THE LINEAR MODELS

Risman Adnan Mattotorang

rismanadnan@telkomuniversity.ac.id

OUTLINE

- LINEAR MODEL FOR REGRESSION
- LINEAR MODEL FOR CLASSIFICATION
- LINEAR MODELS IN SCIKIT-LEARN
- WEEK 2 HOMEWORK

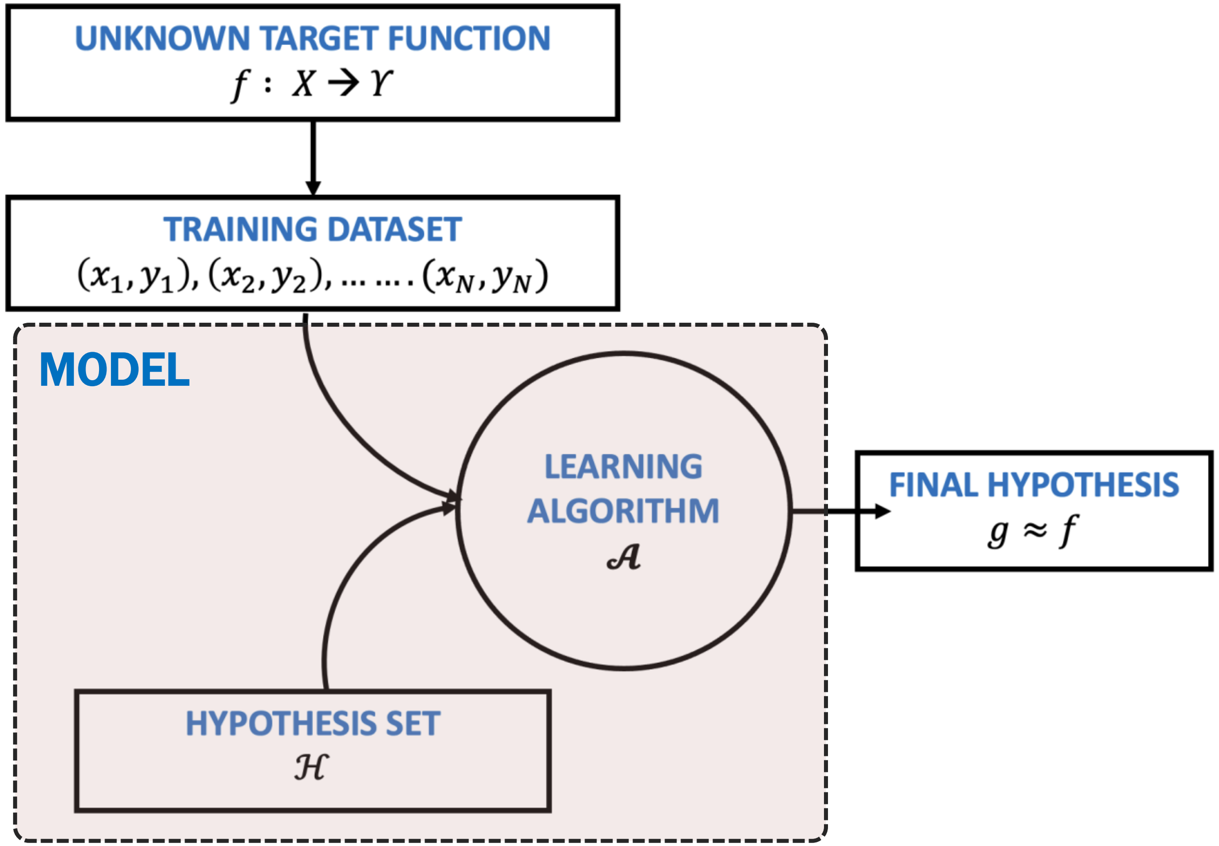
SUPERVISED LEARNING

The **data** that we can learn from.

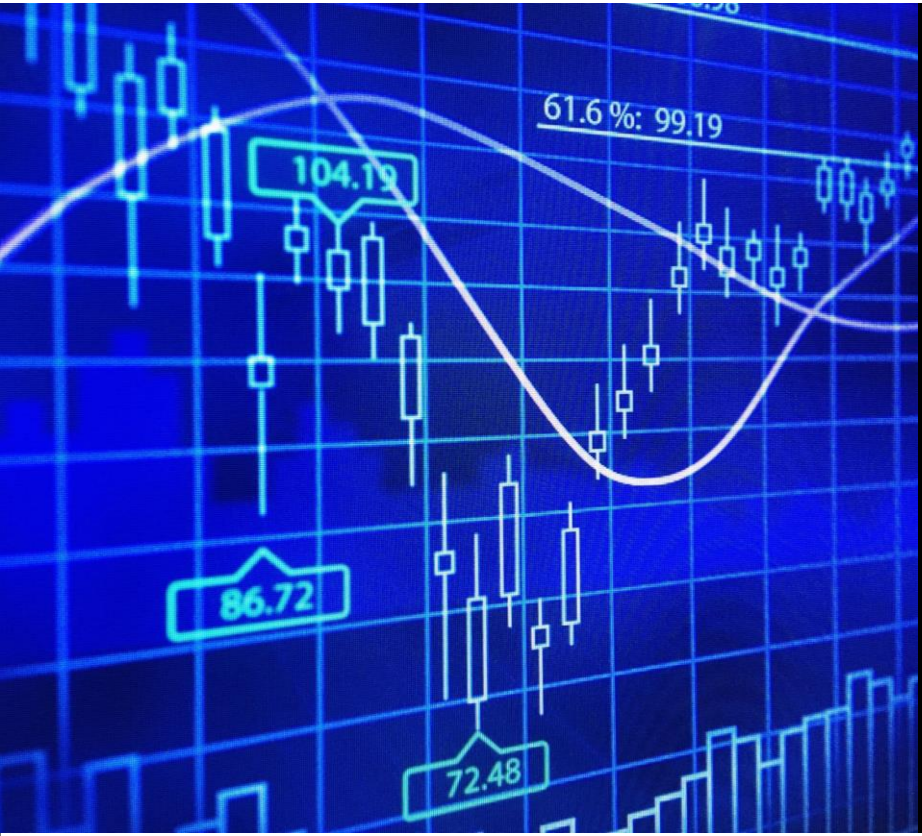
A **model** of how to transform the data.

An **objective function** that quantifies how well (or badly) the model is doing.

An **algorithm** to adjust the model's parameters to optimize the objective function.



LINEAR MODEL FOR REGRESSION

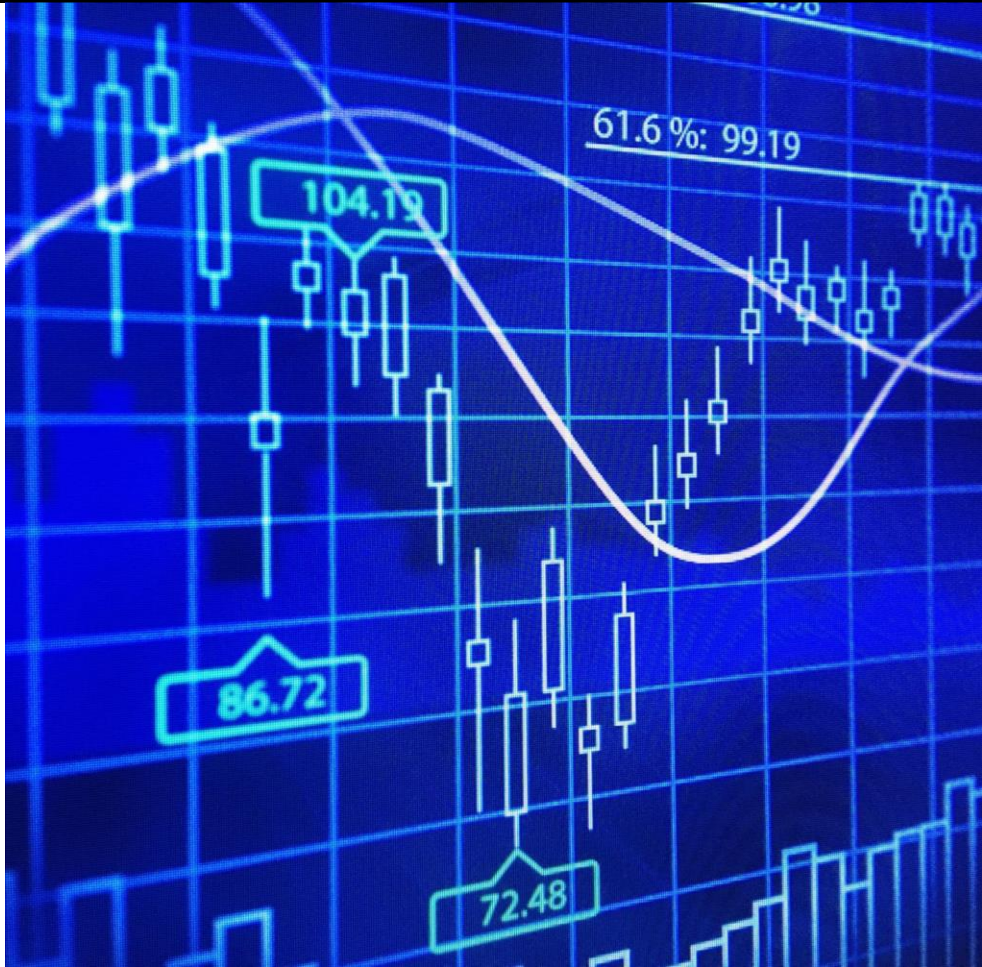


Supervised learning: N observations $\{x_n\}$ with corresponding target values $\{t_n\}$ are provided. The goal is to predict the continuous target t of a new value x.

- The simplest approach: construct a function such that $y(x)$ is a prediction of t.
- Probabilistic perspective: model the predictive distribution $p(t|x)$.

LINEAR MODEL FOR REGRESSION

- Linear Basis Function Models
- The Basis-Variance Decomposition
- Bayesian Linear Regression
- Bayesian Model Comparison
- The Evidence Approximation
- Limitations of Fixed Basis Functions



$$y(\boldsymbol{x}, \boldsymbol{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x})$$

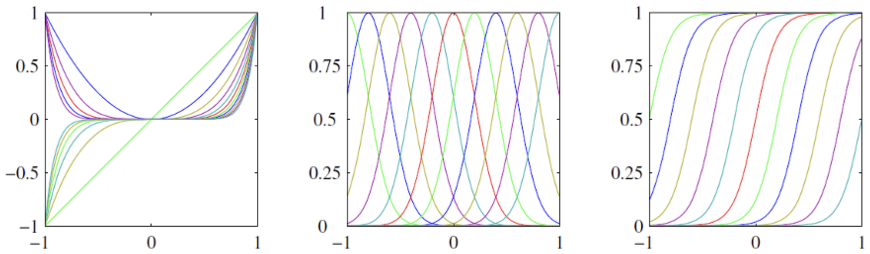
$\mathbf{w} = (w_0, \ldots, w_{M-1})^{\mathrm{T}}$ $\boldsymbol{\phi} = (\phi_0, \ldots, \phi_{M-1})^{\mathrm{T}}$ $\phi_0(\boldsymbol{x}) = 1$ w_0 a bias parameter

Basis function choices

Polynomial $\phi_j(x) = x^j$

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

Sigmoidal $\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$ with $\sigma(a) = \frac{1}{1+e^{-a}}$



LINEAR BASIS FUNCTIONS

$$t = \underbrace{y(\boldsymbol{x}, \boldsymbol{w})}_{\text{deterministic}} + \underbrace{\epsilon}_{\text{Gaussian noise}}$$

For a i.i.d. data set we have the likelihood function:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1})$$

We can use the machinery of MLE to estimate the parameters $\boldsymbol{w}$ and the precision $\boldsymbol{\beta}$:

logarithm of the likelihood

$$\begin{aligned} \ln p(\mathbf{t}|\mathbf{w}, \beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w}) \end{aligned}$$

sum-of-squares error

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

$$\boldsymbol{w}_{ML} = (\Phi^{\top}\Phi)^{-1}\Phi^{\top}\boldsymbol{t} \text{ with } \Phi_{M\times N} = [\phi_{mn}(\boldsymbol{x}_n)]$$

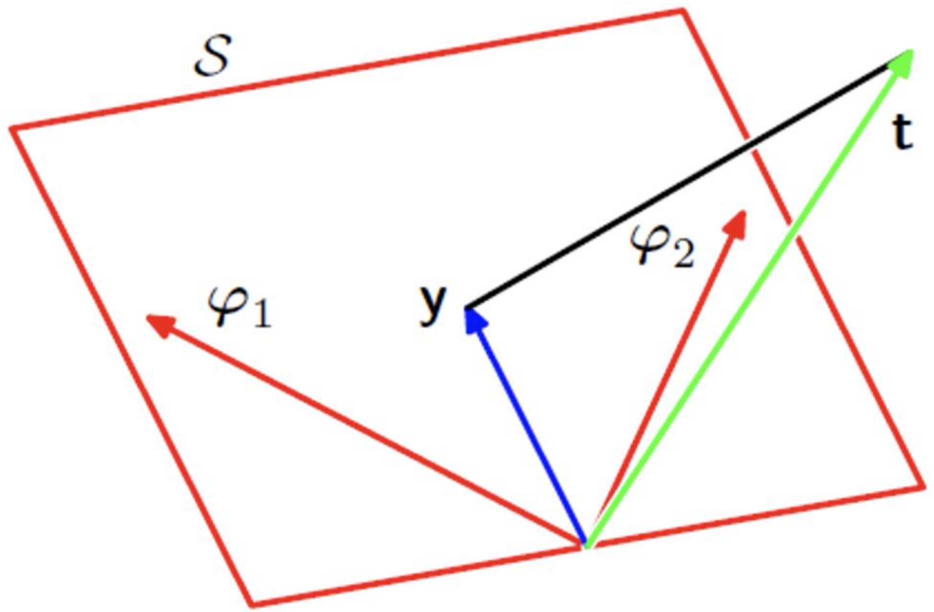
$$\beta_{ML}^{-1} = \frac{1}{N}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}_{ML}^{\top}\phi(\boldsymbol{x}_n)\right)^2$$

Maximization of the likelihood function under a conditional Gaussian noise distribution for a linear model is equivalent to minimizing a sum-of-squares error function given by $E_D(w)$

MLE LEAST SQUARE

Maximum Likelihood and Least Squares

Geometrical interpretation of the least-squares solution, in an N-dimensional space whose axes are the values of $t_1, \ldots, t_N$. The least-squares regression function is obtained by finding the orthogonal projection of the data vector **t** onto the subspace spanned by the basis functions $\phi_j(\mathbf{x})$ in which each basis function is viewed as a vector φ_j of length N with elements $\phi_j(\mathbf{x}_n)$.



GEOMETRIC INTERPRETATION

Sequential learning

Apply a technique known as stochastic gradient descent or sequential gradient descent, i.e.,

updates the parameter vector *w* using $\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n$

$$E_D(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \left(t_n - \boldsymbol{w}^\top \phi(\boldsymbol{x}_n)\right)^2 = \sum_n E_n$$

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} + \eta \underbrace{\left(t_n - \boldsymbol{w}^{(\tau)^\top} \phi(\boldsymbol{x}_n)\right)\phi(\boldsymbol{x}_n)}_{\nabla E_n}$$

η is a learning rate parameter

SEQUENTIAL LEARNING

Regularized least squares

Adding a regularization term to an error function in order to control over-fitting

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

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

weight decay

parameter shrinkage

$$\Longrightarrow \quad \mathbf{w} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}.$$

Regularization allows complex models to be trained on data sets of limited size without severe over-fitting, essentially by limiting the effective model complexity.
However, the problem of determining the optimal model complexity is then shifted from one of finding the appropriate number of basis functions to one of determining a suitable value of the regularization coefficient λ.

REGULARIZED LEAST SQUARES

- Over-fitting occurs whenever the number of basis functions is large and with training data sets of limited size.
- Limiting the number of basis functions limits the flexibility of the model.
- Regularization can control over-fitting but raises the question of how to determine λ.
- The *bias-variance* tradeoff is a frequentist viewpoint of model complexity.

Conditional expectation $$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x})\,\mathrm{d}t$$

Expected squared loss $$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2\, p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t.$$

$$\begin{aligned} &\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2\right] \\ &= \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right]}_{\text{variance}}. \end{aligned}$$

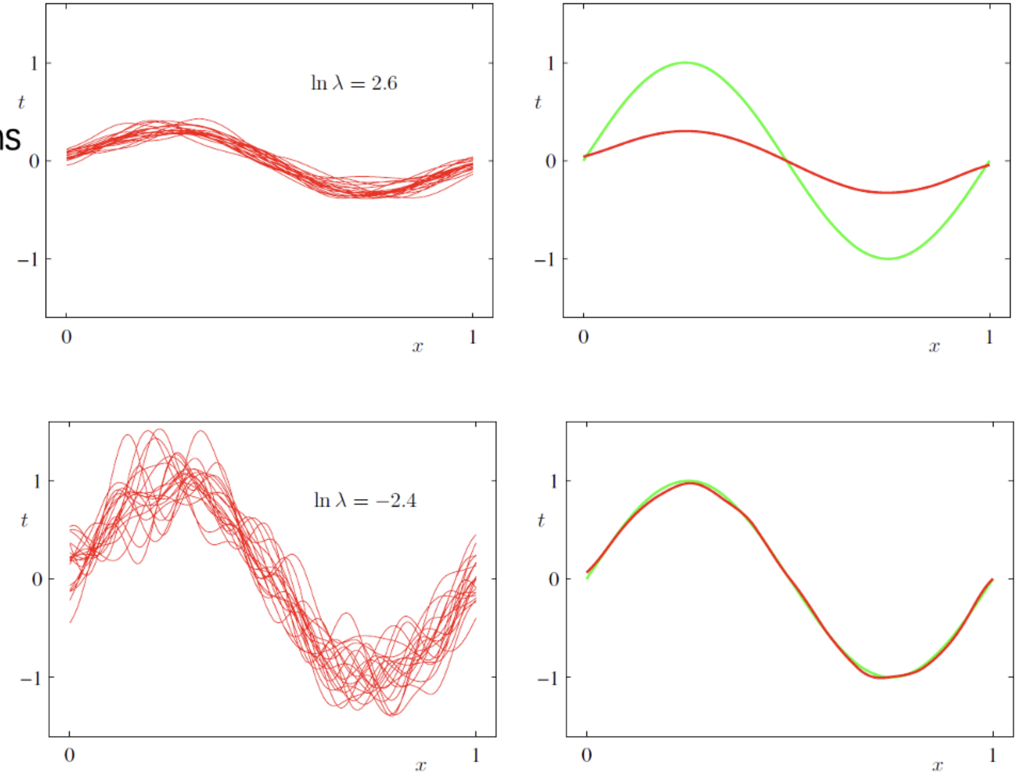
$$\begin{aligned} (\text{bias})^2 &= \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ \text{variance} &= \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ \text{noise} &= \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t \end{aligned}$$

StatQuest Here

BIAS VARIANCE DECOMPOSITION

100 data sets

24 Gaussian basis functions



high bias and low variance

low bias and high variance

BIAS VARIANCE DECOMPOSITION

OUTLINE

LINEAR MODEL FOR REGRESSION

LINEAR MODEL FOR CLASSIFICATION

LINEAR MODELS IN SCIKIT-LEARN

WEEK 2 HOMEWORK

Two classes $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$ Decision boundary: $y(\mathbf{x}) = 0$,

Multiple classes

one-versus-the-rest classifier: $K - 1$ classifiers

one-versus-one classifier: $K(K - 1)/2$ binary discriminant functions

single K-class discriminant: $y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$. $\Rightarrow$ $\mathbf{x} \in C_k$

Least squares for classification

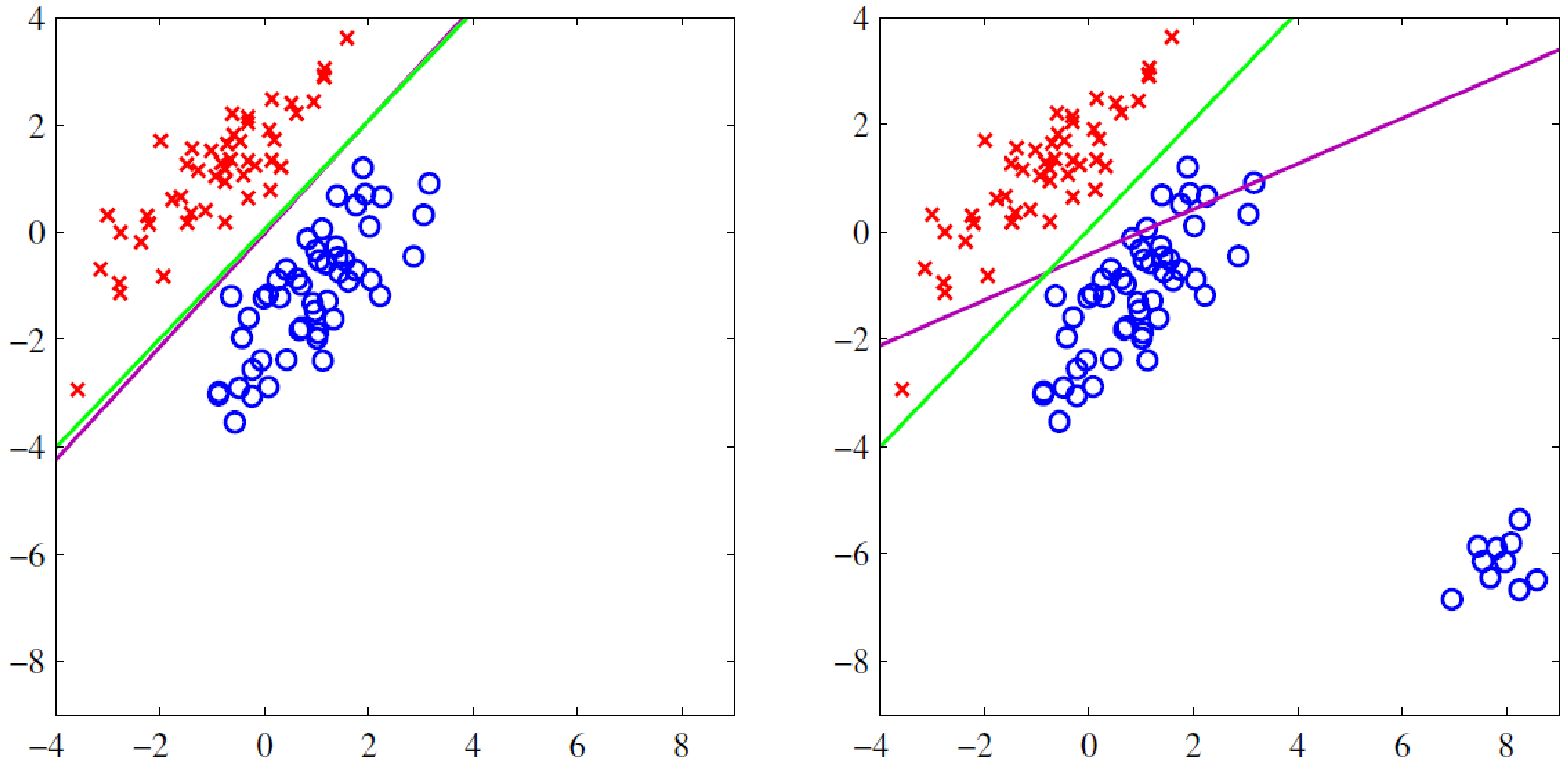
Each class $y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$ $\Rightarrow$ $\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}}$ $\widetilde{\mathbf{W}} = (\widetilde{\mathbf{w}}_1, \ldots, \widetilde{\mathbf{w}}_K)$ $\widetilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_k^{\mathrm{T}})^{\mathrm{T}}$

Training data $\{\mathbf{x}_n, \mathbf{t}_n\}$ $n = 1, \ldots, N$

Sum-of-squares error function $E_D(\widetilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\left\{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^{\mathrm{T}}(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})\right\}$ $\Rightarrow$ $\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^{\mathrm{T}}\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^{\mathrm{T}}\mathbf{T} = \widetilde{\mathbf{X}}^{\dagger}\mathbf{T}$

A new input $\mathbf{x} \in C_k$, if $y_k = \widetilde{\mathbf{w}}_k^{\mathrm{T}}\widetilde{\mathbf{x}}$ is largest.

DISCRIMINANT FUNCTIONS



LEAST SQUARES FOR CLASSIFICATION

select a projection that maximizes the class separation

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n \qquad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$$

maximize $m_2 - m_1 = \mathbf{w}^{\mathrm{T}}(\mathbf{m}_2 - \mathbf{m}_1)$ ➡ $\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)$ have considerable overlap

Fisher' s idea maximizes a function that will give a large separation between the projected class meanwhile also giving a small variance within each class, thereby minimizing the class overlap.

class C_k' s within-class variance $s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$

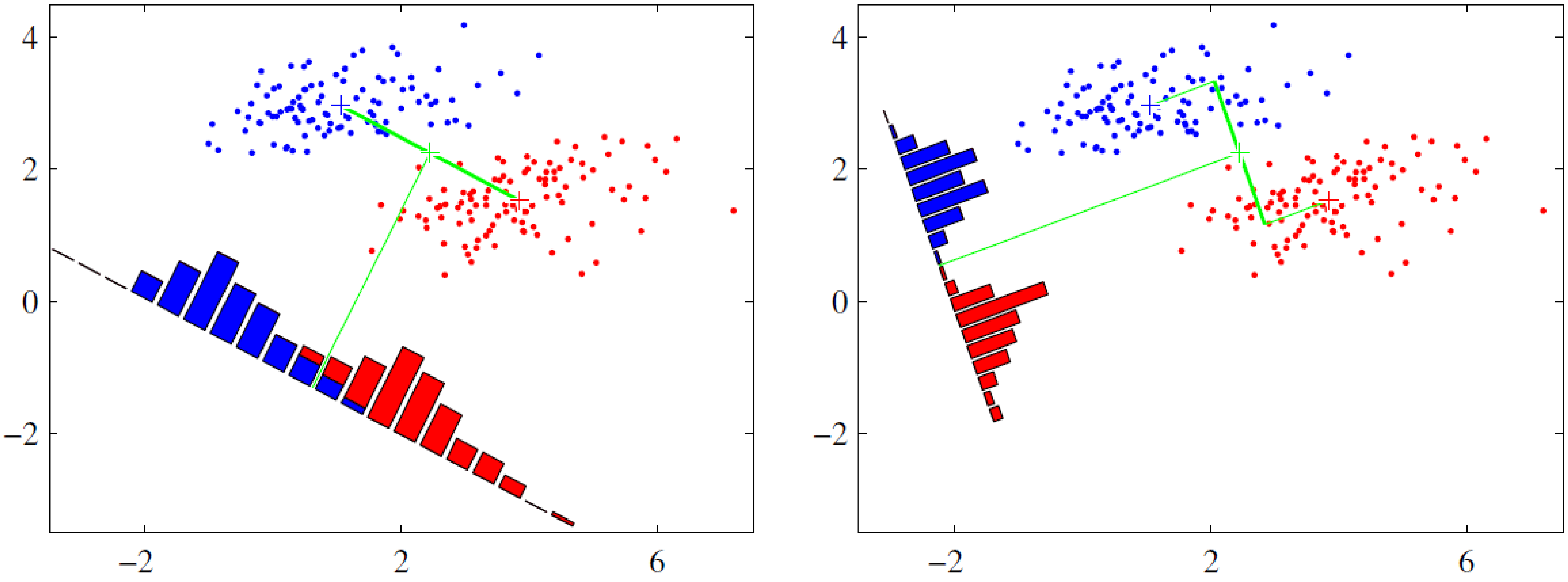
The Fisher criterion $J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$ $\qquad J(\mathbf{w}) = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w}}$ ➡ $\mathbf{w} \propto \mathbf{S}_{\mathrm{W}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$

Fisher linear discriminant

between-class covariance matrix: $\mathbf{S}_{\mathrm{B}} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^{\mathrm{T}}$

total within-class covariance matrix $\mathbf{S}_{\mathrm{W}} = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^{\mathrm{T}} + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^{\mathrm{T}}$

FISHER LINEAR DISCRIMINANT



FISHER LINEAR DISCRIMINANT

Perceptron function $y(\mathbf{x}) = f\left(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x})\right)$ nonlinear activation function $f(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0. \end{cases}$

Target values $t = +1$ for class $\mathcal{C}_1$ and $t = -1$ for class $\mathcal{C}_2$.

perceptron criterion $E_{\mathrm{P}}(\mathbf{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)t_n$ $\mathcal{M}$ denotes the set of all misclassified patterns

Stochastic gradient descent

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_{\mathrm{P}}(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta\phi_n t_n$$

Perceptron convergence theorem: if there exists an exact solution (in other words, if the training data set is linearly separable), then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps.

PERCEPTRON ALGORITHM

https://tamas.xyz/perceptron-demo/app/

OUTLINE

LINEAR MODEL FOR REGRESSION

LINEAR MODEL FOR CLASSIFICATION

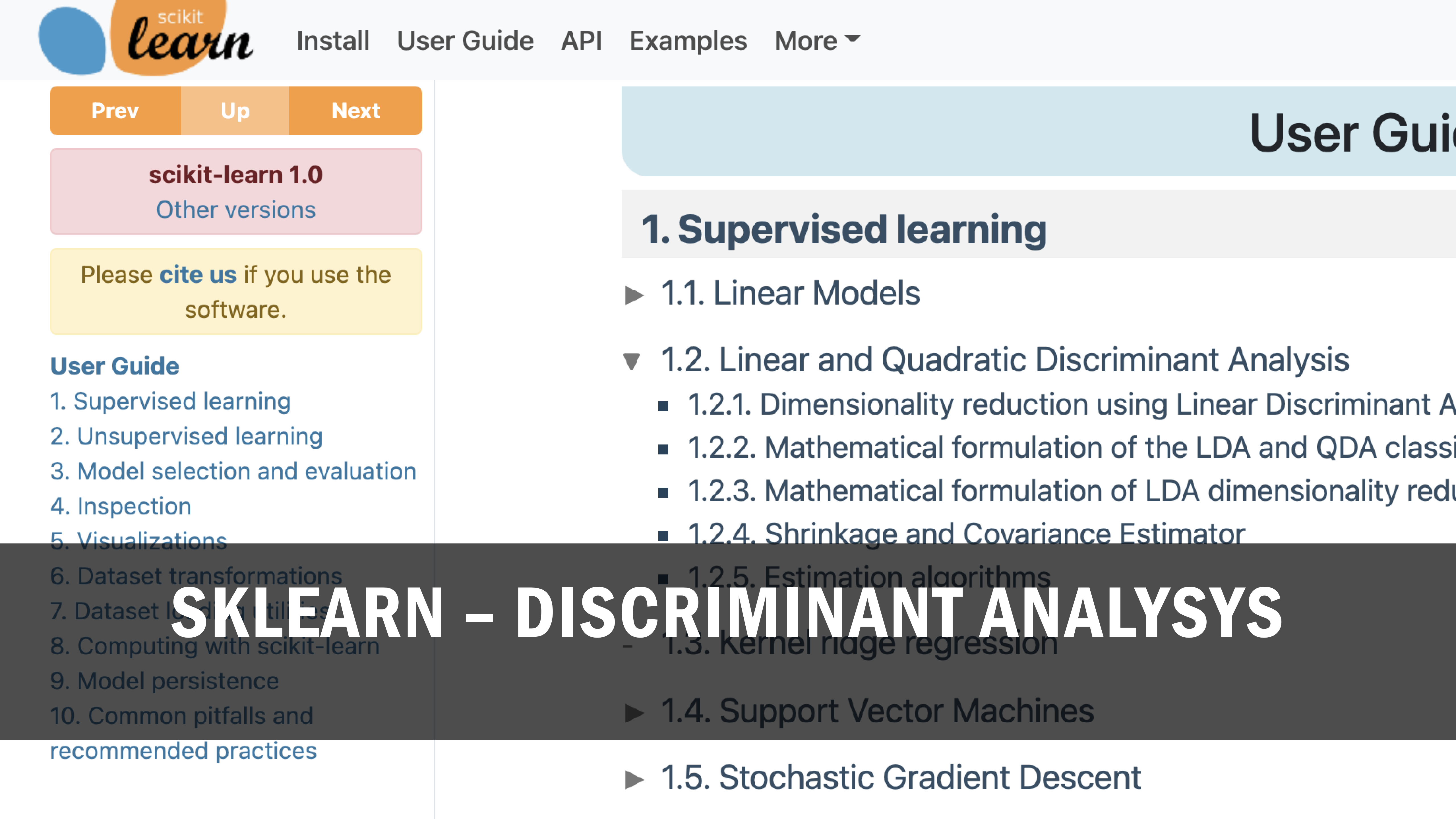
LINEAR MODELS IN SCIKIT-LEARN

WEEK 2 HOMEWORK

SKLEARN – LINEAR MODELS

User Guide

1. Supervised learning



scikit learn
Install
User Guide
API
Examples
More
Prev
Up
Next
scikit-learn 1.0
Other versions
Please cite us if you use the software.
User Guide
1. Supervised learning
2. Unsupervised learning
3. Model selection and evaluation
4. Inspection
5. Visualizations
6. Dataset transformations
8. Computing with scikit-learn
9. Model persistence
10. Common pitfalls and recommended practices
1. Supervised learning
1.1. Linear Models
1.2. Linear and Quadratic Discriminant Analysis
1.2.4. Shrinkage and Covariance Estimator
1.2.5. Estimation algorithms
1.3. Kernel ridge regression
1.4. Support Vector Machines
1.5. Stochastic Gradient Descent
SKLEARN – DISCRIMINANT ANALYSYS

OUTLINE

- LINEAR MODEL FOR REGRESSION
- LINEAR MODEL FOR CLASSIFICATION
- LINEAR MODELS IN SCIKIT-LEARN
- WEEK 2 HOMEWORK

WEEK 2 HOMEWORK

- Youtube Fun Videos!
  - StatQuest Linear Regression
  - Linear Discriminant Analysis (LDA)
- Python Hack!
  - SKLearn Linear Models
  - Kaggle – Linear Regression