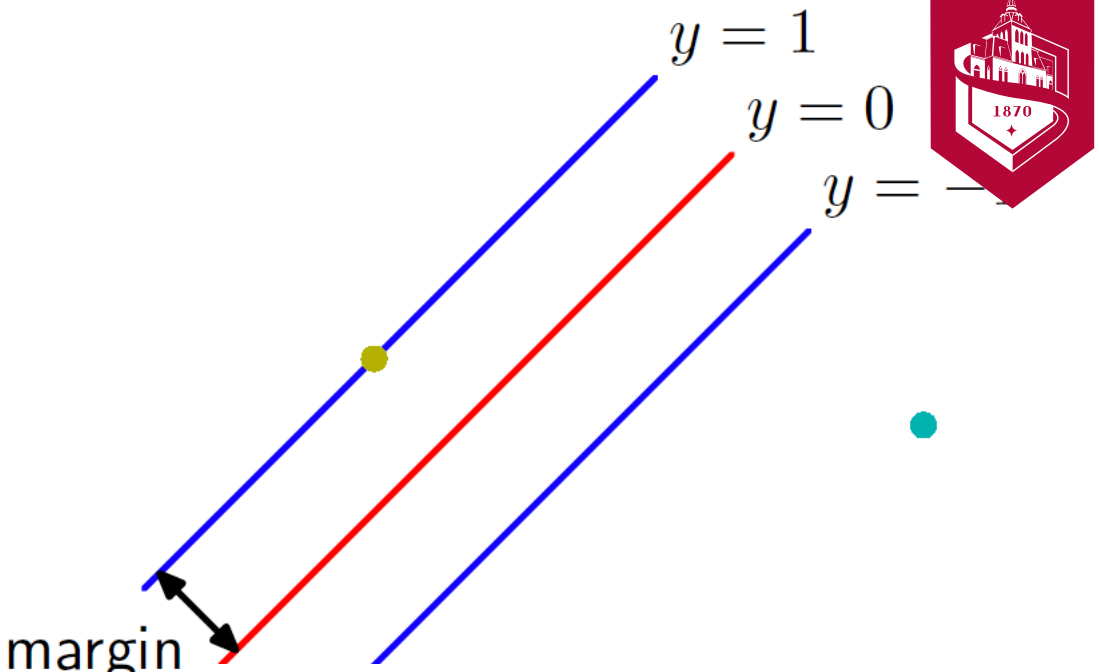
- guarantees to find a solution in a finite number of steps
- initial value for w and b starts from arbitrary chosen value..

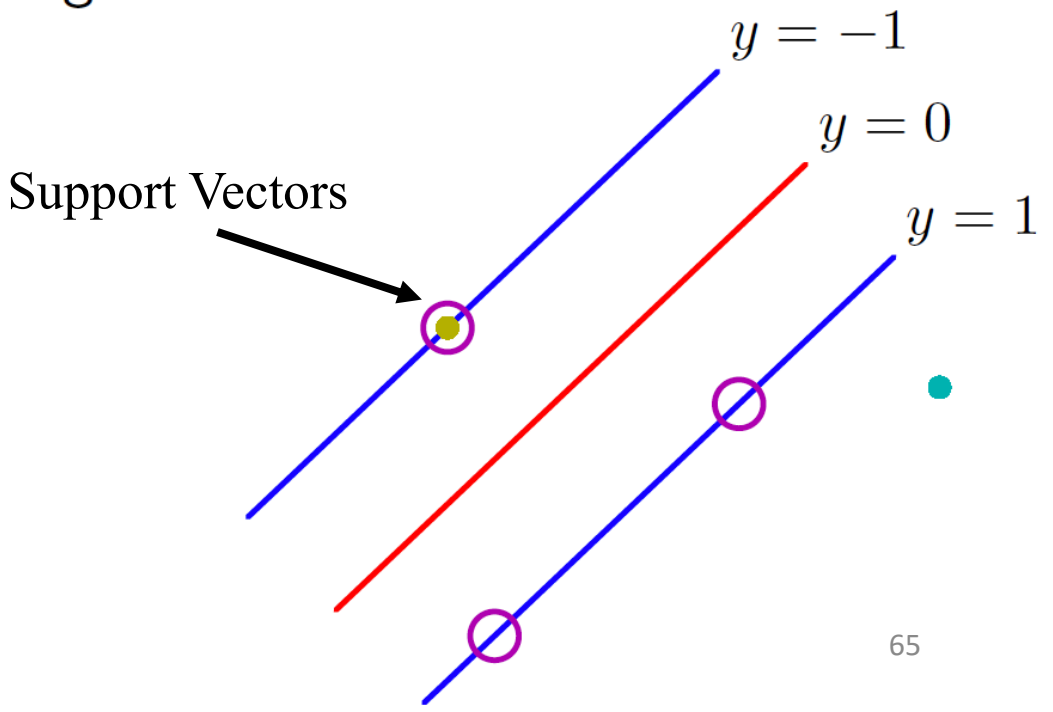
If there are multiple solutions of all of which classify the training data set exactly, then we should try to find the one that give **the smallest generalization error!**



SVM

- An extension of the Perceptron developed by Rosenblatt in 1958.
- Concept of the margin - the smallest distance between the decision boundary and any of the samples.
- Choose the max margin
- Using the optimal boundary, determine the best ***hyperplane*** by minimizing the probability of error relative to the learned density mode.
- hyperplane becomes independent of data points that are not ***support vectors (SV)***.





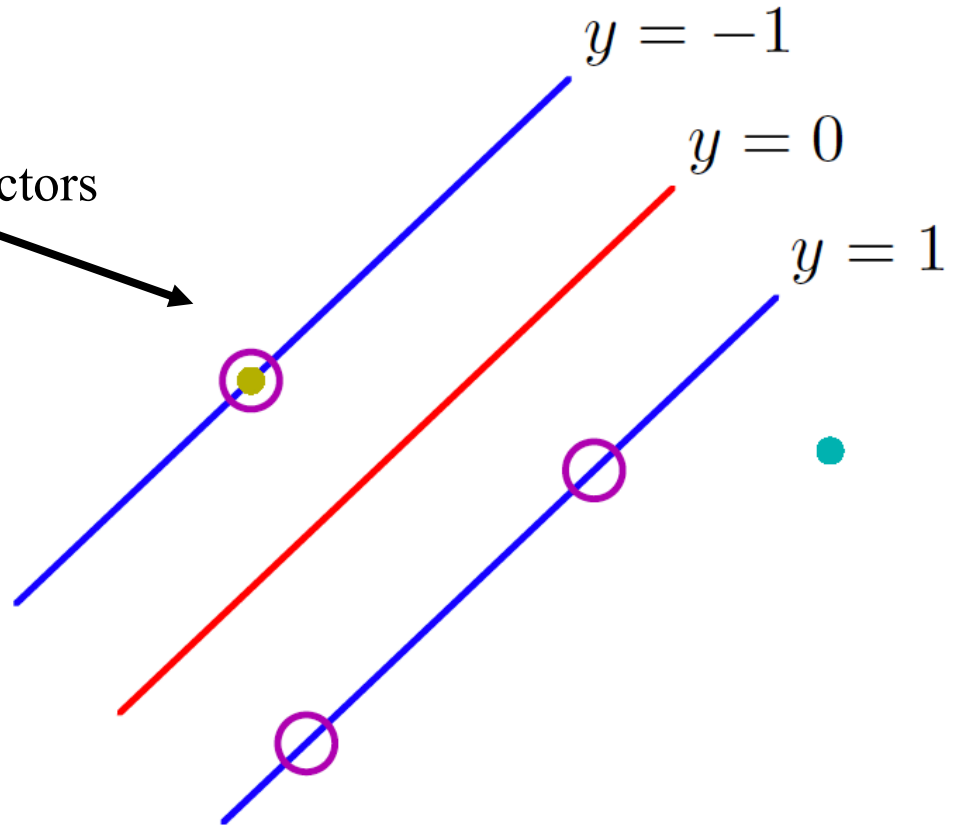
SVM

Hyperplane is $y(x) = 0$

Right classification $t_n y(x_n) > 0$

$$\frac{t_n y(\boldsymbol{x_n})}{||\boldsymbol{w}||} = \frac{t_n[\boldsymbol{w^T}\phi(\boldsymbol{x_n}) + b]}{||\boldsymbol{w}||}$$

Support Vectors



Optimize the parameters w and b to maximize the distance.

Maximum margin solution is

$$\underset{\boldsymbol{w},b}{\operatorname{argmax}} \left\{ \frac{1}{||\boldsymbol{w}||} \min_n \left[t_n \left(\boldsymbol{w^T}\phi(\boldsymbol{x_n}) + b \right) \right] \right\}$$

w does not depend on n. canonical representation of the decision hyperplane.

SVM

Direction solution is quite complex, we convert it into an equivalent problem (rescale)

$$t_n(w^T\phi(x)+b)=1$$

for the point that is closest to the surface. All data points will satisfy the constrains

$$t_n(w^T\phi(x)+b)\geq 1$$

once the margin has been maximized there will be at least two active constraints.

So we maximize $||w||^{-1}$ and this is same as minimizing $||w||^2$

$$\underset{w,b}{\operatorname{argmin}}\frac{1}{2}||w||^2$$

Lagrange multipliers

We introduce Lagrange Multipliers $a_n \geq 0$

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\left|\left|w\right|\right|^2 - \sum_{n=1}^{N} a_n \left\{t_n\left(\boldsymbol{w}^T \phi(\boldsymbol{x_n}) + b\right) - 1\right\} \quad (1)$$

- a strategy for finding the local maxima and minima of a function subject to equality constraints

Derivative:

$$w = \sum_{n=1}^{N} a_n \, t_n \phi(x_n)$$

$$0 = \sum_{n=1}^{N} a_n t_n$$

Dual Representation

Dual representation of the maximum margin problem

$$L(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{m=1}^{N}\sum_{n=1}^{N} a_n\, a_m t_n t_m k(x_n, x_m) \qquad (2)$$

Use the kernel function.

This takes the form of a quadratic programming problem

$$a_n \geq 0$$

$$0 = \sum_{n=1}^{N} a_n t_n$$

Prediction

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x_n}) + b \qquad (3)$$

Satisfies the following conditions:

$$a_n \geq 0$$
$$t_n y(x_n) - 1 \geq 0$$
$$a_n\{t_n y(x_n) - 1\} = 0$$

for point with $a_n = 0$ will not appear in the sum and plays no role making predictions for the new point.

The remaining data points are called support vectors because they satisfy $t_n y(x_n) = 1$ and correspond to points that lie on the maximum margin hyperplanes in feature space.

Prediction

Using Eq (3)

$$t_n\left(\sum_{m\in S} a_m t_m k(x_n, x_m) + b\right) = 1 \tag{4}$$

S denotes the set of indices of the support vectors (SV).

Making a use of $t_n^2 = 1$ averaging over all SV and solve for b

$$b = \frac{1}{N_s}\left(\sum_{n\in S}\left(t_n - \sum_{m\in S} a_m t_m k(x_n, x_m)\right)\right) \tag{5}$$

N_s is the total number of SV.

The maximum margin classifier in terms of the minimization of an error function

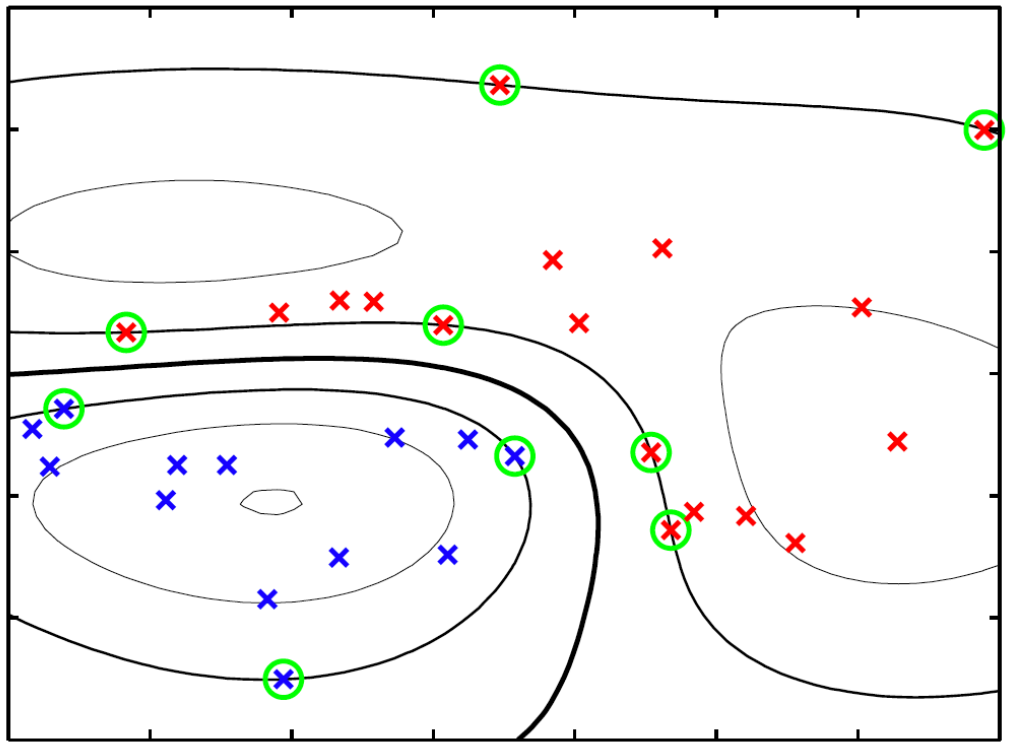
$$\sum_{n=1}^{N} E_\infty(y(x_n)t_n - 1) + \lambda\left|\left|w\right|\right|^2 \tag{6}$$

Prediction

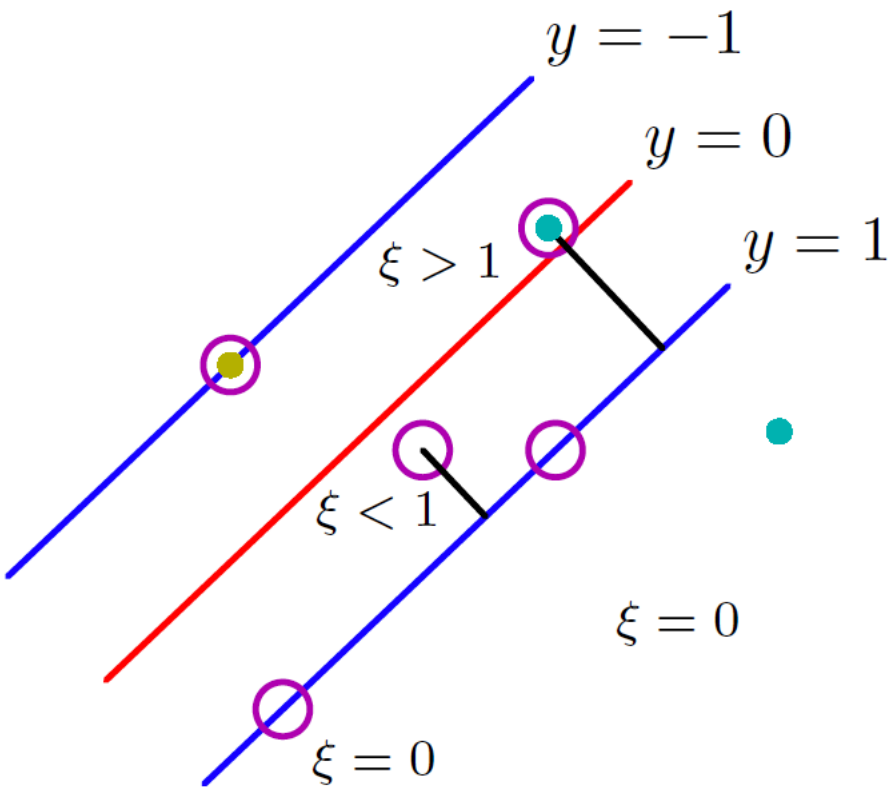
The maximum margin classifier in terms of the minimization of an error function

$$\sum_{n=1}^{N} E_{\infty}(y(x_n)t_n - 1) + \lambda\left|\left|w\right|\right|^2 \qquad (6)$$

where E is a function that is zero if $z \geq 0$ and ∞ otherwise to ensure that the constraints satisfies.



Overlapping Class Distributions



The class-conditional distributions may overlap, in which case exact separation of the training data can lead to poor generalization.

Need to modify SVM

Modify so that data points are allowed to be on the wrong side of the margin boundary, but with penalty that increases with the distance from that boundary.

Overlapping Class Distributions – soft SVM

Make the penalty a linear function of the distance.

slack variable $\xi_n \geq 0$

$\xi = 0$ for data points that are on or inside the correct margin boundary and $\xi = |t_n - y(x_n)|$ for other points.
On the decision boundary, $y(x_n) = 0$ and have $\xi = 1$ and with $\xi > 1$ will be misclassified

$$t_n y(x_n) \geq 1 - \xi_n$$

Goal is to maximize the margin while **softly penalizing points** that lie on the wrong side of the margin boundary

Overlapping Class Distributions – soft SVM

$$C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\left|\left|w\right|\right|^2 \tag{7}$$

where parameter C>0 controls the trade-off between the slack variable penalty and the margin.

$$L(w,b,\xi,a,\mu) = \frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_{n=1}^{N}\xi_n + \sum_{n=1}^{N}a_n\{t_n y_n(x_n) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n \tag{8}$$

Lagrange Multipliers - $\{a_n \geq 0\}$ and $\{\mu_n \geq 0\}$:

$a_n = 0, t_n y_n(x_n) - 1 + \xi_n \geq 0, a_n\{t_n y_n(x_n) - 1 + \xi_n\} = 0$

$\mu_n \geq 0, \xi_n \geq 0, \mu_n\xi_n = 0$

Overlapping Class Distributions – soft SVM

Optimize out w, b, and ξ

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^{N} a_n t_n \phi(x_n)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} a_n t_n = 0$$

$$\frac{\partial L}{\partial \xi_N} = 0 \Rightarrow a_n = C - \mu_n$$

$0 \leq a_n \leq C$ Box Constraints

$$L(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} a_n \, a_m t_n t_m k(x_n, x_m) \Big) \qquad (9)$$

Overlapping Class Distributions – soft SVM

Solution

$$t_n\left(\sum_{m\in S} a_m t_m k(x_n, x_m) + b\right) = 1 \qquad (10\text{-}1)$$

$$b = \frac{1}{N_{\mathcal{M}}}\left(\sum_{n\in\mathcal{M}}\left(t_n - \sum_{m\in S} a_m t_m k(x_n, x_m)\right)\right) \qquad (10\text{-}2)$$

where M is the set of indices of data points having $0 < a_n < C$.

Overlapping Class Distributions – soft SVM

IF we wish to use the SVM as a module in a larger probabilistic system

Platt (2000) has proposed fitting a logistic sigmoid to the outputs of a trained SVM.

$$p(t = 1|x) = \sigma(Ay(x) + B)$$

A & B are found by minimizing the cross-entropy error function defined by a training set consisting of pairs of values y and t.

- The data used to fit the sigmoid needs to be independent to avoid sever over-fitting.
- Give a poor approximation to the posterior prob.

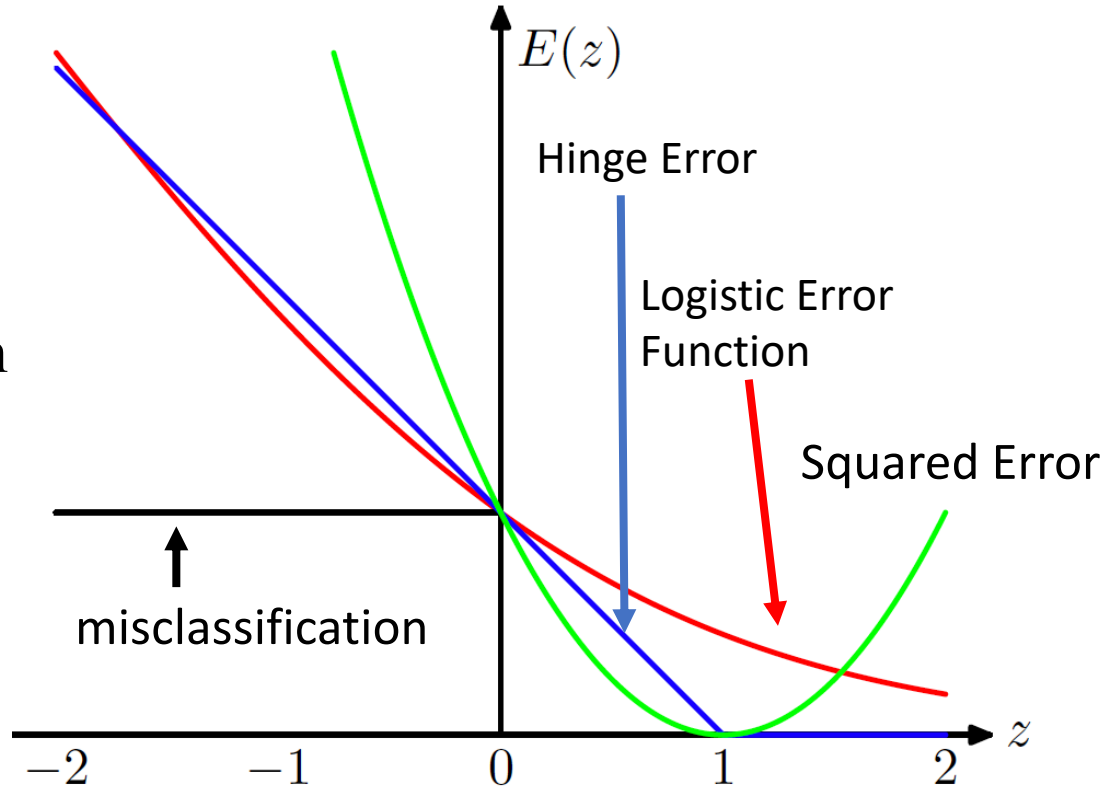
Relation to logistic regression – Hinge Error

Update Eq (7)

$$\sum_{n=1}^{N} E_{SV}(y_n t_n) + \lambda ||w||^2 \qquad (11)$$

where $\lambda = (2C)^{-1}$ and E_{SV} is the **hinge error** function

$$E_{SV}(y_n t_n) = [1 - y_n t_n]_+ \qquad (12)$$



Relation to logistic regression

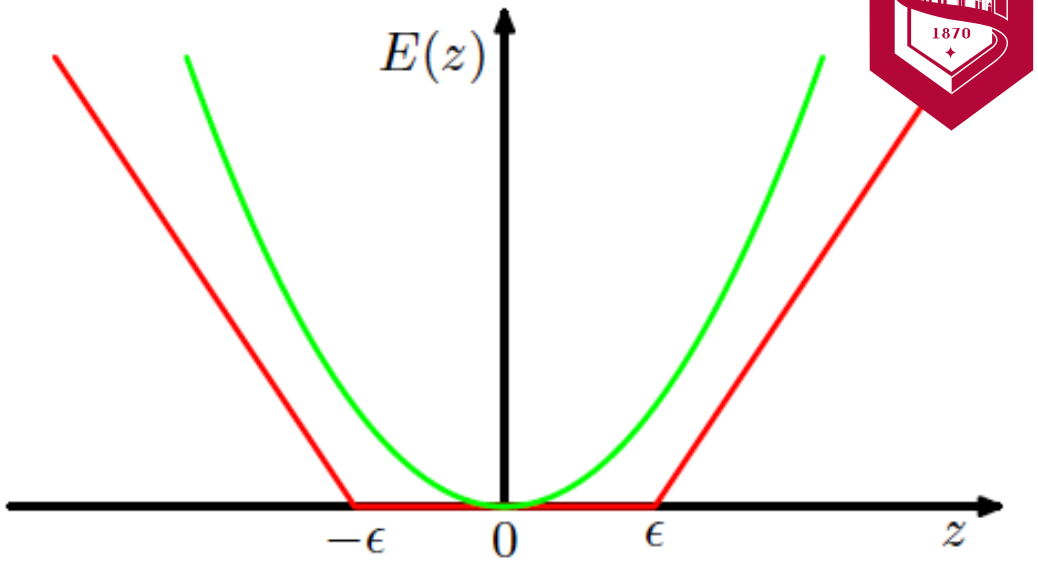
	SVMs	Logistic Regression
Loss Function	Hinge Loss	Log-loss
High dimensional features	Yes	No
Solution Sparse	Yes	Almost no
Output	Margin	Real Probabilities!

SVM for regression

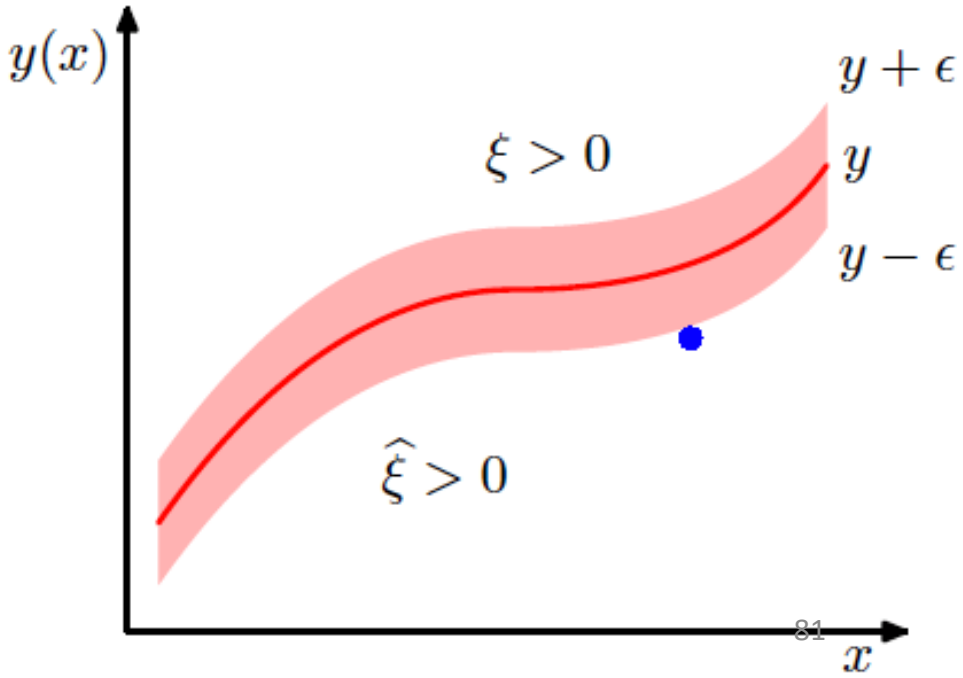
$$\frac{1}{2}\sum_{n=1}^{N}\{y_n - t_n\}^2 + \frac{\lambda}{2}||w||^2 \qquad (13)$$

ϵ-insensitive error function (Vapnik, 1995)

$$E_\epsilon(y(x) - t) = \begin{cases} 0 & if\ |y(x) - t| < \epsilon \\ |y(x) - t| & \text{Otherwise} \end{cases}$$



$$C\sum_{n=1}^{N} E_\epsilon(y_n(x_n) - t_n) + \frac{\lambda}{2}||w||^2 \qquad (14)$$



$\xi_n > 0$ corresponds to a point $t_n > y(x_n) + \epsilon$ and
$\hat{\xi}_n \geq 0$ for $t_n < y(x_n) - \epsilon$

$$C\sum_{n=1}^{N}\left(\xi_n + \hat{\xi}_n\right) + \frac{1}{2}||w||^2 \qquad (15)$$

SVM for regression - Optimize by slack variable

$$
\begin{aligned}
&\mathrm{L} \\
&= C\sum_{n=1}^{N}\left(\xi_n + \hat{\xi}_n\right) + \frac{1}{2}\left|\left|w\right|\right|^2 - \sum_{n=1}^{N}\left(\mu_n\xi_n + \hat{\mu}_n\hat{\xi}_n\right) - \sum_{n=1}^{N} a_n(\epsilon + \xi_n + y_n - t_n) \\
&- \sum_{n=1}^{N} \hat{a}_n\left(\epsilon + \hat{\xi}_n - y_n + t_n\right)
\end{aligned}
\tag{16}
$$

The derivative w.r.t. w, b, ξ_n, and $\hat{\xi}_n$

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^{N}(a_n - \hat{a}_n)\phi(x_n)\,, \frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N}(a_n - \hat{a}_n) = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow a_n = C - \mu_n, \frac{\partial L}{\partial \hat{\xi}_n} = 0 \Rightarrow \hat{a}_n = C - \hat{\mu}_n$$

SVM for regression - Optimize by slack variable

$$L(a,\hat{a}) = -\frac{1}{2}\sum_{m=1}^{N}\sum_{n=1}^{N}(a_n-\hat{a}_n)(a_m-\hat{a}_m)k(x_n,x_m) - \epsilon\sum_{n=1}^{N}(a_n+\hat{a}_n) + \sum_{n=1}^{N}(a_n-\hat{a}_n)\,t_n \quad (17)$$

$$y(x) = \sum_{n=1}^{N}(a_n-\hat{a}_n)k(x,x_n) + b$$

$$a_n(\epsilon+\xi_n+y_n-t_n) = 0$$
$$\hat{a}_n(\epsilon+\hat{\xi}_n-y_n+t_n) = 0$$
$$(C-a_n)\xi_n = 0$$
$$(C-\hat{a}_n)\hat{\xi}_n = 0$$

$$b = t_n - \epsilon - w^T\phi(x_n) = t_n - \epsilon - \sum_{m=1}^{N}(a_m-\hat{a}_m)k(x_n,x_m) \quad (18)$$

$$L(a,\hat{a}) = -\frac{1}{2}\sum_{m=1}^{N}\sum_{n=1}^{N}(a_n-\hat{a}_n)(a_m-\hat{a}_m)k(x_n,x_m) + \sum_{n=1}^{N}(a_n-\hat{a}_n)\,t_n \quad (19)$$

Kernel SVM

- Outputs represents decisions (can be an advantage but also can be a disadvantage)
- Originally formulated for two classes
- Predictions are linear combination of kernel functions **but centered on training data points and that are required to be positive-definite**.

- How can we overcome the limitation of centered kernel?

- The relevance vector machine or RVM (Tipping 2001)
  - Faster performance on test data while the generalization error is maintained.

RVM - Regression

The modified prior:

$$p(t|x, w, \beta) = \mathcal{N}(t|y(x), \beta^{-1}) \quad (20)$$

where $\beta = \sigma^{-2}$ is the noise precision.

The mean of the linear model

$$y(x) = \sum_{i=1}^{M} w_i \phi_i(x) = w^T \phi(x)$$

In RVM, we use kernels with one kernel associated with each of the data points in training set.

$$y(x) = \sum_{i=1}^{N} w_n k(x, x_n) + b \quad (21)$$

where b is the bias parameter and $M = N + 1$.

Here, **there is no restriction of positive-definite and the number nor the location to the training data sets.**

RVM - Regression

Suppose we have N observations of data matrix X. The likelihood function is

$$p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{w},\beta) = \prod_{n=1}^{N} p(t_n|\boldsymbol{x}_n,\boldsymbol{w},\beta)$$

and a weight prior distribution is

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{n=1}^{M} \mathcal{N}(w_i|0,\alpha_i^{-1})$$

α_i is a separate hyperparameter for each of weight parameter.

If the posterior distribution for the weights is Gaussian, then it is

$$p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{w},\beta) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m},\boldsymbol{\Sigma})$$

Mean: $\beta\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{t} = \beta\boldsymbol{\Sigma}\mathbf{K}^{\mathbf{T}}\mathbf{t}$

Covariance: $\Sigma = \left(\boldsymbol{A} + \beta\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\Phi}\right)^{-1}$

$= \left(\boldsymbol{A} + \beta\boldsymbol{K}^{\boldsymbol{T}}\boldsymbol{K}\right)^{-1}$

The maximization of likelihood function by integration of the weight parameters

$$p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{\alpha},\beta) = \int p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{w},\beta)p(\boldsymbol{w}|\boldsymbol{\alpha})d\boldsymbol{w} \qquad (22)$$

RVM - Regression

If the posterior distribution for the weights is Gaussian, then it is

$$p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{w},\beta) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m},\boldsymbol{\Sigma})$$

The maximization of likelihood function by integration of the weight parameters

$$p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{\alpha},\beta) = \int p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{w},\beta)p(\boldsymbol{w}|\boldsymbol{\alpha})d\boldsymbol{w} \quad (22)$$

The log likelihood also can be formatted as

$$\ln p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{\alpha},\beta) = \ln \mathcal{N}(\boldsymbol{t}|\boldsymbol{0},\boldsymbol{C}) - \frac{1}{2}\{N\ln(2\pi) + \ln|\boldsymbol{C}| + \boldsymbol{t}^T\boldsymbol{C}^{-1}\boldsymbol{t}\} \quad (23)$$

where $\boldsymbol{C} = \beta^{-1}\boldsymbol{I} + \boldsymbol{\Phi}\boldsymbol{A}^{-1}\boldsymbol{\Phi}^T$.

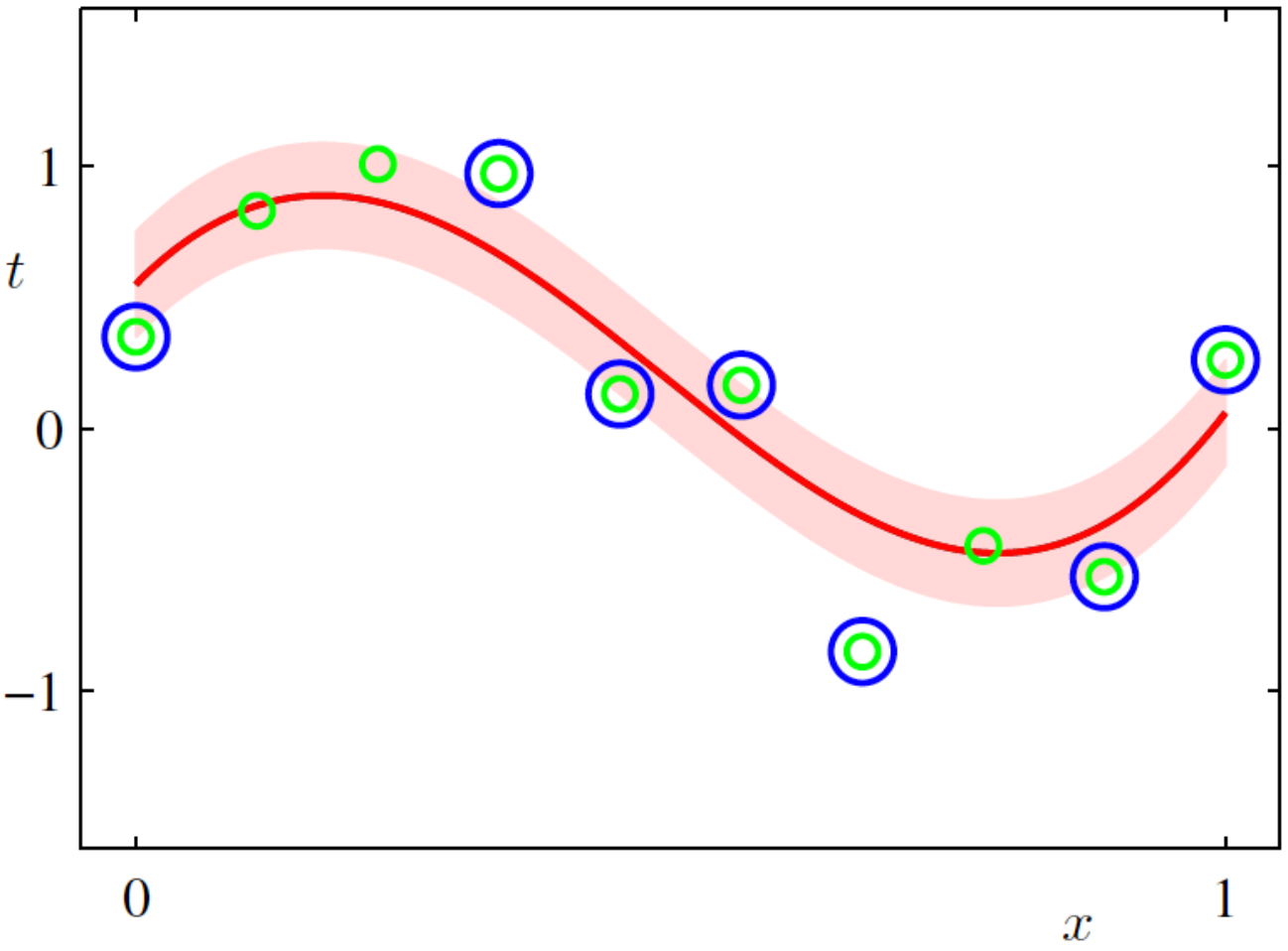
Set the required derivatives of the likelihood to zero and find the hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$:

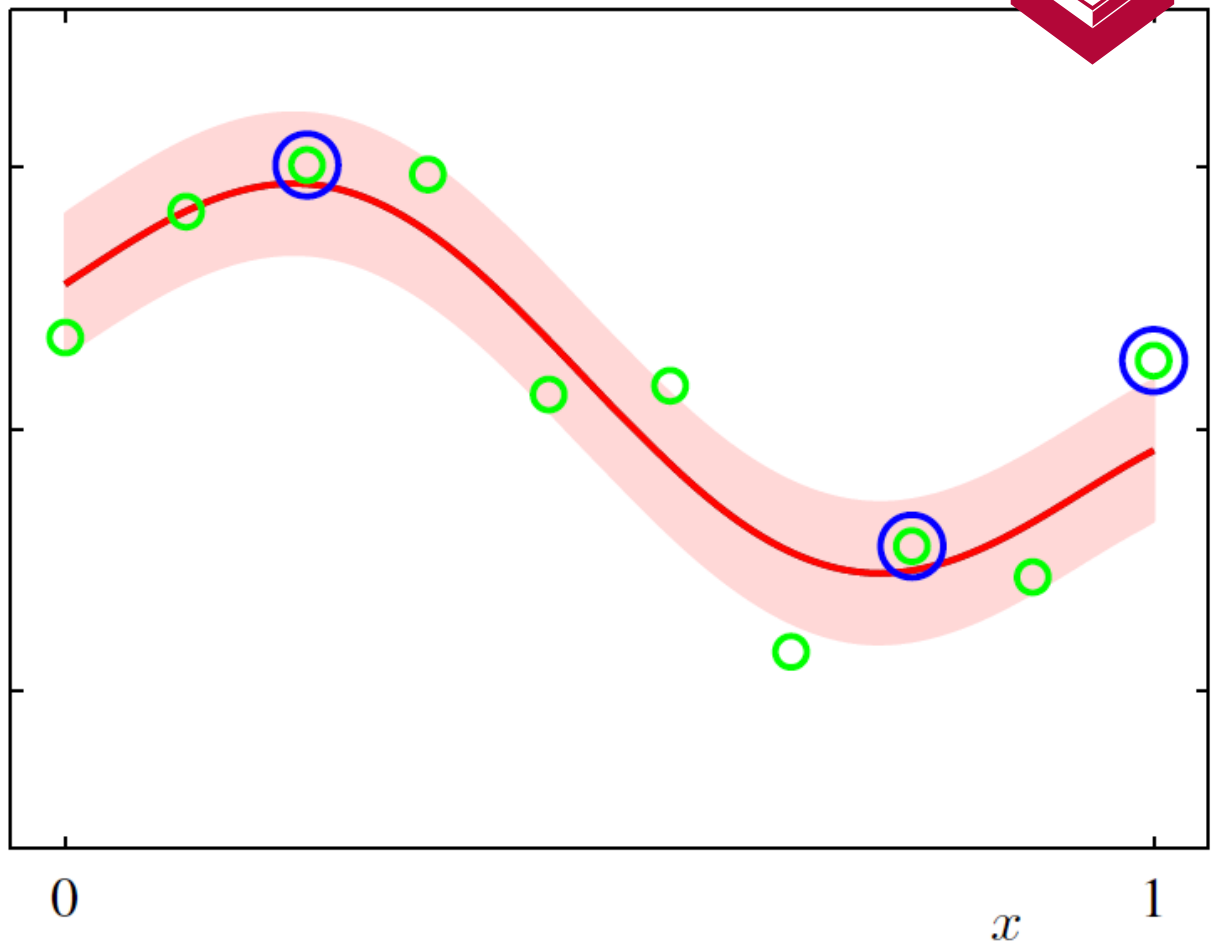
$$\alpha_i^{new} = \frac{\gamma_i}{m_i^2}$$

$$(\beta^{new})^{-1} = \frac{\left||\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{m}|\right|^2}{N - \sum_i \gamma_i}$$

where m_i is the i^{th} component of the poster mean $\boldsymbol{m}$ and the quantity $\gamma_i = 1 - \alpha_i\Sigma_{ii}$ is the measurement of how well w_i is measured.

RVM - Regression





RVM - Regression

Linear Regression – the predictive variance becomes small in regions of input space.

SVM Regression - ϕ is centered on data points and the model will become increasingly certain of its predictions when extrapolating outside the domain of the data.

Gaussian Process – the computation cost is high.

RVM – A significant improvement in the speed of processing on test data. This greater sparsity is achieved with little or no reduction in generalization error compared with SVM.

But… Training involves optimizing a nonconvex function and takes longer.

RVM - Classification

Let's begin from the two-class problem with a binary target $t \in \{0,1\}$.

If the model takes the linear combination format of transformation function $\phi(\cdot)$ by a logistic sigmoid function

$$y(\boldsymbol{x}, \boldsymbol{w}) = \sigma\left(w^T \phi(\boldsymbol{x})\right).$$

The posterior distribution over w (assume it is a Gaussian prior) is

$$p(\boldsymbol{w}|\boldsymbol{t}) \propto p(\boldsymbol{w}) p(\boldsymbol{t}|\boldsymbol{w}).$$

We can solve for $\boldsymbol{\alpha}$ by following the way we saw in RVM for regression using Hassian matrix.

RVM - Classification

$$\ln p(\boldsymbol{w}|\boldsymbol{t},\boldsymbol{\alpha}) = \ln\{p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})\} - \ln p(\boldsymbol{t}|\boldsymbol{\alpha})$$

$$= \sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} - \frac{1}{2}\boldsymbol{w}^T\boldsymbol{A}\boldsymbol{w} + const.$$

From logistic regression lecture,
Refer to equation 22.

Regularization comes from the prior in the previous slide,
$p(\boldsymbol{w}|\boldsymbol{t}) \propto p(\boldsymbol{w})p(\boldsymbol{t}|\boldsymbol{w})$

Note: $\ln p(w|t) = -\frac{1}{2}(\boldsymbol{w}-\boldsymbol{m}_0)^T\boldsymbol{S}_0^{-1}(\boldsymbol{w}-\boldsymbol{m}_0) + \sum_{n=1}^{N}\{t-y_n)\ln(1-y_n)\} + Const.$ when the Gaussian prior of **w** is $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m_0},\boldsymbol{S}_0)$

From here, we can take the gradient of $\ln p(w|t,\alpha)$ be zero to obtain the mean and α_i^{new} is

$$\alpha_i^{new} = \frac{\gamma_i}{m_i^2}$$