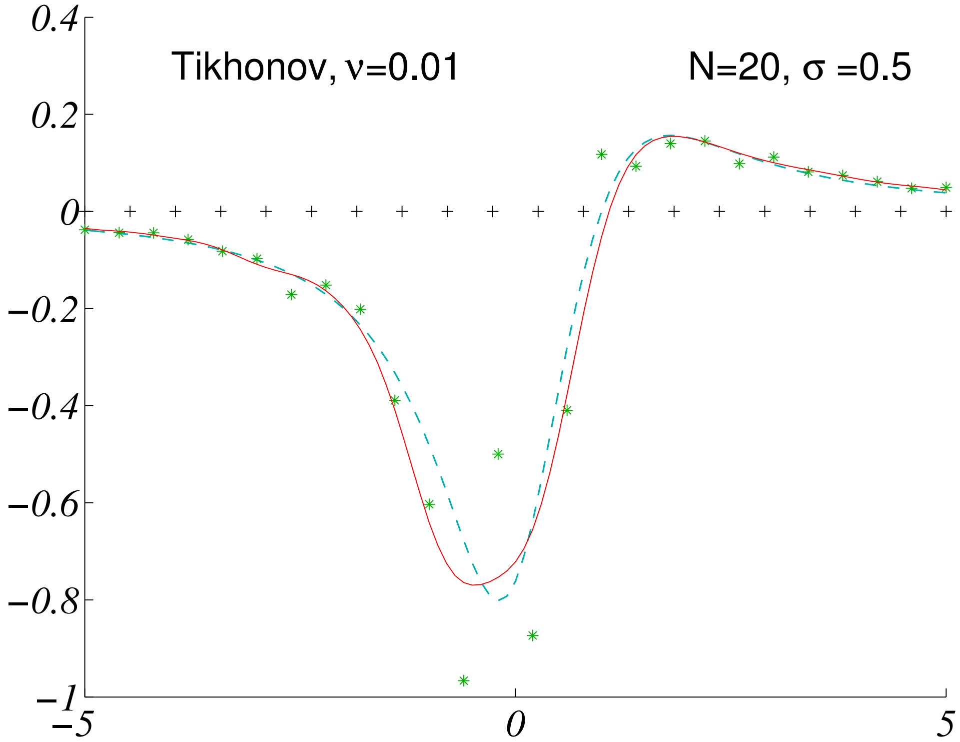
Advanced Machine Learning

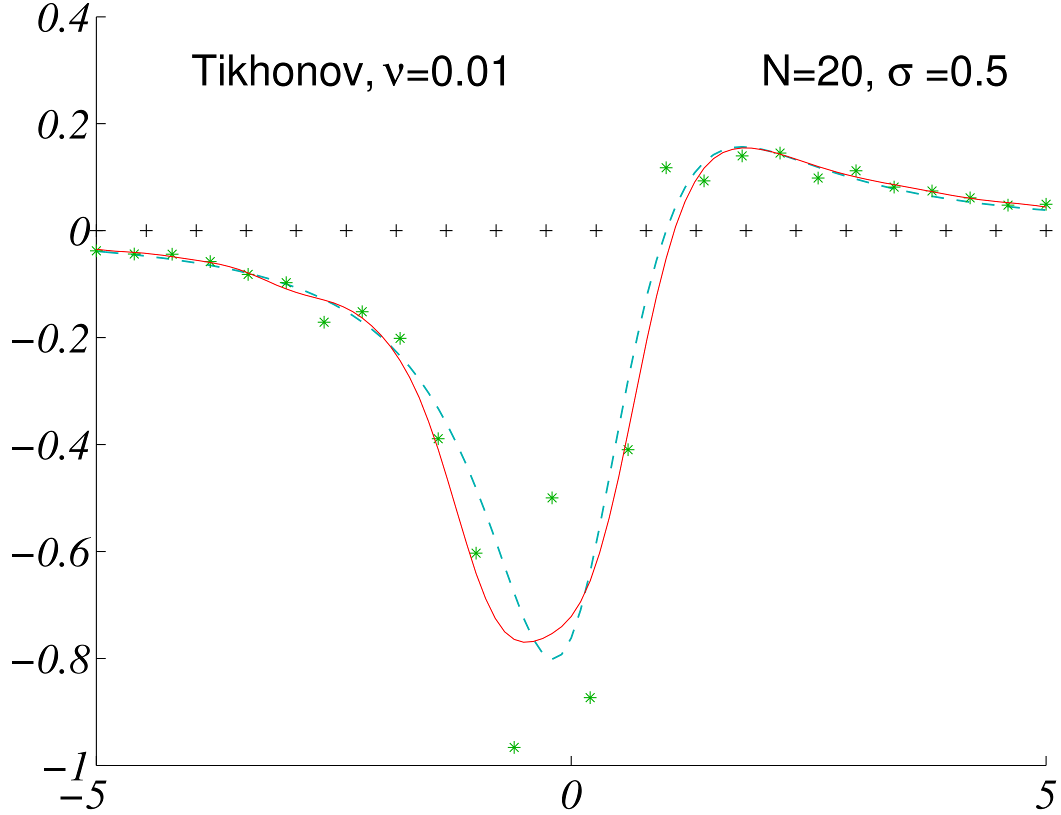
Regularisation



Regularisation, weight decay, Tikhonov

Outline

1. **Training Output Layer**
2. Regularisation
   - Weight Decay
   - Tikhonov



Back to Radial Basis Functions

- Recall in the last lecture that an RBF is a machine of the form

$$f(\boldsymbol{x}|\boldsymbol{w}, \{\boldsymbol{\mu}_i, \sigma_i\}_{i=1}^{K}) = \sum_{i=1}^{K} w_i \phi_i(\boldsymbol{x}) + w_0$$

- where $\phi_i(\boldsymbol{x}) = \psi\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{c}_i}{\sigma_i}\right\|\right)$ are the radial basis functions

- Given data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$, the weights in the output layer are learnt by linear least squares

$$E(\boldsymbol{w}|\mathcal{D}) = \frac{1}{P}\sum_{k=1}^{P}\left(\boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}}\boldsymbol{w} - y_k\right)^2 = \frac{1}{P}\left\|\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{w} - \boldsymbol{y}\right\|^2$$

Back to Radial Basis Functions

- Recall in the last lecture that an RBF is a machine of the form

$$f(\boldsymbol{x}|\boldsymbol{w}, \{\boldsymbol{\mu}_i, \sigma_i\}_{i=1}^{K}) = \sum_{i=1}^{K} w_i \phi_i(\boldsymbol{x}) + w_0$$

- where $\phi_i(\boldsymbol{x}) = \psi\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{c}_i}{\sigma_i}\right\|\right)$ are the radial basis functions

- Given data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$, the weights in the output layer are learnt by linear least squares

$$E(\boldsymbol{w}|\mathcal{D}) = \frac{1}{P}\sum_{k=1}^{P}\left(\boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}}\boldsymbol{w} - y_k\right)^2 = \frac{1}{P}\left\|\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{w} - \boldsymbol{y}\right\|^2$$

Back to Radial Basis Functions

- Recall in the last lecture that an RBF is a machine of the form

$$f(\boldsymbol{x}|\boldsymbol{w}, \{\boldsymbol{\mu}_i, \sigma_i\}_{i=1}^K) = \sum_{i=1}^{K} w_i \phi_i(\boldsymbol{x}) + w_0$$

- where $\phi_i(\boldsymbol{x}) = \psi\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{c}_i}{\sigma_i}\right\|\right)$ are the radial basis functions

- Given data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$, the weights in the output layer are learnt by linear least squares

$$E(\boldsymbol{w}|\mathcal{D}) = \frac{1}{P}\sum_{k=1}^{P}\left(\boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}}\boldsymbol{w} - y_k\right)^2 = \frac{1}{P}\left\|\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{w} - \boldsymbol{y}\right\|^2$$

Linear Least Squares

- Same as linear perceptron
- Output $\boldsymbol{\phi}(\boldsymbol{x})^\mathsf{T}\boldsymbol{w}$ where $\phi_k(\boldsymbol{x}) = \phi\left(\frac{\|\boldsymbol{x}-\boldsymbol{\mu}_k\|}{\sigma_k}\right)$
- Mean squared error

$$E = \frac{1}{P}\sum_{k=1}^{P}\left(\boldsymbol{\phi}(\boldsymbol{x}_k)^\mathsf{T}\boldsymbol{w} - y_k\right)^2 = \frac{1}{P}\left\|\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{w} - \boldsymbol{y}\right\|^2$$

where

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_2) & \cdots & \phi_1(\boldsymbol{x}_P) \\ \phi_2(\boldsymbol{x}_1) & \phi_2(\boldsymbol{x}_2) & \cdots & \phi_2(\boldsymbol{x}_P) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N(\boldsymbol{x}_1) & \phi_N(\boldsymbol{x}_2) & \cdots & \phi_N(\boldsymbol{x}_P) \end{pmatrix} \qquad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_P \end{pmatrix}$$

Linear Least Squares

- Same as linear perceptron
- Output $\boldsymbol{\phi}(\boldsymbol{x})^\mathsf{T}\boldsymbol{w}$ where $\phi_k(\boldsymbol{x}) = \phi\left(\frac{\|\boldsymbol{x}-\boldsymbol{\mu}_k\|}{\sigma_k}\right)$
- Mean squared error

$$E = \frac{1}{P}\sum_{k=1}^{P}\left(\boldsymbol{\phi}(\boldsymbol{x}_k)^\mathsf{T}\boldsymbol{w} - y_k\right)^2 = \frac{1}{P}\left\|\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{w} - \boldsymbol{y}\right\|^2$$

where

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_2) & \cdots & \phi_1(\boldsymbol{x}_P) \\ \phi_2(\boldsymbol{x}_1) & \phi_2(\boldsymbol{x}_2) & \cdots & \phi_2(\boldsymbol{x}_P) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N(\boldsymbol{x}_1) & \phi_N(\boldsymbol{x}_2) & \cdots & \phi_N(\boldsymbol{x}_P) \end{pmatrix} \qquad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_P \end{pmatrix}$$

Linear Least Squares

- Same as linear perceptron
- Output $\boldsymbol{\phi}(\boldsymbol{x})^\mathsf{T}\boldsymbol{w}$ where $\phi_k(\boldsymbol{x}) = \phi\left(\frac{\|\boldsymbol{x}-\boldsymbol{\mu}_k\|}{\sigma_k}\right)$
- Mean squared error

$$E = \frac{1}{P}\sum_{k=1}^{P}\left(\boldsymbol{\phi}(\boldsymbol{x}_k)^\mathsf{T}\boldsymbol{w} - y_k\right)^2 = \frac{1}{P}\left\|\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{w} - \boldsymbol{y}\right\|^2$$

where

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_2) & \cdots & \phi_1(\boldsymbol{x}_P) \\ \phi_2(\boldsymbol{x}_1) & \phi_2(\boldsymbol{x}_2) & \cdots & \phi_2(\boldsymbol{x}_P) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N(\boldsymbol{x}_1) & \phi_N(\boldsymbol{x}_2) & \cdots & \phi_N(\boldsymbol{x}_P) \end{pmatrix} \qquad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_P \end{pmatrix}$$

Linear Least Squares

- Same as linear perceptron
- Output $\boldsymbol{\phi}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{w}$ where $\phi_k(\boldsymbol{x}) = \phi\left(\frac{\|\boldsymbol{x}-\boldsymbol{\mu}_k\|}{\sigma_k}\right)$
- Mean squared error

$$E = \frac{1}{P}\sum_{k=1}^{P}\left(\boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}}\boldsymbol{w} - y_k\right)^2 = \frac{1}{P}\left\|\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{w} - \boldsymbol{y}\right\|^2$$

where

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_2) & \cdots & \phi_1(\boldsymbol{x}_P) \\ \phi_2(\boldsymbol{x}_1) & \phi_2(\boldsymbol{x}_2) & \cdots & \phi_2(\boldsymbol{x}_P) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N(\boldsymbol{x}_1) & \phi_N(\boldsymbol{x}_2) & \cdots & \phi_N(\boldsymbol{x}_P) \end{pmatrix} \qquad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_P \end{pmatrix}$$

Linear Least Squares

- Same as linear perceptron
- Output $\boldsymbol{\phi}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{w}$ where $\phi_k(\boldsymbol{x}) = \phi\left(\frac{\|\boldsymbol{x}-\boldsymbol{\mu}_k\|}{\sigma_k}\right)$
- Mean squared error

$$E = \frac{1}{P}\sum_{k=1}^{P}\left(\boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}}\boldsymbol{w} - y_k\right)^2 = \frac{1}{P}\left\|\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{w} - \boldsymbol{y}\right\|^2$$

where

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_2) & \cdots & \phi_1(\boldsymbol{x}_P) \\ \phi_2(\boldsymbol{x}_1) & \phi_2(\boldsymbol{x}_2) & \cdots & \phi_2(\boldsymbol{x}_P) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N(\boldsymbol{x}_1) & \phi_N(\boldsymbol{x}_2) & \cdots & \phi_N(\boldsymbol{x}_P) \end{pmatrix} \qquad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_P \end{pmatrix}$$

Least Squares Solution

- In matrix form

$$E = \frac{1}{P}\left(\boldsymbol{w}^{\top}\,\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\top}\,\boldsymbol{w} - 2\boldsymbol{w}^{\top}\,\boldsymbol{\Phi}\,\boldsymbol{y} + \boldsymbol{y}^{\top}\,\boldsymbol{y}\right)$$

- Minimum given by $\boldsymbol{\nabla} E = 0$

$$\boldsymbol{\nabla} E = \frac{2}{P}\left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\top}\,\boldsymbol{w} - \boldsymbol{\Phi}\,\boldsymbol{y}\right) = 0$$

- Optimal weight vector

$$\boldsymbol{w}^{*} = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\top}\right)^{-1}\boldsymbol{\Phi}\,\boldsymbol{y}$$

Least Squares Solution

- In matrix form

$$E = \frac{1}{P}\left(\boldsymbol{w}^{\mathsf{T}}\,\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}\,\boldsymbol{w} - 2\boldsymbol{w}^{\mathsf{T}}\,\boldsymbol{\Phi}\,\boldsymbol{y} + \boldsymbol{y}^{\mathsf{T}}\,\boldsymbol{y}\right)$$

- Minimum given by $\boldsymbol{\nabla} E = 0$

$$\boldsymbol{\nabla} E = \frac{2}{P}\left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}\,\boldsymbol{w} - \boldsymbol{\Phi}\,\boldsymbol{y}\right) = 0$$

- Optimal weight vector

$$\boldsymbol{w}^{*} = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}\right)^{-1}\boldsymbol{\Phi}\,\boldsymbol{y}$$

Least Squares Solution

- In matrix form

$$E = \frac{1}{P}\left(\boldsymbol{w}^{\mathsf{T}}\,\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}\,\boldsymbol{w} - 2\boldsymbol{w}^{\mathsf{T}}\,\boldsymbol{\Phi}\,\boldsymbol{y} + \boldsymbol{y}^{\mathsf{T}}\,\boldsymbol{y}\right)$$

- Minimum given by $\boldsymbol{\nabla} E = 0$

$$\boldsymbol{\nabla} E = \frac{2}{P}\left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}\,\boldsymbol{w} - \boldsymbol{\Phi}\,\boldsymbol{y}\right) = 0$$

- Optimal weight vector

$$\boldsymbol{w}^{*} = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}\right)^{-1}\boldsymbol{\Phi}\,\boldsymbol{y}$$

Singular Value Decomposition

- Remember the singular valued decomposition formula

$$\mathbf{\Phi} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}}$$

where

$$\mathbf{S} = \begin{pmatrix} s_1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & s_N & \cdots & 0 \end{pmatrix}$$

- $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, i.e. $\mathbf{U}^{\mathsf{T}} = \mathbf{U}^{-1}$ and $\mathbf{V}^{\mathsf{T}} = \mathbf{V}^{-1}$

Singular Value Decomposition

- Remember the singular valued decomposition formula

$$\mathbf{\Phi} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}}$$

where

$$\mathbf{S} = \begin{pmatrix} s_1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & s_N & \cdots & 0 \end{pmatrix}$$

- $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, i.e. $\mathbf{U}^{\mathsf{T}} = \mathbf{U}^{-1}$ and $\mathbf{V}^{\mathsf{T}} = \mathbf{V}^{-1}$

Singular Value Decomposition

- Remember the singular valued decomposition formula

$$\mathbf{\Phi} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}}$$

where

$$\mathbf{S} = \begin{pmatrix} s_1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & s_N & \cdots & 0 \end{pmatrix}$$

- $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, i.e. $\mathbf{U}^{\mathsf{T}} = \mathbf{U}^{-1}$ and $\mathbf{V}^{\mathsf{T}} = \mathbf{V}^{-1}$

$\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ Matrix

- Now the 'covariance-like matrix' is equal to

$$\begin{aligned}\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}} &= \left(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}}\right)\left(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}}\right)^{\mathsf{T}} \\ &= \left(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}}\right)\left(\mathbf{V}\,\mathbf{S}^{\mathsf{T}}\,\mathbf{U}^{\mathsf{T}}\right) \\ &= \mathbf{U}\left(\mathbf{S}\,\mathbf{S}^{\mathsf{T}}\right)\mathbf{U}^{\mathsf{T}} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathsf{T}}\end{aligned}$$

- $\boldsymbol{\Lambda} = \mathbf{S}\,\mathbf{S}^{\mathsf{T}} = \operatorname{diag}(s_1^2, s_2^2, \ldots, s_N^2) = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$

- ($\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ is a positive definite matrix with eigenvalues $\lambda_i = s_i^2$)

- The inverse covariance matrix is

$$\left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}\right)^{-1} = \mathbf{U}\,\boldsymbol{\Lambda}^{-1}\,\mathbf{U}^{\mathsf{T}}$$

- Since $(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}})^{-1}(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}) = \mathbf{U}\,\boldsymbol{\Lambda}^{-1}\,\mathbf{U}^{\mathsf{T}}\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathsf{T}} = \mathbf{I}$

$\Phi\,\Phi^\mathsf{T}$ Matrix

- Now the 'covariance-like matrix' is equal to

$$\begin{aligned}\Phi\,\Phi^\mathsf{T} &= \left(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^\mathsf{T}\right)\left(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^\mathsf{T}\right)^\mathsf{T} \\ &= \left(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^\mathsf{T}\right)\left(\mathbf{V}\,\mathbf{S}^\mathsf{T}\,\mathbf{U}^\mathsf{T}\right) \\ &= \mathbf{U}\left(\mathbf{S}\,\mathbf{S}^\mathsf{T}\right)\mathbf{U}^\mathsf{T} = \mathbf{U}\Lambda\mathbf{U}^\mathsf{T}\end{aligned}$$

- $\Lambda = \mathbf{S}\,\mathbf{S}^\mathsf{T} = \mathrm{diag}(s_1^2, s_2^2, \ldots, s_N^2) = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$

- ($\Phi\,\Phi^\mathsf{T}$ is a positive definite matrix with eigenvalues $\lambda_i = s_i^2$)

- The inverse covariance matrix is

$$\left(\Phi\,\Phi^\mathsf{T}\right)^{-1} = \mathbf{U}\,\Lambda^{-1}\,\mathbf{U}^\mathsf{T}$$

- Since $(\Phi\,\Phi^\mathsf{T})^{-1}(\Phi\,\Phi^\mathsf{T}) = \mathbf{U}\,\Lambda^{-1}\,\mathbf{U}^\mathsf{T}\mathbf{U}\Lambda\mathbf{U}^\mathsf{T} = \mathbf{I}$

$\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ Matrix

- Now the 'covariance-like matrix' is equal to

$$\begin{aligned}\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}} &= (\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}})\,(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}})^{\mathsf{T}} \\ &= (\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}})\,(\mathbf{V}\,\mathbf{S}^{\mathsf{T}}\,\mathbf{U}^{\mathsf{T}}) \\ &= \mathbf{U}\,(\mathbf{S}\,\mathbf{S}^{\mathsf{T}})\,\mathbf{U}^{\mathsf{T}} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathsf{T}}\end{aligned}$$

- $\boldsymbol{\Lambda} = \mathbf{S}\,\mathbf{S}^{\mathsf{T}} = \mathrm{diag}(s_1^2, s_2^2, \ldots, s_N^2) = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$

- ($\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ is a positive definite matrix with eigenvalues $\lambda_i = s_i^2$)

- The inverse covariance matrix is

$$(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}})^{-1} = \mathbf{U}\,\boldsymbol{\Lambda}^{-1}\,\mathbf{U}^{\mathsf{T}}$$

- Since $(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}})^{-1}(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}) = \mathbf{U}\,\boldsymbol{\Lambda}^{-1}\,\mathbf{U}^{\mathsf{T}}\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathsf{T}} = \mathbf{I}$

$\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}}$ Matrix

- Now the 'covariance-like matrix' is equal to

$$\begin{aligned}\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}} &= \left(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}}\right)\left(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}}\right)^{\mathsf{T}}\\ &= \left(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}}\right)\left(\mathbf{V}\,\mathbf{S}^{\mathsf{T}}\,\mathbf{U}^{\mathsf{T}}\right)\\ &= \mathbf{U}\left(\mathbf{S}\,\mathbf{S}^{\mathsf{T}}\right)\mathbf{U}^{\mathsf{T}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathsf{T}}\end{aligned}$$

- $\mathbf{\Lambda} = \mathbf{S}\,\mathbf{S}^{\mathsf{T}} = \mathrm{diag}(s_1^2, s_2^2, \ldots, s_N^2) = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$

- ($\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}}$ is a positive definite matrix with eigenvalues $\lambda_i = s_i^2$)

- The inverse covariance matrix is

$$\left(\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}}\right)^{-1} = \mathbf{U}\,\mathbf{\Lambda}^{-1}\,\mathbf{U}^{\mathsf{T}}$$

- Since $(\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}})^{-1}(\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}}) = \mathbf{U}\,\mathbf{\Lambda}^{-1}\,\mathbf{U}^{\mathsf{T}}\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathsf{T}} = \mathbf{I}$

$\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ Matrix

- Now the 'covariance-like matrix' is equal to

$$\begin{aligned} \boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}} &= (\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}})\,(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}})^{\mathsf{T}} \\ &= (\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}})\,(\mathbf{V}\,\mathbf{S}^{\mathsf{T}}\,\mathbf{U}^{\mathsf{T}}) \\ &= \mathbf{U}\,(\mathbf{S}\,\mathbf{S}^{\mathsf{T}})\,\mathbf{U}^{\mathsf{T}} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathsf{T}} \end{aligned}$$

- $\boldsymbol{\Lambda} = \mathbf{S}\,\mathbf{S}^{\mathsf{T}} = \mathrm{diag}(s_1^2, s_2^2, \ldots, s_N^2) = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$

- ($\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ is a positive definite matrix with eigenvalues $\lambda_i = s_i^2$)

- The inverse covariance matrix is

$$\left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}\right)^{-1} = \mathbf{U}\,\boldsymbol{\Lambda}^{-1}\,\mathbf{U}^{\mathsf{T}}$$

- Since $(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}})^{-1}(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}) = \mathbf{U}\,\boldsymbol{\Lambda}^{-1}\,\mathbf{U}^{\mathsf{T}}\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathsf{T}} = \mathbf{I}$

Least-Squares Solution

- The inverse covariance matrix is

$$\left(\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}}\right)^{-1} = \mathbf{U}\,\left(\mathbf{S}\,\mathbf{S}^{\mathsf{T}}\right)^{-1}\mathbf{U}^{\mathsf{T}}$$

- where $\left(\mathbf{S}\,\mathbf{S}^{\mathsf{T}}\right)^{-1} = \mathrm{diag}(s_1^{-2}, s_2^{-2}, \cdots, s_N^{-2})$
- Thus

$$\begin{aligned}\left(\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}}\right)^{-1}\mathbf{\Phi} &= \left(\mathbf{U}\left(\mathbf{S}\,\mathbf{S}^{\mathsf{T}}\right)^{-1}\mathbf{U}^{\mathsf{T}}\right)\left(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}}\right)\\ &= \mathbf{U}\left(\mathbf{S}\,\mathbf{S}^{\mathsf{T}}\right)^{-1}\mathbf{S}\,\mathbf{V}^{\mathsf{T}}\\ &= \mathbf{U}\,\mathbf{S}^{+}\,\mathbf{V}^{\mathsf{T}}\end{aligned}$$

where $\mathbf{S}^{+} = \left(\mathbf{S}\,\mathbf{S}^{\mathsf{T}}\right)^{-1}\mathbf{S}$

Least-Squares Solution

- The inverse covariance matrix is

$$\left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}\right)^{-1} = \mathbf{U}\,\left(\mathbf{S}\,\mathbf{S}^{\mathsf{T}}\right)^{-1}\mathbf{U}^{\mathsf{T}}$$

- where $\left(\mathbf{S}\,\mathbf{S}^{\mathsf{T}}\right)^{-1} = \mathrm{diag}(s_1^{-2}, s_2^{-2}, \cdots, s_N^{-2})$
- Thus

$$\begin{aligned}
\left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}\right)^{-1}\boldsymbol{\Phi} &= \left(\mathbf{U}\,\left(\mathbf{S}\,\mathbf{S}^{\mathsf{T}}\right)^{-1}\mathbf{U}^{\mathsf{T}}\right)\left(\mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\mathsf{T}}\right) \\
&= \mathbf{U}\,\left(\mathbf{S}\,\mathbf{S}^{\mathsf{T}}\right)^{-1}\mathbf{S}\,\mathbf{V}^{\mathsf{T}} \\
&= \mathbf{U}\,\mathbf{S}^{+}\,\mathbf{V}^{\mathsf{T}}
\end{aligned}$$

where $\mathbf{S}^{+} = \left(\mathbf{S}\,\mathbf{S}^{\mathsf{T}}\right)^{-1}\mathbf{S}$

Pseudo-Inverse

- The **pseudo-inverse** is defined as $\mathbf{\Phi}^+ = \mathbf{U}\,\mathbf{S}^+\mathbf{V}^\mathsf{T}$ where

$$\mathbf{S}^+ = \left(\mathbf{S}\,\mathbf{S}^\mathsf{T}\right)^{-1}\mathbf{S} = \begin{pmatrix} s_1^{-1} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & s_N^{-1} & \cdots & 0 \end{pmatrix}$$

- Thus the solution to the least squares problem is

$$\boldsymbol{w}^* = \left(\mathbf{\Phi}\,\mathbf{\Phi}^\mathsf{T}\right)^{-1}\mathbf{\Phi}\boldsymbol{y} = \mathbf{\Phi}^+\boldsymbol{y} = \mathbf{U}\,\mathbf{S}^+\mathbf{V}^\mathsf{T}\boldsymbol{y}$$

- For non-square matrices Matlab uses the pseudo-inverse so in Matlab we can write

```
w = Psi\t
```

Pseudo-Inverse

- The **pseudo-inverse** is defined as $\mathbf{\Phi}^+ = \mathbf{U}\, \boldsymbol{S}^+ \mathbf{V}^\mathsf{T}$ where

$$\boldsymbol{S}^+ = \left(\boldsymbol{S}\, \boldsymbol{S}^\mathsf{T}\right)^{-1} \boldsymbol{S} = \begin{pmatrix} s_1^{-1} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & s_N^{-1} & \cdots & 0 \end{pmatrix}$$

- Thus the solution to the least squares problem is

$$\boldsymbol{w}^* = \left(\mathbf{\Phi}\, \mathbf{\Phi}^\mathsf{T}\right)^{-1} \mathbf{\Phi} \boldsymbol{y} = \mathbf{\Phi}^+ \boldsymbol{y} = \mathbf{U}\, \boldsymbol{S}^+ \mathbf{V}^\mathsf{T} \boldsymbol{y}$$

- For non-square matrices Matlab uses the pseudo-inverse so in Matlab we can write

```
    w = Psi\t
```

Pseudo-Inverse

- The **pseudo-inverse** is defined as $\mathbf{\Phi}^+ = \mathbf{U}\,\boldsymbol{S}^+\mathbf{V}^\mathsf{T}$ where

$$\boldsymbol{S}^+ = \left(\boldsymbol{S}\,\boldsymbol{S}^\mathsf{T}\right)^{-1}\boldsymbol{S} = \begin{pmatrix} s_1^{-1} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & s_N^{-1} & \cdots & 0 \end{pmatrix}$$

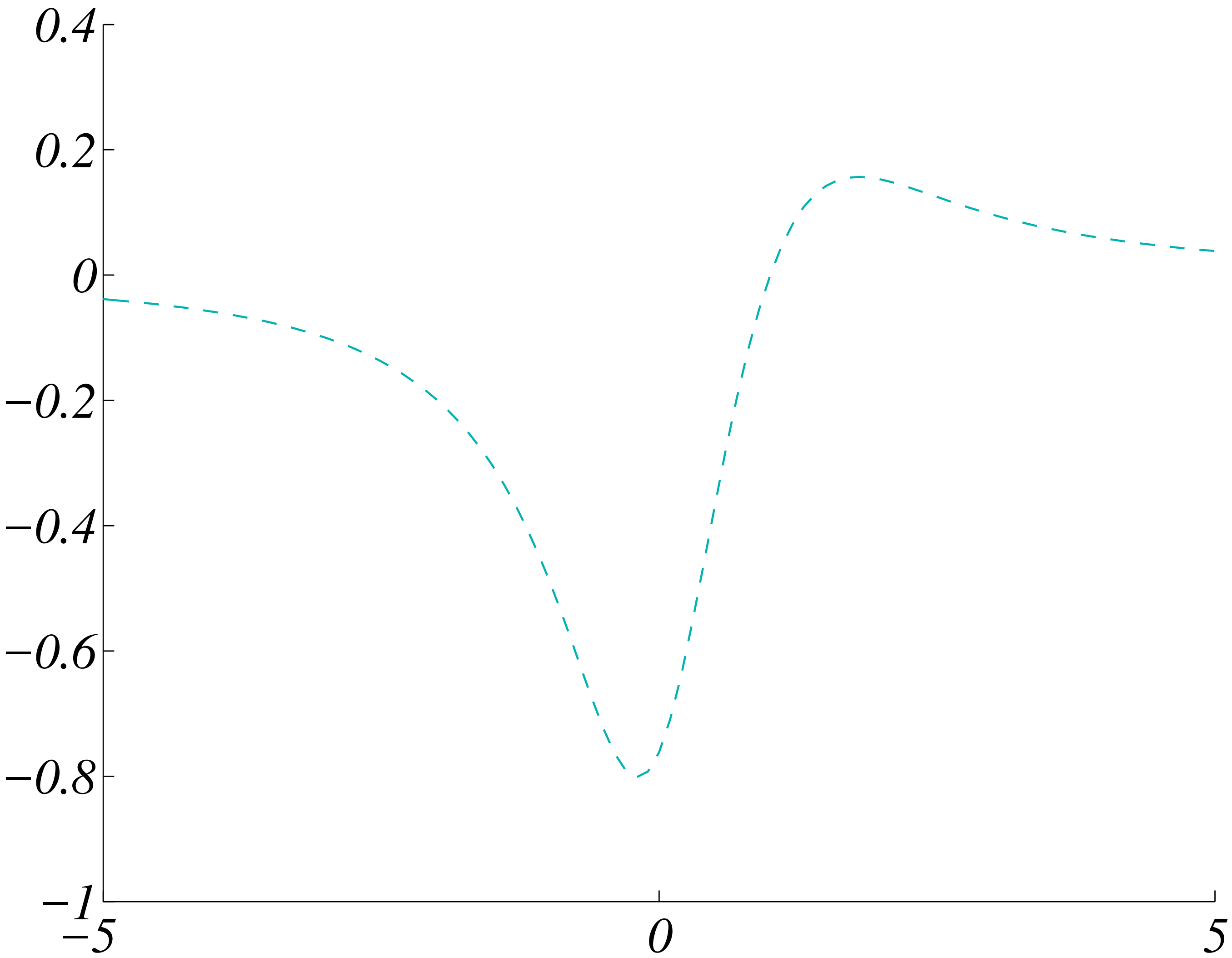
- Thus the solution to the least squares problem is

$$\boldsymbol{w}^* = \left(\mathbf{\Phi}\,\mathbf{\Phi}^\mathsf{T}\right)^{-1}\mathbf{\Phi}\boldsymbol{y} = \mathbf{\Phi}^+\boldsymbol{y} = \mathbf{U}\,\boldsymbol{S}^+\mathbf{V}^\mathsf{T}\boldsymbol{y}$$

- For non-square matrices Matlab uses the pseudo-inverse so in Matlab we can write

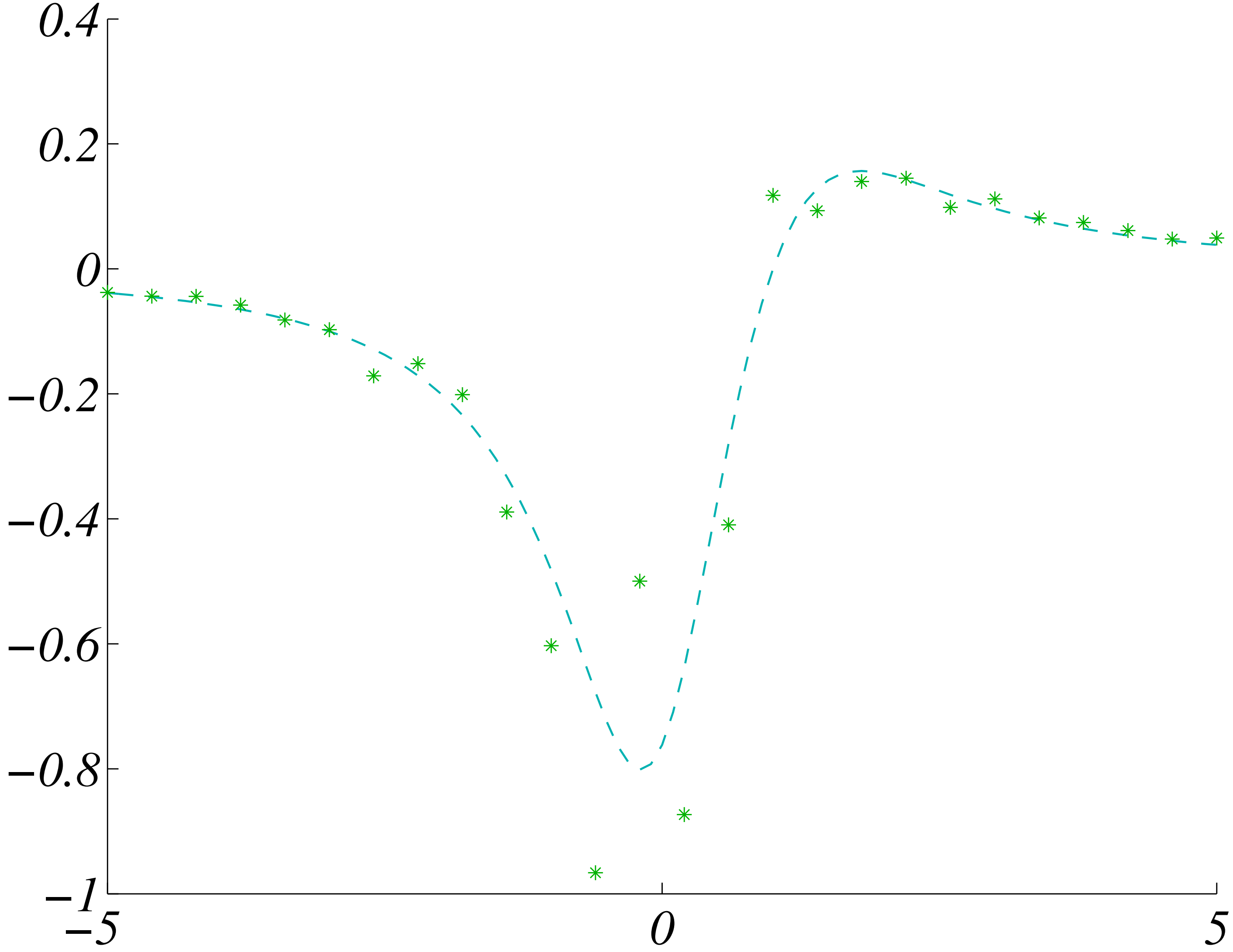
```
w = Psi\t
```

Example



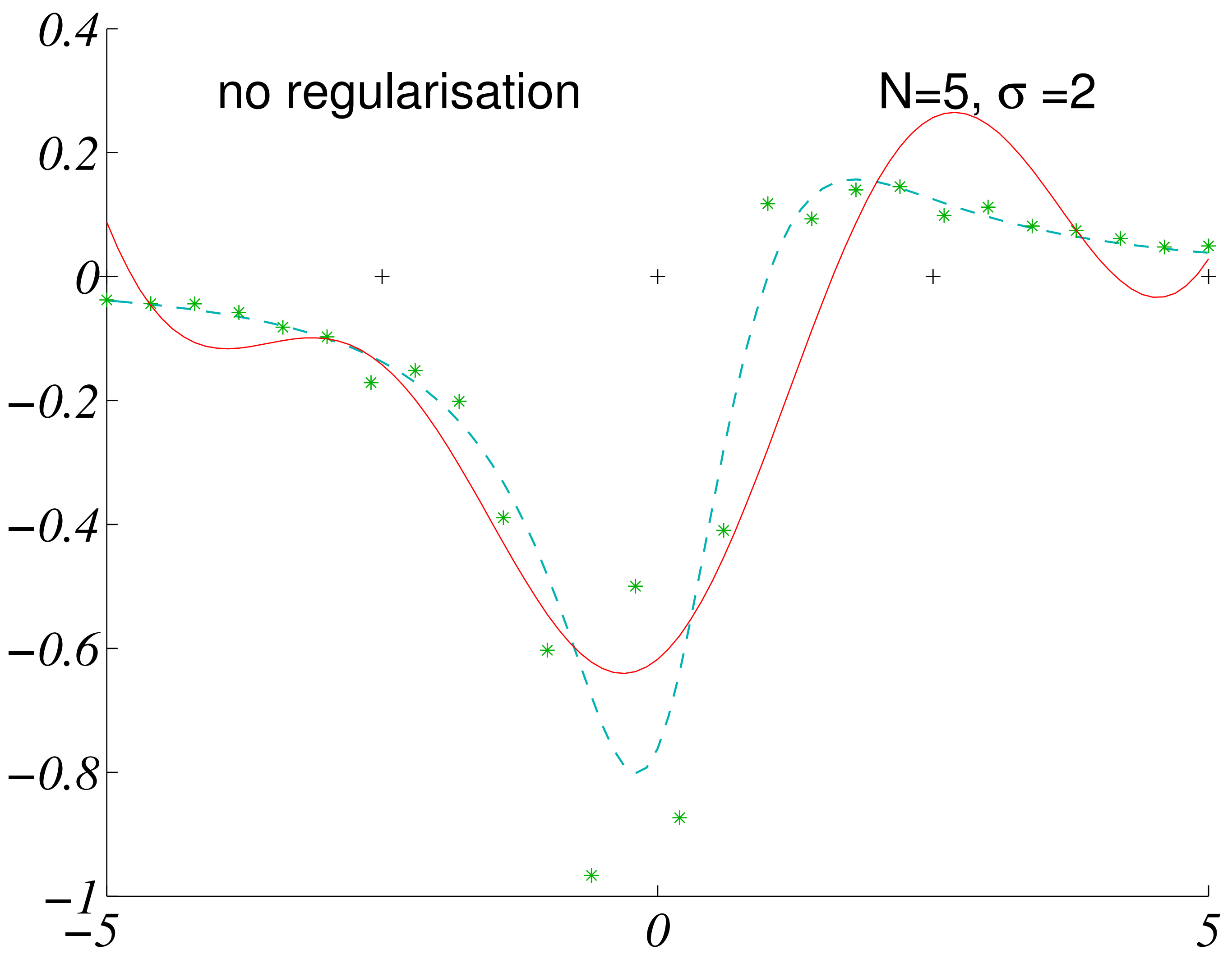
Function

Example



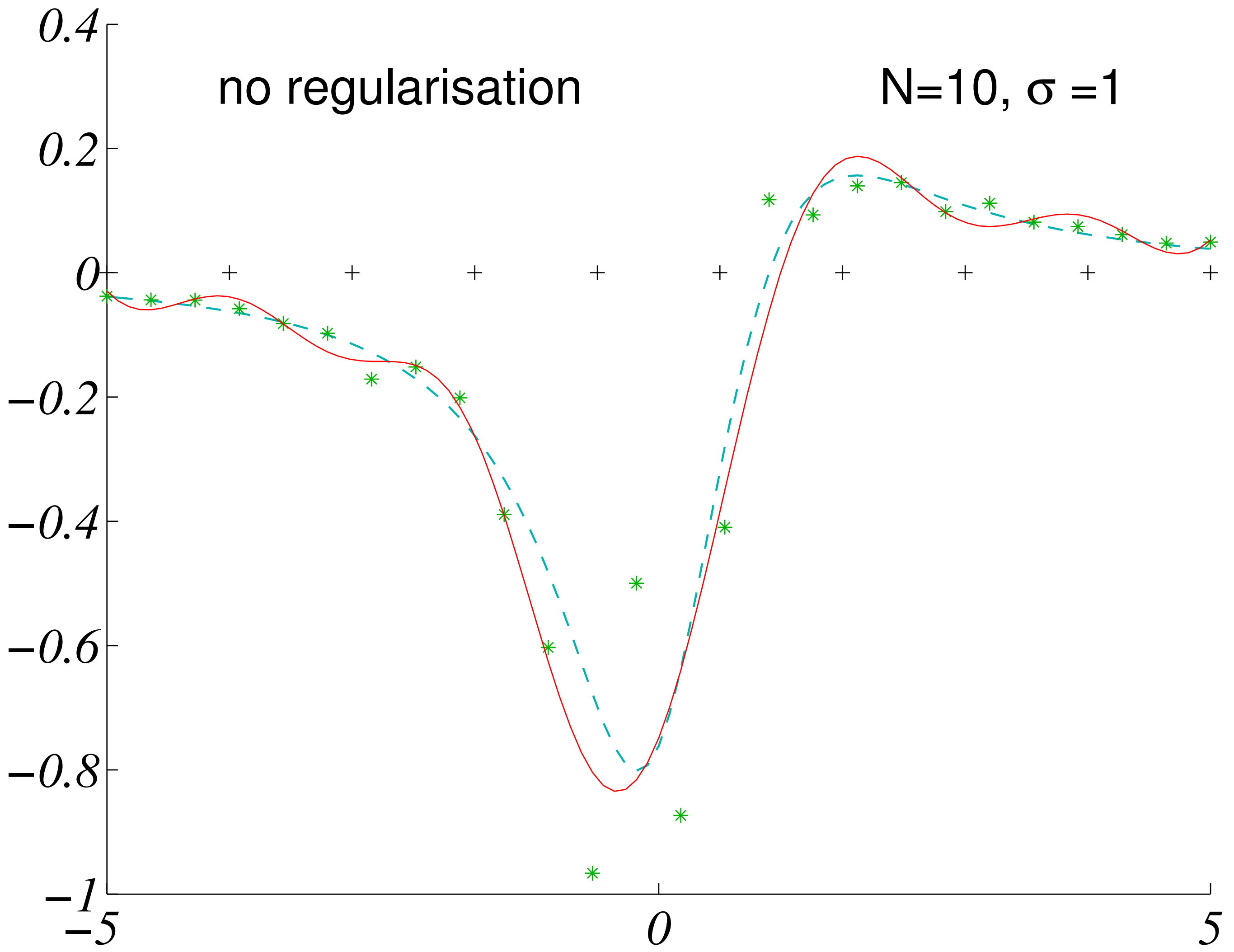
Data

Example



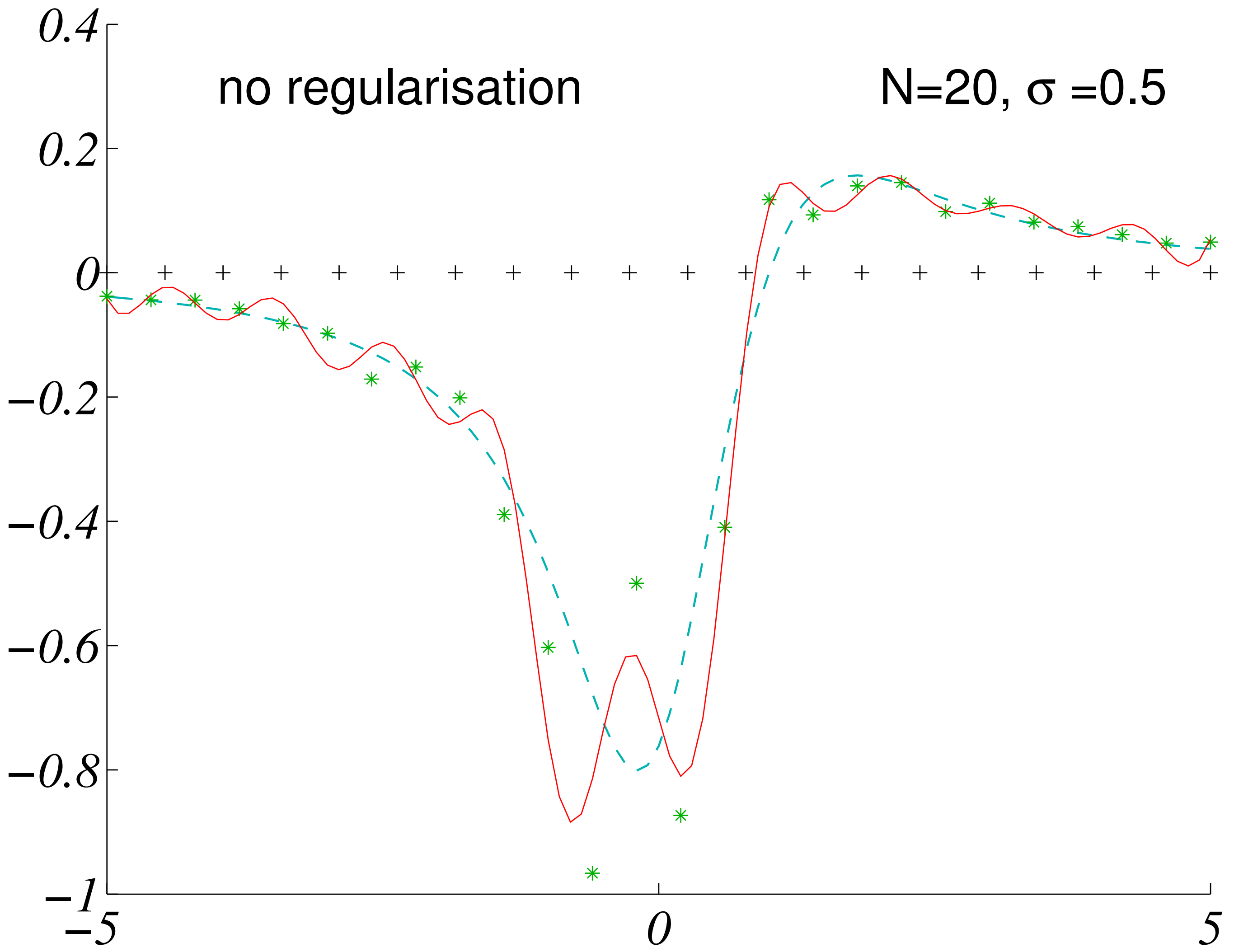
Centres too far apart

Example



Centres about right

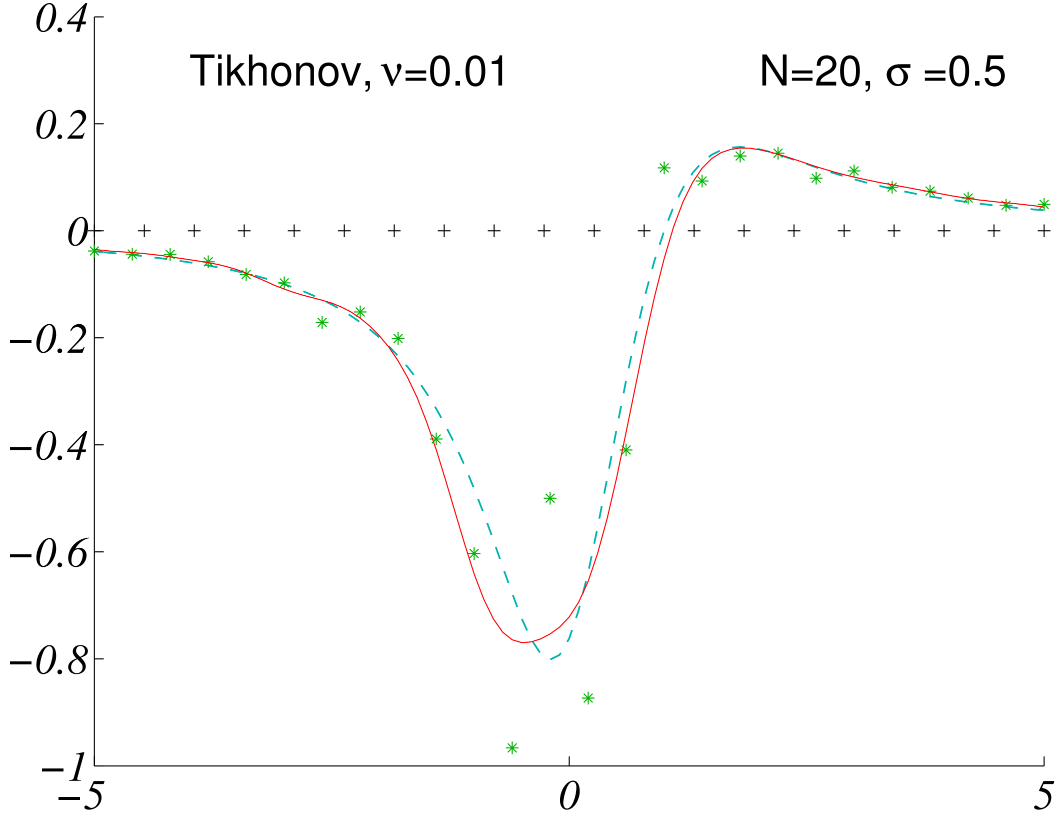
Example



Too many centres

Outline

1. Training Output Layer
2. **Regularisation**
   - Weight Decay
   - Tikhonov



Regularisation

- One way to improve generalisation performance is to bias a learning machine to learn simpler (smoother) functions
- This should reduce the sensitivity of the learning machine on the learning data
- To achieve this we can add **regularisation terms** that punishes complex functions

Regularisation

- One way to improve generalisation performance is to bias a learning machine to learn simpler (smoother) functions
- This should reduce the sensitivity of the learning machine on the learning data
- To achieve this we can add **regularisation terms** that punishes complex functions

Regularisation

- One way to improve generalisation performance is to bias a learning machine to learn simpler (smoother) functions
- This should reduce the sensitivity of the learning machine on the learning data
- To achieve this we can add **regularisation terms** that punishes complex functions

Regularisation Term

- We can add a regularisation term to force smoothness

$$E = \left\|\mathbf{\Phi}^\mathsf{T}\boldsymbol{w} - \boldsymbol{y}\right\|^2 + \nu R(\boldsymbol{w})$$

- Common regularisation terms are
  - ⋆ Weight decay $R(\boldsymbol{x}) = \|\boldsymbol{w}\|^2$
    - ∗ Easy to implement
    - ∗ *Ad hoc*
  - ⋆ Tikhonov regularisation $R(\boldsymbol{x}) = \int \sum_i \left(\frac{\partial^2 f(\boldsymbol{x};\boldsymbol{w})}{\partial x_i^2}\right)^2 p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$
    - ∗ Slightly more complicated to implement
    - ∗ Punishes rapid changes in gradient

Regularisation Term

- We can add a regularisation term to force smoothness

$$E = \left\| \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{w} - \boldsymbol{y} \right\|^2 + \nu R(\boldsymbol{w})$$

- Common regularisation terms are
  - ⋆ Weight decay $R(\boldsymbol{x}) = \|\boldsymbol{w}\|^2$
    - ∗ Easy to implement
    - ∗ *Ad hoc*
  - ⋆ Tikhonov regularisation $R(\boldsymbol{x}) = \int \sum_i \left(\frac{\partial^2 f(\boldsymbol{x};\boldsymbol{w})}{\partial x_i^2} \right)^2 p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$
    - ∗ Slightly more complicated to implement
    - ∗ Punishes rapid changes in gradient

Regularisation Term

- We can add a regularisation term to force smoothness

$$E = \left\| \mathbf{\Phi}^{\mathsf{T}} \boldsymbol{w} - \boldsymbol{y} \right\|^2 + \nu R(\boldsymbol{w})$$

- Common regularisation terms are
  - ⋆ Weight decay $R(\boldsymbol{x}) = \|\boldsymbol{w}\|^2$
    - ∗ Easy to implement
    - ∗ *Ad hoc*
  - ⋆ Tikhonov regularisation $R(\boldsymbol{x}) = \int \sum_i \left(\frac{\partial^2 f(\boldsymbol{x};\boldsymbol{w})}{\partial x_i^2} \right)^2 p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$
    - ∗ Slightly more complicated to implement
    - ∗ Punishes rapid changes in gradient

Weight Decay

- Define the 'error'

$$E = \left\| \mathbf{\Phi}^{\mathsf{T}} \boldsymbol{w} - \boldsymbol{y} \right\|^2 + \nu \left\| \boldsymbol{w} \right\|^2$$

- First term is proportional to means squared error
- Second term is **regularisation** term
- ν (Greek letter nu) controls balance between minimising the training error and preferring smooth solutions

Weight Decay

- Define the 'error'

$$E = \left\| \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{w} - \boldsymbol{y} \right\|^2 + \nu \left\| \boldsymbol{w} \right\|^2$$

- First term is proportional to means squared error
- Second term is **regularisation** term
- ν (Greek letter nu) controls balance between minimising the training error and preferring smooth solutions

Weight Decay

- Define the 'error'

$$E = \left\| \mathbf{\Phi}^{\mathsf{T}} \boldsymbol{w} - \boldsymbol{y} \right\|^2 + \nu \left\| \boldsymbol{w} \right\|^2$$

- First term is proportional to means squared error
- Second term is **regularisation** term
- ν (Greek letter nu) controls balance between minimising the training error and preferring smooth solutions

Weight Decay

- Define the 'error'

$$E = \left\| \mathbf{\Phi}^{\mathsf{T}} \boldsymbol{w} - \boldsymbol{y} \right\|^2 + \nu \left\| \boldsymbol{w} \right\|^2$$

- First term is proportional to means squared error
- Second term is **regularisation** term
- ν (Greek letter nu) controls balance between minimising the training error and preferring smooth solutions

Least Mean Squared Solution

- Regularised learning error

$$
\begin{aligned}
E &= \left\| \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{w} - \boldsymbol{y} \right\|^2 + \nu \left\| \boldsymbol{w} \right\|^2 \\
&= \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\Phi}\, \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{w} - 2 \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\Phi}\, \boldsymbol{y} + \boldsymbol{y}^{\mathsf{T}} \boldsymbol{y} + \nu\, \boldsymbol{w}^{\mathsf{T}} \boldsymbol{w} \\
&= \boldsymbol{w}^{\mathsf{T}} \left(\boldsymbol{\Phi} \boldsymbol{\Phi}^{\mathsf{T}} + \nu\, \mathbf{I} \right) \boldsymbol{w} - 2 \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\Phi}\, \boldsymbol{y} + \boldsymbol{y}^{\mathsf{T}} \boldsymbol{y}
\end{aligned}
$$

- Gradient

$$
\boldsymbol{\nabla} E = 2 \left(\boldsymbol{\Phi} \boldsymbol{\Phi}^{\mathsf{T}} + \nu\, \mathbf{I} \right) \boldsymbol{w} - 2\, \boldsymbol{\Phi}\, \boldsymbol{y}
$$

- Minimum $\boldsymbol{\nabla} E = 0$

$$
\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\, \boldsymbol{\Phi}^{\mathsf{T}} + \nu\, \mathbf{I} \right)^{-1} \boldsymbol{\Phi}\, \boldsymbol{y}
$$

Least Mean Squared Solution

- Regularised learning error

$$
\begin{aligned}
E &= \left\|\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{w}-\boldsymbol{y}\right\|^2+\nu\left\|\boldsymbol{w}\right\|^2\\
&= \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{w}-2\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Phi}\,\boldsymbol{y}+\boldsymbol{y}^{\mathsf{T}}\boldsymbol{y}+\nu\,\boldsymbol{w}^{\mathsf{T}}\boldsymbol{w}\\
&= \boldsymbol{w}^{\mathsf{T}}\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathsf{T}}+\nu\,\mathbf{I}\right)\boldsymbol{w}-2\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\Phi}\,\boldsymbol{y}+\boldsymbol{y}^{\mathsf{T}}\boldsymbol{y}
\end{aligned}
$$

- Gradient

$$
\boldsymbol{\nabla}E=2\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathsf{T}}+\nu\,\mathbf{I}\right)\boldsymbol{w}-2\,\boldsymbol{\Phi}\,\boldsymbol{y}
$$

- Minimum $\boldsymbol{\nabla}E=0$

$$
\boldsymbol{w}^*=\left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}+\nu\,\mathbf{I}\right)^{-1}\boldsymbol{\Phi}\,\boldsymbol{y}
$$

Least Mean Squared Solution

- Regularised learning error

$$\begin{aligned} E &= \left\| \mathbf{\Phi}^{\mathsf{T}} \boldsymbol{w} - \boldsymbol{y} \right\|^2 + \nu \left\| \boldsymbol{w} \right\|^2 \\ &= \boldsymbol{w}^{\mathsf{T}} \mathbf{\Phi}\, \mathbf{\Phi}^{\mathsf{T}} \boldsymbol{w} - 2\boldsymbol{w}^{\mathsf{T}} \mathbf{\Phi}\, \boldsymbol{y} + \boldsymbol{y}^{\mathsf{T}} \boldsymbol{y} + \nu\, \boldsymbol{w}^{\mathsf{T}} \boldsymbol{w} \\ &= \boldsymbol{w}^{\mathsf{T}} \left(\mathbf{\Phi} \mathbf{\Phi}^{\mathsf{T}} + \nu\, \mathbf{I} \right) \boldsymbol{w} - 2\boldsymbol{w}^{\mathsf{T}} \mathbf{\Phi}\, \boldsymbol{y} + \boldsymbol{y}^{\mathsf{T}} \boldsymbol{y} \end{aligned}$$

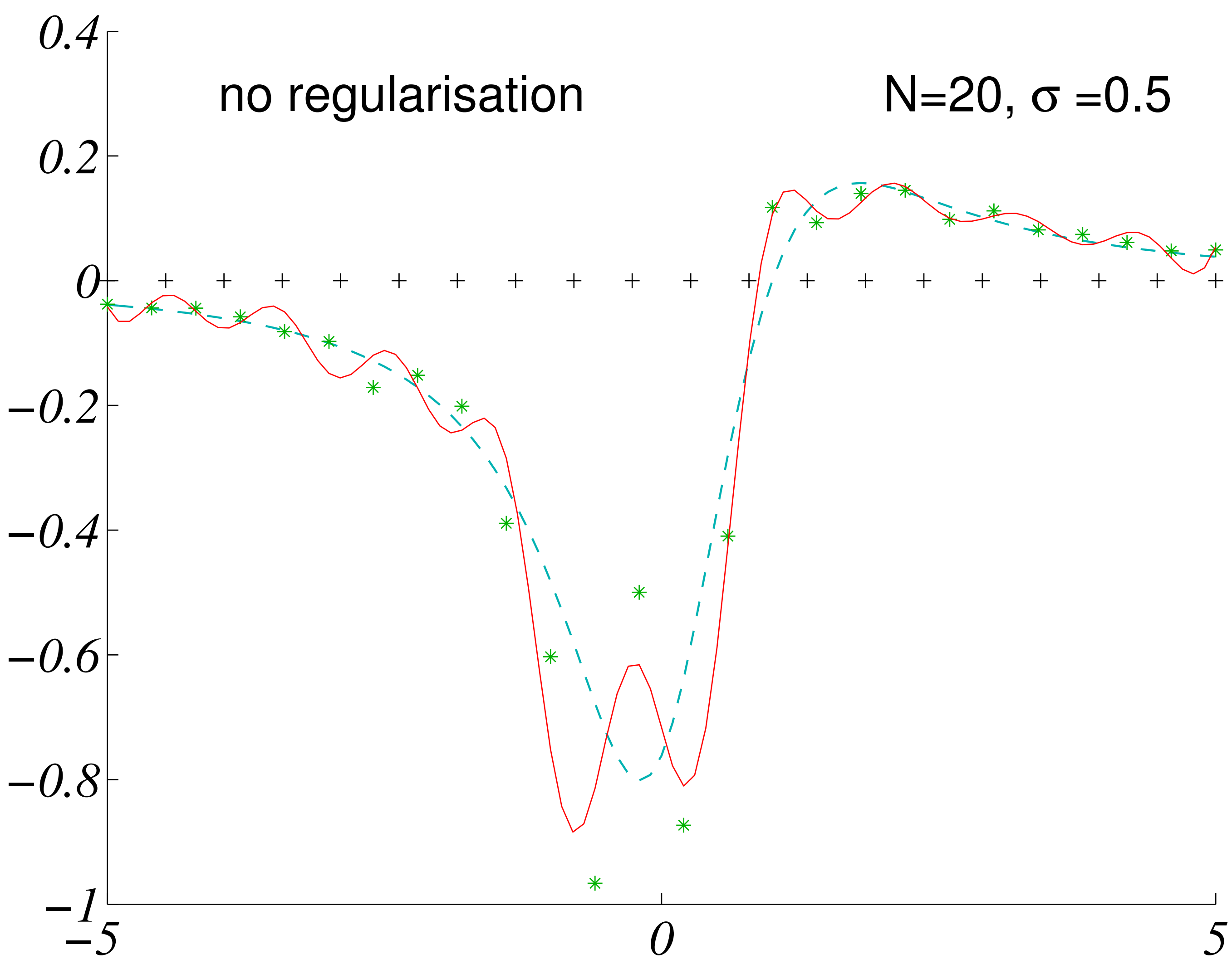
- Gradient

$$\boldsymbol{\nabla} E = 2 \left(\mathbf{\Phi} \mathbf{\Phi}^{\mathsf{T}} + \nu\, \mathbf{I} \right) \boldsymbol{w} - 2\, \mathbf{\Phi}\, \boldsymbol{y}$$

- Minimum $\boldsymbol{\nabla} E = 0$

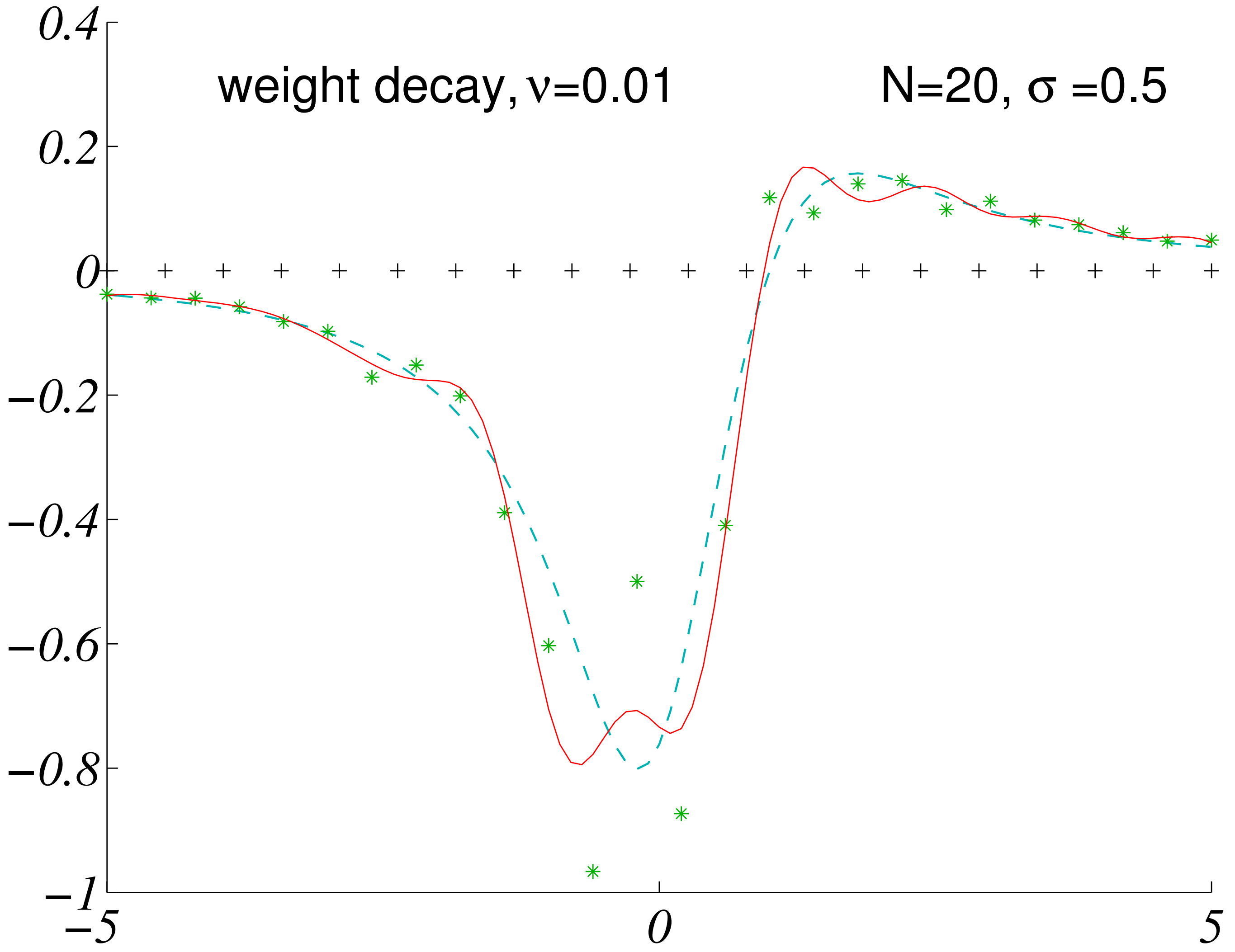
$$\boldsymbol{w}^* = \left(\mathbf{\Phi}\, \mathbf{\Phi}^{\mathsf{T}} + \nu\, \mathbf{I} \right)^{-1} \mathbf{\Phi}\, \boldsymbol{y}$$

Example



No regularisation

Example



Weight decay regularisation

Eigen Analysis

- Regularised least mean squared solution

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}} + \nu\,\mathbf{I}\right)^{-1}\boldsymbol{\Phi}\,\boldsymbol{y}$$

- Now $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ is a symmetric matrix which can be written

$$\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}} = \mathbf{U}\,\boldsymbol{\Lambda}\,\mathbf{U}^{\mathsf{T}}$$

- $\boldsymbol{\Lambda} = \boldsymbol{S}\,\boldsymbol{S}^{\mathsf{T}} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$, $\mathbf{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N)$
- $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ is positive semi-definite so $\lambda_i \geq 0$

Eigen Analysis

- Regularised least mean squared solution

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}} + \nu\,\mathbf{I}\right)^{-1} \boldsymbol{\Phi}\,\boldsymbol{y}$$

- Now $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ is a symmetric matrix which can be written

$$\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}} = \mathbf{U}\,\boldsymbol{\Lambda}\,\mathbf{U}^{\mathsf{T}}$$

- $\boldsymbol{\Lambda} = \boldsymbol{S}\,\boldsymbol{S}^{\mathsf{T}} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$, $\mathbf{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N)$
- $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ is positive semi-definite so $\lambda_i \geq 0$

Eigen Analysis

- Regularised least mean squared solution

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}} + \nu\,\mathbf{I}\right)^{-1}\boldsymbol{\Phi}\,\boldsymbol{y}$$

- Now $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ is a symmetric matrix which can be written

$$\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}} = \mathbf{U}\,\boldsymbol{\Lambda}\,\mathbf{U}^{\mathsf{T}}$$

- $\boldsymbol{\Lambda} = \boldsymbol{S}\,\boldsymbol{S}^{\mathsf{T}} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$, $\mathbf{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N)$
- $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ is positive semi-definite so $\lambda_i \geq 0$

Eigen Analysis

- Regularised least mean squared solution

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^\mathsf{T} + \nu\,\mathbf{I}\right)^{-1}\boldsymbol{\Phi}\,\boldsymbol{y}$$

- Now $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^\mathsf{T}$ is a symmetric matrix which can be written

$$\boldsymbol{\Phi}\,\boldsymbol{\Phi}^\mathsf{T} = \mathbf{U}\,\boldsymbol{\Lambda}\,\mathbf{U}^\mathsf{T}$$

- $\boldsymbol{\Lambda} = \boldsymbol{S}\,\boldsymbol{S}^\mathsf{T} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$, $\mathbf{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N)$
- $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^\mathsf{T}$ is positive semi-definite so $\lambda_i \geq 0$

Properties of Regularised Matrix

- Regularised matrix

$$
\begin{aligned}
\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}} + \nu\,\mathbf{I} &= \mathbf{U}\,(\mathbf{\Lambda} + \nu\mathbf{I})\,\mathbf{U}^{\mathsf{T}} \\
\left(\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}} + \nu\,\mathbf{I}\right)^{-1} &= \mathbf{U}\,(\mathbf{\Lambda} + \nu\,\mathbf{I})^{-1}\,\mathbf{U}^{\mathsf{T}}
\end{aligned}
$$

  where

$$
(\mathbf{\Lambda} + \nu\,\mathbf{I})^{-1} = \operatorname{diag}\left(\frac{1}{\lambda_1 + \nu}, \cdots, \frac{1}{\lambda_N + \nu}\right)
$$

- Regularisation makes inverse well posed if it wasn't
- Regularisation improves the conditioning (i.e. sensitive to changes in data)

$$
\frac{|\lambda_{max}|}{|\lambda_{min}|} \longrightarrow \frac{|\lambda_{max} + \nu|}{|\lambda_{min} + \nu|} < \frac{|\lambda_{max}|}{|\lambda_{min}|}
$$

Properties of Regularised Matrix

- Regularised matrix

$$
\begin{aligned}
\Phi\,\Phi^{\mathsf{T}} + \nu\,\mathbf{I} &= \mathbf{U}\left(\mathbf{\Lambda} + \nu\mathbf{I}\right)\mathbf{U}^{\mathsf{T}} \\
\left(\Phi\,\Phi^{\mathsf{T}} + \nu\,\mathbf{I}\right)^{-1} &= \mathbf{U}\left(\mathbf{\Lambda} + \nu\,\mathbf{I}\right)^{-1}\mathbf{U}^{\mathsf{T}}
\end{aligned}
$$

where

$$
\left(\mathbf{\Lambda} + \nu\,\mathbf{I}\right)^{-1} = \mathrm{diag}\left(\frac{1}{\lambda_1 + \nu}, \cdots, \frac{1}{\lambda_N + \nu}\right)
$$

- Regularisation makes inverse well posed if it wasn't
- Regularisation improves the conditioning (i.e. sensitive to changes in data)

$$
\frac{|\lambda_{max}|}{|\lambda_{min}|} \longrightarrow \frac{|\lambda_{max} + \nu|}{|\lambda_{min} + \nu|} < \frac{|\lambda_{max}|}{|\lambda_{min}|}
$$

Properties of Regularised Matrix

- Regularised matrix

$$\begin{aligned} \boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}} + \nu\,\mathbf{I} &= \mathbf{U}\,(\boldsymbol{\Lambda} + \nu\mathbf{I})\,\mathbf{U}^{\mathsf{T}} \\ \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}} + \nu\,\mathbf{I}\right)^{-1} &= \mathbf{U}\,(\boldsymbol{\Lambda} + \nu\,\mathbf{I})^{-1}\,\mathbf{U}^{\mathsf{T}} \end{aligned}$$

  where

$$(\boldsymbol{\Lambda} + \nu\,\mathbf{I})^{-1} = \mathrm{diag}\left(\frac{1}{\lambda_1 + \nu}, \cdots, \frac{1}{\lambda_N + \nu}\right)$$

- Regularisation makes inverse well posed if it wasn't

- Regularisation improves the conditioning (i.e. sensitive to changes in data)

$$\frac{|\lambda_{max}|}{|\lambda_{min}|} \longrightarrow \frac{|\lambda_{max} + \nu|}{|\lambda_{min} + \nu|} < \frac{|\lambda_{max}|}{|\lambda_{min}|}$$

Properties of Regularised Matrix

- Regularised matrix

$$\begin{aligned}
\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}} + \nu\,\mathbf{I} &= \mathbf{U}\,(\mathbf{\Lambda} + \nu \mathbf{I})\,\mathbf{U}^{\mathsf{T}} \\
\left(\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}} + \nu\,\mathbf{I}\right)^{-1} &= \mathbf{U}\,(\mathbf{\Lambda} + \nu\,\mathbf{I})^{-1}\,\mathbf{U}^{\mathsf{T}}
\end{aligned}$$

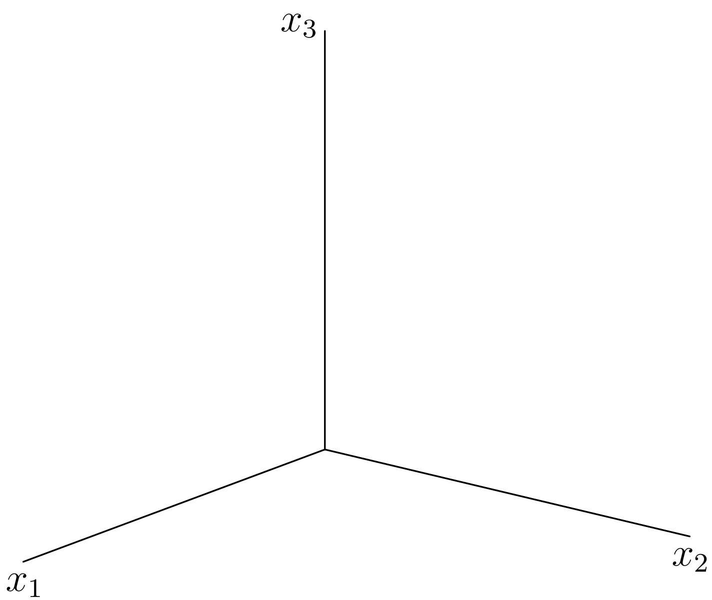
where

$$(\mathbf{\Lambda} + \nu\,\mathbf{I})^{-1} = \mathrm{diag}\left(\frac{1}{\lambda_1 + \nu}, \cdots, \frac{1}{\lambda_N + \nu}\right)$$

- Regularisation makes inverse well posed if it wasn't
- Regularisation improves the conditioning (i.e. sensitive to changes in data)

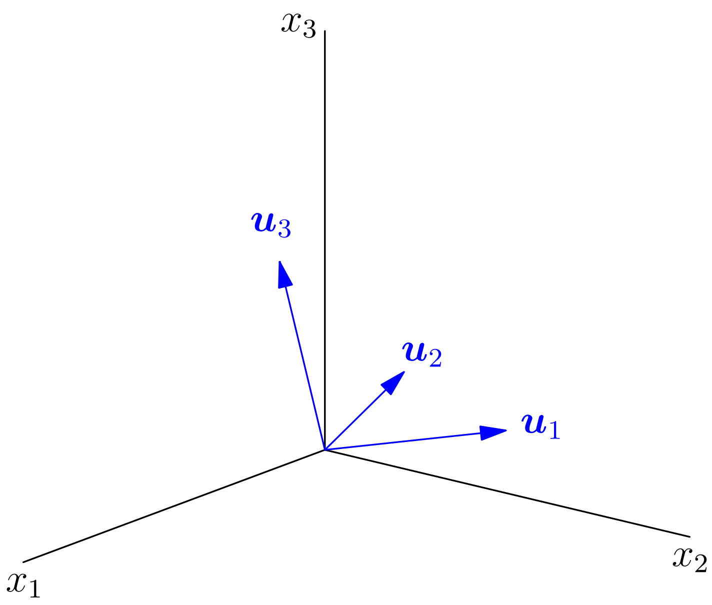
$$\frac{|\lambda_{max}|}{|\lambda_{min}|} \longrightarrow \frac{|\lambda_{max} + \nu|}{|\lambda_{min} + \nu|} < \frac{|\lambda_{max}|}{|\lambda_{min}|}$$

Ill-Conditioning



$$\boldsymbol{M} = \begin{pmatrix} 0.80 & 0.01 & 0.25 \\ 0.01 & 0.85 & 0.48 \\ 0.25 & 0.48 & 0.5 \end{pmatrix}$$
$$= \boldsymbol{U} \begin{pmatrix} 0.8 & 0 & 0 \\ 0 & 1.25 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \boldsymbol{U}^\mathsf{T}$$

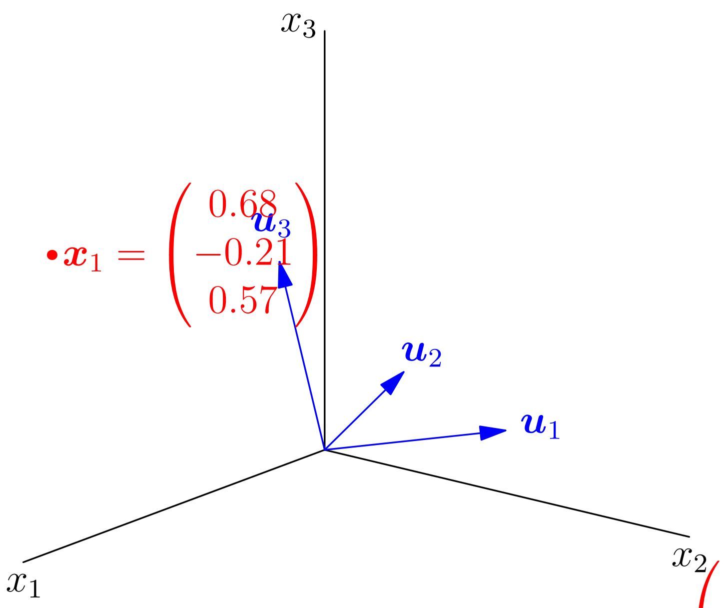
Ill-Conditioning



$$
\begin{aligned}
\boldsymbol{M} &= \begin{pmatrix} 0.80 & 0.01 & 0.25 \\ 0.01 & 0.85 & 0.48 \\ 0.25 & 0.48 & 0.5 \end{pmatrix} \\
&= \boldsymbol{U} \begin{pmatrix} 0.8 & 0 & 0 \\ 0 & 1.25 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \boldsymbol{U}^{\mathsf{T}}
\end{aligned}
$$

Ill-Conditioning

$$\boldsymbol{M} = \begin{pmatrix} 0.80 & 0.01 & 0.25 \\ 0.01 & 0.85 & 0.48 \\ 0.25 & 0.48 & 0.5 \end{pmatrix} = \boldsymbol{U} \begin{pmatrix} 0.8 & 0 & 0 \\ 0 & 1.25 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \boldsymbol{U}^\mathsf{T}$$



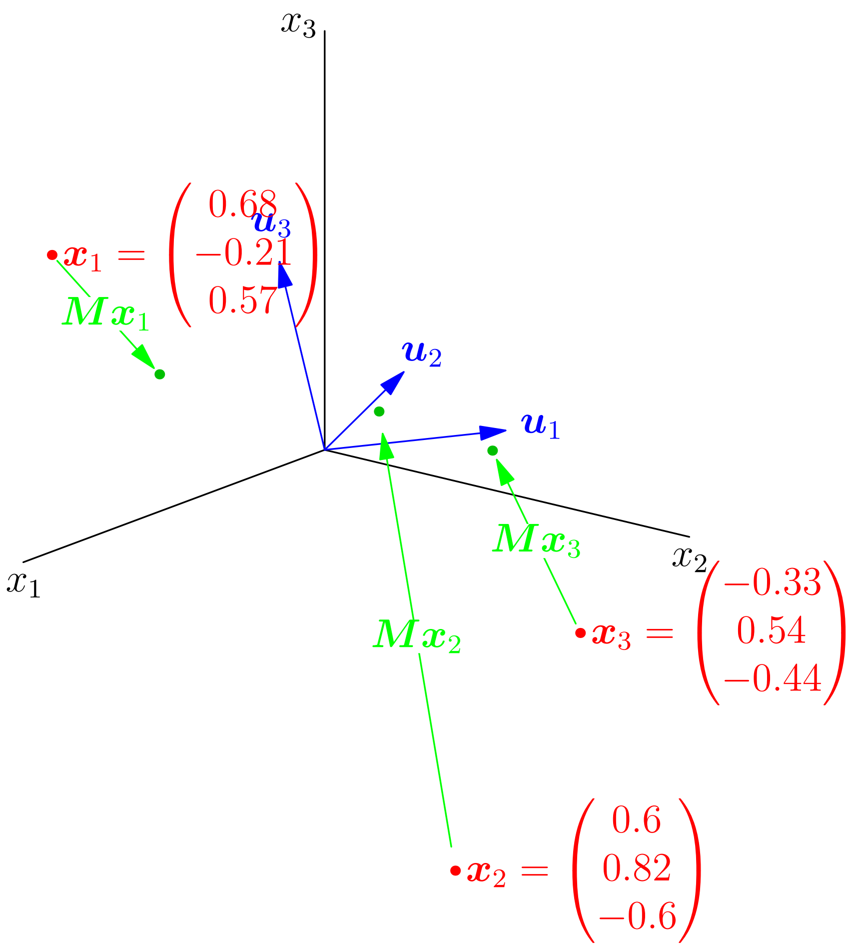
$$\bullet\, \boldsymbol{x}_1 = \begin{pmatrix} 0.68 \\ -0.21 \\ 0.57 \end{pmatrix}$$

$$\bullet\, \boldsymbol{x}_3 = \begin{pmatrix} -0.33 \\ 0.54 \\ -0.44 \end{pmatrix}$$

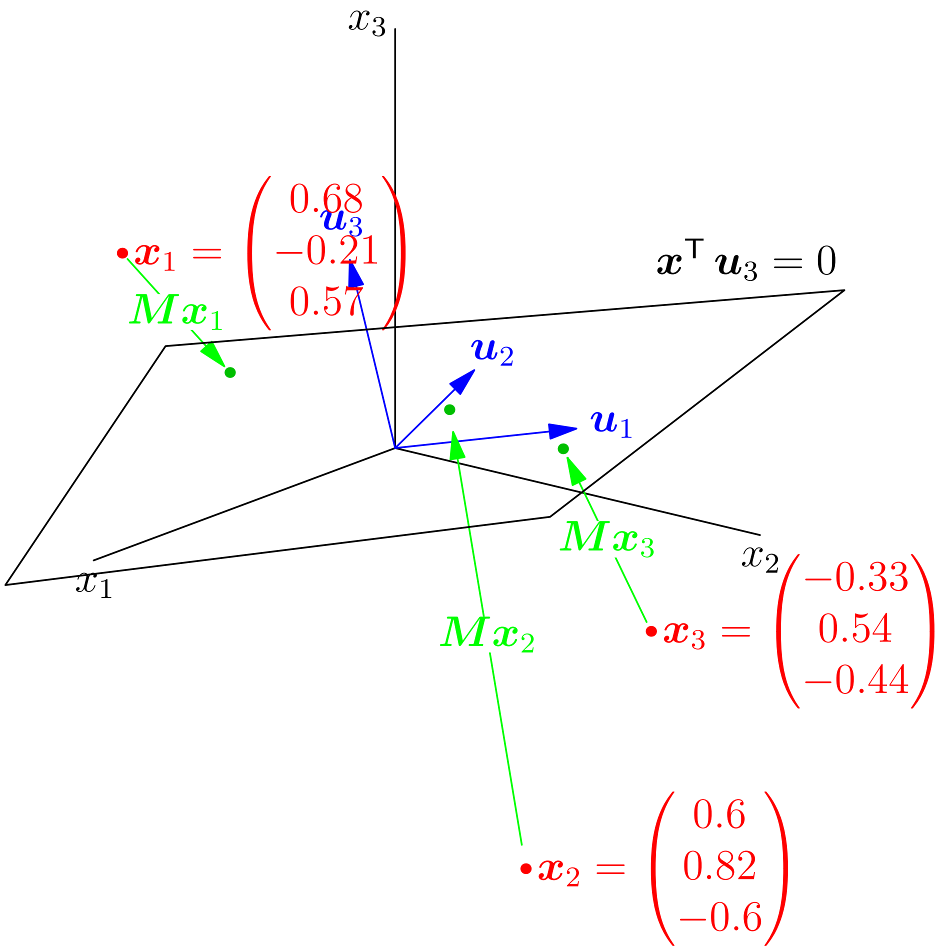
$$\bullet\, \boldsymbol{x}_2 = \begin{pmatrix} 0.6 \\ 0.82 \\ -0.6 \end{pmatrix}$$

Ill-Conditioning

$$\boldsymbol{M} = \begin{pmatrix} 0.80 & 0.01 & 0.25 \\ 0.01 & 0.85 & 0.48 \\ 0.25 & 0.48 & 0.5 \end{pmatrix} = \boldsymbol{U} \begin{pmatrix} 0.8 & 0 & 0 \\ 0 & 1.25 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \boldsymbol{U}^{\mathsf{T}}$$

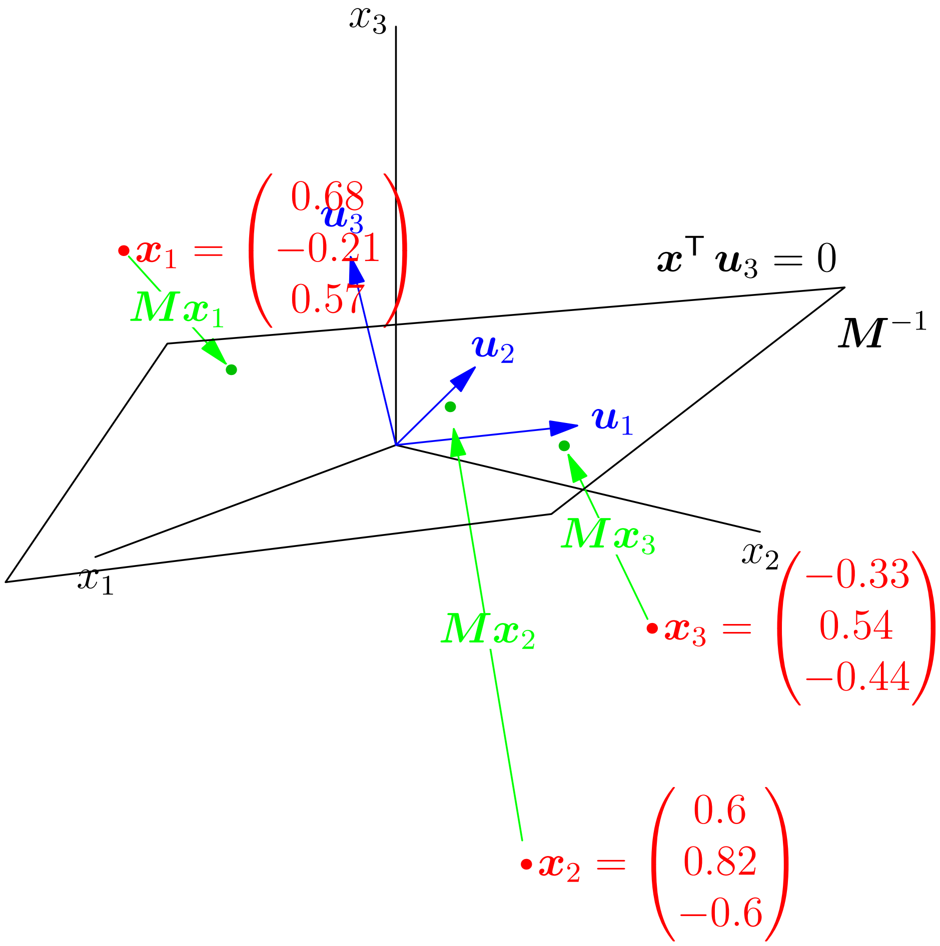


Ill-Conditioning



$$\boldsymbol{M} = \begin{pmatrix} 0.80 & 0.01 & 0.25 \\ 0.01 & 0.85 & 0.48 \\ 0.25 & 0.48 & 0.5 \end{pmatrix} = \boldsymbol{U} \begin{pmatrix} 0.8 & 0 & 0 \\ 0 & 1.25 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \boldsymbol{U}^\mathsf{T}$$

Ill-Conditioning



$$\boldsymbol{M} = \begin{pmatrix} 0.80 & 0.01 & 0.25 \\ 0.01 & 0.85 & 0.48 \\ 0.25 & 0.48 & 0.5 \end{pmatrix} = \boldsymbol{U} \begin{pmatrix} 0.8 & 0 & 0 \\ 0 & 1.25 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \boldsymbol{U}^\mathsf{T}$$

$$\boldsymbol{M}^{-1} = \begin{pmatrix} 1.9 & 1.17 & -2.1 \\ 1.17 & 3.34 & -3.8 \\ -2.1 & -3.8 & 6.8 \end{pmatrix} = \boldsymbol{U} \begin{pmatrix} 1.25 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 10 \end{pmatrix} \boldsymbol{U}^\mathsf{T}$$

What Does It Mean?

- Without regularisation the optimal weights are

$$\boldsymbol{w}^* = \left(\mathbf{\Phi}\,\mathbf{\Phi}^\top\right)^{-1}\mathbf{\Phi}\,\boldsymbol{y}$$

- The matrix $\mathbf{\Phi}\,\mathbf{\Phi}^\top$ depends on the data
- If $\mathbf{\Phi}\,\mathbf{\Phi}^\top$ is ill-conditioned the inverse is unreliable
- With a weight decay regulariser the optimal weights are

$$\boldsymbol{w}^* = \left(\mathbf{\Phi}\,\mathbf{\Phi}^\top + \nu\,\mathbf{I}\right)^{-1}\mathbf{\Phi}\,\boldsymbol{y}$$

- These don't fit the data quite so well, but they are less sensitive to the data

What Does It Mean?

- Without regularisation the optimal weights are

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^\mathsf{T}\right)^{-1} \boldsymbol{\Phi}\,\boldsymbol{y}$$

- The matrix $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^\mathsf{T}$ depends on the data

- If $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^\mathsf{T}$ is ill-conditioned the inverse is unreliable

- With a weight decay regulariser the optimal weights are

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^\mathsf{T} + \nu\,\mathbf{I}\right)^{-1} \boldsymbol{\Phi}\,\boldsymbol{y}$$

- These don't fit the data quite so well, but they are less sensitive to the data

What Does It Mean?

- Without regularisation the optimal weights are

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^\mathsf{T}\right)^{-1} \boldsymbol{\Phi}\,\boldsymbol{y}$$

- The matrix $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^\mathsf{T}$ depends on the data
- If $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^\mathsf{T}$ is ill-conditioned the inverse is unreliable
- With a weight decay regulariser the optimal weights are

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^\mathsf{T} + \nu\,\mathbf{I}\right)^{-1} \boldsymbol{\Phi}\,\boldsymbol{y}$$

- These don't fit the data quite so well, but they are less sensitive to the data

What Does It Mean?

- Without regularisation the optimal weights are

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}\right)^{-1} \boldsymbol{\Phi}\,\boldsymbol{y}$$

- The matrix $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ depends on the data
- If $\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}}$ is ill-conditioned the inverse is unreliable
- With a weight decay regulariser the optimal weights are

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\,\boldsymbol{\Phi}^{\mathsf{T}} + \nu\,\mathbf{I}\right)^{-1} \boldsymbol{\Phi}\,\boldsymbol{y}$$

- These don't fit the data quite so well, but they are less sensitive to the data

What Does It Mean?

- Without regularisation the optimal weights are

$$\boldsymbol{w}^* = \left(\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}}\right)^{-1}\mathbf{\Phi}\,\boldsymbol{y}$$

- The matrix $\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}}$ depends on the data
- If $\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}}$ is ill-conditioned the inverse is unreliable
- With a weight decay regulariser the optimal weights are

$$\boldsymbol{w}^* = \left(\mathbf{\Phi}\,\mathbf{\Phi}^{\mathsf{T}} + \nu\,\mathbf{I}\right)^{-1}\mathbf{\Phi}\,\boldsymbol{y}$$

- These don't fit the data quite so well, but they are less sensitive to the data

Tikhonov Regularisation

- We can use a more direct penalty for non-smooth functions

$$\int \sum_{i=1}^{N} \left(\frac{\partial^2 f(\boldsymbol{x}; \boldsymbol{w})}{\partial x_i^2} \right)^2 p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

- In practice we cannot compute this
- We can estimate this quantity by

$$\frac{1}{P} \sum_{p=1}^{P} \sum_{i=1}^{N} \left(\frac{\partial^2 f(\boldsymbol{x}_p; \boldsymbol{w})}{\partial x_i^2} \right)^2$$

- Average over all input patterns

Tikhonov Regularisation

- We can use a more direct penalty for non-smooth functions

$$\int \sum_{i=1}^{N} \left(\frac{\partial^2 f(\boldsymbol{x}; \boldsymbol{w})}{\partial x_i^2} \right)^2 p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

- In practice we cannot compute this

- We can estimate this quantity by

$$\frac{1}{P} \sum_{p=1}^{P} \sum_{i=1}^{N} \left(\frac{\partial^2 f(\boldsymbol{x}_p; \boldsymbol{w})}{\partial x_i^2} \right)^2$$

- Average over all input patterns

Tikhonov Regularisation

- We can use a more direct penalty for non-smooth functions

$$\int \sum_{i=1}^{N} \left(\frac{\partial^2 f(\boldsymbol{x}; \boldsymbol{w})}{\partial x_i^2} \right)^2 p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

- In practice we cannot compute this
- We can estimate this quantity by

$$\frac{1}{P} \sum_{p=1}^{P} \sum_{i=1}^{N} \left(\frac{\partial^2 f(\boldsymbol{x}_p; \boldsymbol{w})}{\partial x_i^2} \right)^2$$

- Average over all input patterns

Tikhonov Regularisation

- We can use a more direct penalty for non-smooth functions

$$\int \sum_{i=1}^{N} \left(\frac{\partial^2 f(\boldsymbol{x}; \boldsymbol{w})}{\partial x_i^2} \right)^2 p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

- In practice we cannot compute this
- We can estimate this quantity by

$$\frac{1}{P} \sum_{p=1}^{P} \sum_{i=1}^{N} \left(\frac{\partial^2 f(\boldsymbol{x}_p; \boldsymbol{w})}{\partial x_i^2} \right)^2$$

- Average over all input patterns

Tikhonov Regulariser

- Now, $f(\boldsymbol{x};\boldsymbol{w}) = \sum_k w_k \phi_k(\boldsymbol{x})$ so

$$\frac{\partial^2 f(\boldsymbol{x};\boldsymbol{w})}{\partial x_i^2} = \sum_k w_k \frac{\partial^2 \phi_k(\boldsymbol{x})}{\partial x_i^2}$$

- And

$$\frac{1}{P}\sum_{p=1}^{P}\sum_{i=1}^{N}\left(\frac{\partial^2 f(\boldsymbol{x}_p;\boldsymbol{w})}{\partial x_i^2}\right)^2 = \boldsymbol{w}^{\mathsf{T}}\,\mathbf{T}\,\boldsymbol{w}$$

- where

$$T_{jk} = \frac{1}{P}\sum_{p=1}^{P}\sum_{i=1}^{N}\frac{\partial^2 \phi_j(\boldsymbol{x}_p)}{\partial x_i^2}\,\frac{\partial^2 \phi_k(\boldsymbol{x}_p)}{\partial x_i^2}$$

Tikhonov Regulariser

- Now, $f(\boldsymbol{x};\boldsymbol{w}) = \sum_k w_k \phi_k(\boldsymbol{x})$ so

$$\frac{\partial^2 f(\boldsymbol{x};\boldsymbol{w})}{\partial x_i^2} = \sum_k w_k \frac{\partial^2 \phi_k(\boldsymbol{x})}{\partial x_i^2}$$

- And

$$\frac{1}{P}\sum_{p=1}^{P}\sum_{i=1}^{N}\left(\frac{\partial^2 f(\boldsymbol{x}_p;\boldsymbol{w})}{\partial x_i^2}\right)^2 = \boldsymbol{w}^{\mathsf{T}}\,\mathbf{T}\,\boldsymbol{w}$$

- where

$$T_{jk} = \frac{1}{P}\sum_{p=1}^{P}\sum_{i=1}^{N}\frac{\partial^2 \phi_j(\boldsymbol{x}_p)}{\partial x_i^2}\,\frac{\partial^2 \phi_k(\boldsymbol{x}_p)}{\partial x_i^2}$$

Tikhonov Regulariser

- For Gaussian RBFs, $\phi_k(\boldsymbol{x}) = \mathrm{e}^{-\|\boldsymbol{x}-\boldsymbol{\mu}\|^2/2\sigma^2}$

$$\frac{\partial^2 \phi_k(\boldsymbol{x})}{\partial x_i^2} = \left(\frac{(x_i - \mu_i)^2 - \sigma^2}{\sigma^4} \right) \phi_k(\boldsymbol{x})$$

- Minimise

$$E = \left\| \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{w} - \boldsymbol{y} \right\|^2 + \nu\, \boldsymbol{w}^{\mathsf{T}} \, \mathbf{T} \, \boldsymbol{w}$$

- $\boldsymbol{\nabla} E = 0$ gives

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\, \boldsymbol{\Phi}^{\mathsf{T}} + \nu\, \mathbf{T} \right)^{-1} \boldsymbol{\Phi}\, \boldsymbol{y}$$

Tikhonov Regulariser

- For Gaussian RBFs, $\phi_k(\boldsymbol{x}) = \mathrm{e}^{-\|\boldsymbol{x}-\boldsymbol{\mu}\|^2/2\sigma^2}$

$$\frac{\partial^2 \phi_k(\boldsymbol{x})}{\partial x_i^2} = \left(\frac{(x_i - \mu_i)^2 - \sigma^2}{\sigma^4} \right) \phi_k(\boldsymbol{x})$$

- Minimise

$$E = \left\| \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{w} - \boldsymbol{y} \right\|^2 + \nu\, \boldsymbol{w}^{\mathsf{T}}\, \mathbf{T}\, \boldsymbol{w}$$

- $\boldsymbol{\nabla} E = 0$ gives

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\, \boldsymbol{\Phi}^{\mathsf{T}} + \nu\, \mathbf{T} \right)^{-1} \boldsymbol{\Phi}\, \boldsymbol{y}$$

Tikhonov Regulariser

- For Gaussian RBFs, $\phi_k(\boldsymbol{x}) = \mathrm{e}^{-\|\boldsymbol{x}-\boldsymbol{\mu}\|^2/2\sigma^2}$

$$\frac{\partial^2 \phi_k(\boldsymbol{x})}{\partial x_i^2} = \left(\frac{(x_i - \mu_i)^2 - \sigma^2}{\sigma^4} \right) \phi_k(\boldsymbol{x})$$

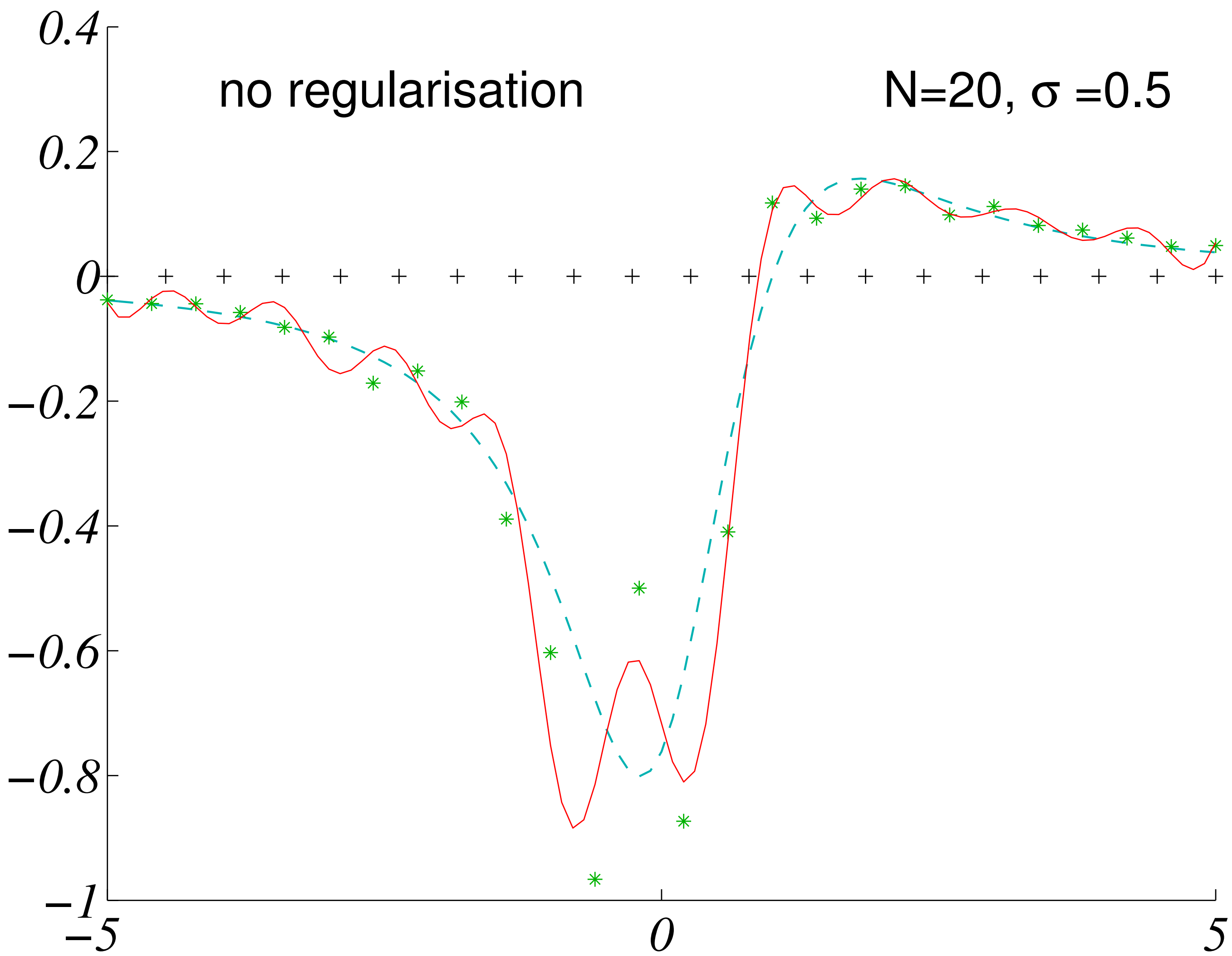
- Minimise

$$E = \left\| \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{w} - \boldsymbol{y} \right\|^2 + \nu\, \boldsymbol{w}^{\mathsf{T}} \mathbf{T}\, \boldsymbol{w}$$

- $\boldsymbol{\nabla} E = 0$ gives

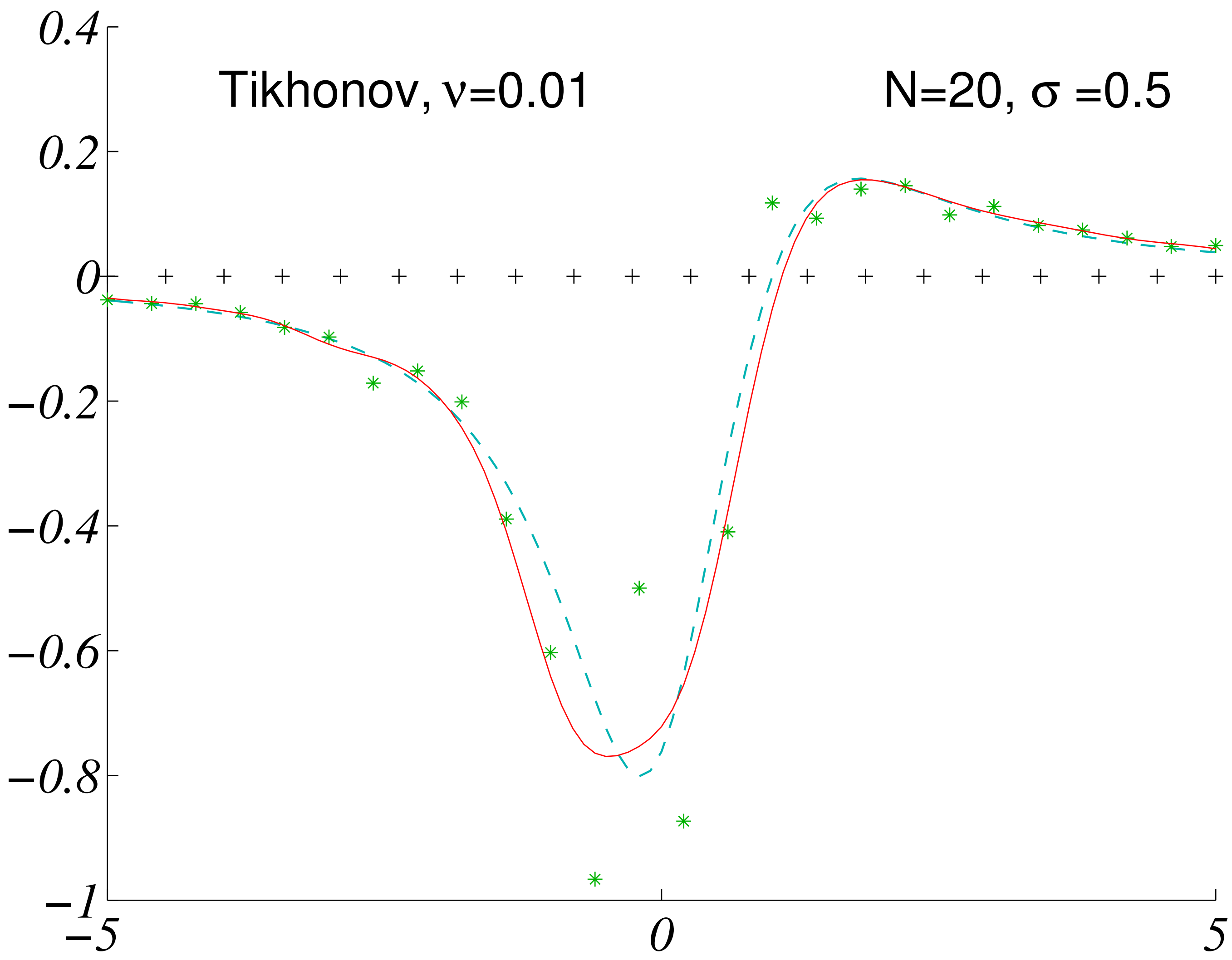
$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}\, \boldsymbol{\Phi}^{\mathsf{T}} + \nu\, \mathbf{T} \right)^{-1} \boldsymbol{\Phi}\, \boldsymbol{y}$$

Example



No regularisation

Example



Tikhonov Regularisation

Using Regularisers

- Regularisation terms are widely used—they really improve performance
- They can also be used in MLPs
- They prevent a learning machine with lots of parameters from **overfitting** the data
- They raise a new question "How do we choose the regularisation parameters, ν, etc."
- Usually, we use another set of unseen data to test for the best regularisation parameters

Using Regularisers

- Regularisation terms are widely used—they really improve performance
- They can also be used in MLPs
- They prevent a learning machine with lots of parameters from **overfitting** the data
- They raise a new question "How do we choose the regularisation parameters, ν, etc."
- Usually, we use another set of unseen data to test for the best regularisation parameters

Using Regularisers

- Regularisation terms are widely used—they really improve performance
- They can also be used in MLPs
- They prevent a learning machine with lots of parameters from **overfitting** the data
- They raise a new question "How do we choose the regularisation parameters, ν, etc."
- Usually, we use another set of unseen data to test for the best regularisation parameters

Using Regularisers

- Regularisation terms are widely used—they really improve performance
- They can also be used in MLPs
- They prevent a learning machine with lots of parameters from **overfitting** the data
- They raise a new question "How do we choose the regularisation parameters, ν, etc."
- Usually, we use another set of unseen data to test for the best regularisation parameters

Using Regularisers

- Regularisation terms are widely used—they really improve performance
- They can also be used in MLPs
- They prevent a learning machine with lots of parameters from **overfitting** the data
- They raise a new question "How do we choose the regularisation parameters, ν, etc."
- Usually, we use another set of unseen data to test for the best regularisation parameters

Summary

- In machine learning we care about performance on unseen data
- Over complicated machines will tend to learn the data set well, but will have poor generalisation performance
- We often wish to tailor the complexity of the machine to the data set
- One way to do this automatically is by introducing regularisation terms

Summary

- In machine learning we care about performance on unseen data
- Over complicated machines will tend to learn the data set well, but will have poor generalisation performance
- We often wish to tailor the complexity of the machine to the data set
- One way to do this automatically is by introducing regularisation terms

Summary

- In machine learning we care about performance on unseen data
- Over complicated machines will tend to learn the data set well, but will have poor generalisation performance
- We often wish to tailor the complexity of the machine to the data set
- One way to do this automatically is by introducing regularisation terms

Summary

- In machine learning we care about performance on unseen data
- Over complicated machines will tend to learn the data set well, but will have poor generalisation performance
- We often wish to tailor the complexity of the machine to the data set
- One way to do this automatically is by introducing regularisation terms