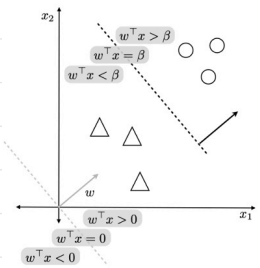
## NCC and Perceptron

| | NCC | Perceptron |
|---|---|---|
| Problem | Classification | Classification, (Regression) |
| Model | $y = \text{sign}(\mathbf{w}^T\mathbf{x})$ | $y = f(\mathbf{w}^T\mathbf{x})$ |
| Error | distance to $\mathbf{w}_{+1}, \mathbf{w}_{-1}$ | $-\sum_{m\in M} \mathbf{w}^T\mathbf{x}_m y_m$ |
| Optimization | closed form | SGD |
| Result | always the same | can differ |
| Application | Cancer Prediction[1] | NLP[2] |

NCC:

$$0 < \underbrace{(\mathbf{w}_o - \mathbf{w}_\Delta)^\top}_{\mathbf{w}}\mathbf{x} - \underbrace{\frac{1}{2}(\mathbf{w}_o^\top\mathbf{w}_o - \mathbf{w}_\Delta^\top\mathbf{w}_\Delta)}_{\beta}$$

$$y(x) = \mathbf{w}^\top\mathbf{x} - \beta$$




$\mathbf{x}, \mathbf{w} \in \mathbb{R}^D, \beta \in \mathbb{R}$

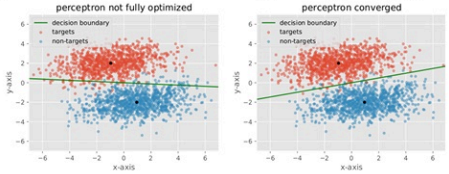
$$\mathbf{w}^\top\mathbf{x} - \beta = \begin{cases} > 0 & \text{if } \mathbf{x} \text{ belongs to o} \\ < 0 & \text{if } \mathbf{x} \text{ belongs to } \Delta \end{cases}$$

Points on the decision boundary satisfy $y(\mathbf{x}) = \mathbf{w}^\top\mathbf{x} - \beta = 0$

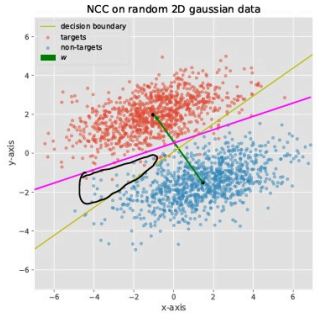
## Repetition Perceptron

- algorithm to reduce the classification error iterative with gradient descent
- include the bias $\beta$ into $\mathbf{w}$ (see first lecture)

1 initialize $\mathbf{w}$ randomly
2 until convergence (runtime constraint or no misclassifications)
  a pick a random misclassified point $\mathbf{x}_k$
  b update the weights with gradient descent: $\mathbf{w}_{new} = \mathbf{w}_{old} - \eta\mathbf{x}_k y_k$




## Repetition Nearest Centroid Classifier

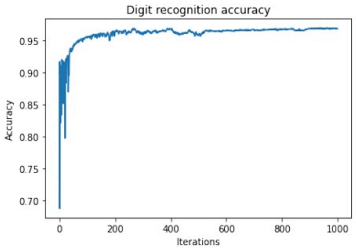



- compare distance of data points $\mathbf{x}$ to class means
- decide for class with closer center

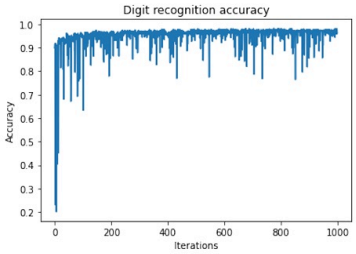
$$\mathbf{w} = \mathbf{w}_o - \mathbf{w}_\Delta$$
$$\beta = \frac{1}{2}(\mathbf{w}_o^T\mathbf{w}_o - \mathbf{w}_\Delta^T\mathbf{w}_\Delta)$$
$$f(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x} - \beta)$$

- correlation not considered, thus often not optimal

Perceptron: A good learning rate is useful, but not always necessary to reach convergence.




learning rate $\eta = \frac{1}{t}$




learning rate $\eta = 1$

## Task 4 - NCC vs. LDA [6 points]

Last lecture we looked at NCC. This lecture we looked at LDA. These models are closely related, as you saw in the lecture. They are both linear classifiers, which means they both work using a weight vector $\mathbf{w}$ and a bias $\beta$, and classify points $\mathbf{x}_i$ using the expression: $\mathbf{w}^T\mathbf{x}_i - \beta \geq 0$
Recall the definition for LDA given in the lecture, given a dataset $X \in \mathbb{R}^{d\times n}$ and class means $\bar{\mathbf{x}}_+, \bar{\mathbf{x}}_- \in \mathbb{R}^d$

$$\mathbf{w} = \bar{S}^{-1}(\bar{\mathbf{x}}_+ - \bar{\mathbf{x}}_-)$$
$$\beta = \frac{1}{2}\mathbf{w}^T(\bar{\mathbf{x}}_+ + \bar{\mathbf{x}}_-)\left[+\log\left(\frac{n_-}{n_+}\right)\right].$$

vs for NCC

$Y = sign(w^Tx - \beta)$

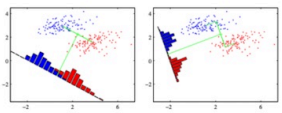
$$\mathbf{w} = \bar{\mathbf{x}}_+ - \bar{\mathbf{x}}_-$$
$$\beta = \frac{1}{2}\mathbf{w}^T(\bar{\mathbf{x}}_+ + \bar{\mathbf{x}}_-)$$

## Linear Discriminant Analysis

- given class means $\mathbf{w}_o, \mathbf{w}_\Delta \in \mathbb{R}^d$ and number of points $n_o, n_\Delta$ per class

$\Sigma_X = \frac{1}{n}(x-\bar{x})(x-\bar{x})^T$

$$\mathbf{w} = \Sigma_X^{-1}(\mathbf{w}_o - \mathbf{w}_\Delta)$$
$$\beta = \mathbf{w}^T\left(\frac{\mathbf{w}_o + \mathbf{w}_\Delta}{2}\right)\left[+\log\left(\frac{n_o}{n_\Delta}\right)\right]$$



- the term in square brackets is zero for $n_o = n_\Delta$
- LDA is the optimal classifier when
  - both classes are gaussian distributed
  - both covariance matrices $\Sigma_X$ are equal and known

**Goal**

Find $\mathbf{w} \in \mathbb{R}^d$ that
- maximizes mean class difference
- minimizes variance in each class

**Formalization**

Maximize the **Fisher criterion**

$$J(\mathbf{w}) = \frac{\text{between class variance}}{\text{within class variance}} = \frac{(\mu_o - \mu_\Delta)^2}{\sigma_o^2 + \sigma_\Delta^2}$$

**Solution**

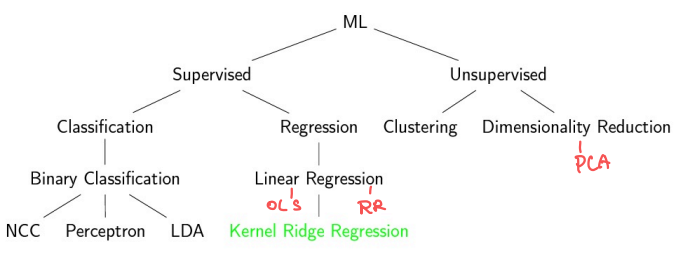
After some calculations...

$$\mathbf{w} \propto S_W^{-1}(\bar{\mathbf{x}}_o - \bar{\mathbf{x}}_\Delta)$$

$S_B$: between class scatter

$S_W$: within class scatter

The Tree of CA



# Linear Regression - The Solution

OLS

- data $x_1, \ldots, x_n \in \mathbb{R}^1$ with respective class labels $y_1, \ldots, y_n \in \mathbb{R}$
- the function $f(x) = w^\top x$ is optimized for

$$w = (XX^\top)^{-1} X y^\top$$

- in the sense of ordinary least squares, that is

$$\mathcal{E}_{lsq}(w) = \sum_{i=1}^{N} (y_i - w^\top x_i)^2 = ||y - w^\top X||^2$$

- *Optimizing* $\mathcal{E}_{lsq}(w)$ *by setting* $\frac{\partial \mathcal{E}_{lsq}(w)}{\partial w} = 0$ *leads to the same result.*

# Polynomial Regression - LSE

- This leads again to a LSE, that can be solved

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

$$\Phi^\top w = y^\top$$

$$\Phi\Phi^\top w = \Phi y^\top$$

$$w = (\Phi\Phi^\top)^{-1} \Phi y^\top$$

- Other choices of basis expansion $\phi$ yield e.g. spline regression

# Ridge Regression

- Regression with panelization: restrict large values for w
- Often it is important to control the complexity of the solution w

$$\mathcal{E}_{RR}(w) = \underbrace{||y - w^\top X||^2}_{OLS} + \underbrace{\lambda ||w||^2}_{RR}$$

$$\frac{\partial \mathcal{E}_{RR}}{\partial w} = 0$$

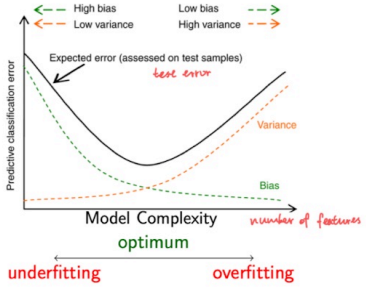
- Solution is given by

$$w = (XX^\top + \lambda I)^{-1} X y^\top$$

Hyperparameter

biased estimator
but smaller variance

# The bias-variance trade-off




**Careful. . .**

*Bias* and *variance* are terms with multiple (related) usages

- $w^\top x + b$, $w \cdot x$
- Bias/variance of estimator
- Bias/variance of general ML mod

Underfitting occurs when a model is too simple to capture the underlying structure of the data, while overfitting happens when a model is too complex, capturing noise in the data as if it were signal.

# Constructing Valid Kernels

**1. Option**

- Find it by starting with a feature space mapping: $k(x, x') = \phi(x)^T \phi(x')$

**2. Option**

- Show the kernel function corresponds to a scalar product in some feature space
- Option 1: show $K := [k(x_i, x_j)]_{ij}$ is symmetric PSD for all possible choices of $X \subseteq \mathcal{X}$
- Option 2: construct directly from other valid kernels!

# Kernelizing Ridge Regression

- Instead we compute:

$$f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\phi(\mathbf{x}) = \phi(\mathbf{x})^\mathsf{T}\mathbf{w} = \underbrace{\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{X})}_{k(\mathbf{x},\mathbf{X})}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}$$

$$= k(\mathbf{x}, \mathbf{X}) \underbrace{(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}}_{\alpha}$$

Instead of $\mathbf{w}$ we compute $\alpha$. What is $\alpha$?

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$$

# Kernelized RR - The Algorithm

1. Compute $\alpha$:

$$\alpha = (\mathbf{K} + \lambda I)^{-1} \mathbf{y}_{train}^T$$
$$\text{with } \mathbf{K} := k(\mathbf{X}_{train}, \mathbf{X}_{train})$$

2. Predictions:

$$\hat{y}_{new} = \alpha^T k(\mathbf{X}_{train}, x_{new})$$

Kernels:

- Unknown (data in) feature space
- $D \gg n$
- Non-numeric data
- The entire dataset has to be stored

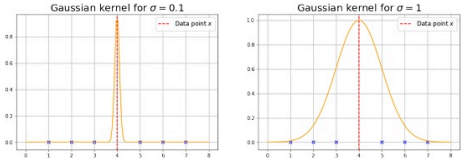
Feature maps:

- Feature map often infeasible to compute
- $n \gg D$

# Gaussian Kernel

- $k(x, x') := exp\{-\frac{||x-x'||^2}{2\sigma^2}\}$

$k_{RBF}$



## Repetition: Eigenvalues and Eigenvectors

Given a Matrix $A \in \mathbb{R}^{d \times d}$, then a non-zero vector $\mathbf{v} \in \mathbb{C}^d \setminus \{\mathbf{0}\}$ is called an eigenvector, if there is an eigenvalue $\lambda \in \mathbb{C}$, such that

$$A\mathbf{v} = \lambda\mathbf{v}$$

- eigenvectors are special directions, for which a matrix $A$ is only scales, but not rotate
- A matrix is singular if one or more eigenvalues are zero

$$\lambda = 0 \Leftrightarrow \text{Kern}(A) \neq \emptyset$$

## Rotation Matrices II

- Rotation matrices $R_\theta$ are always orthogonal with $\det |R_\theta| = +1$
- Are all orthogonal matrices also rotation matrices? (Task 4.2)
- For a orthogonal matrix $V$ the determinant is $\det |V| = \pm 1$

$$V = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- V is orthogonal, since $V^\top V = I$, but $\det |V| = -1$ and thus it is not a rotation matrix

## Summary: Principal Component Analysis

1. Estimate the covariance matrix $S$ of the data $X \in \mathbb{R}^{d \times n}$
2. Compute the eigenvectors of $S$
3. $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k] \in \mathbb{R}^{d \times k}$ where $\mathbf{w}_1, \ldots, \mathbf{w}_k \in \mathbb{R}^d$ are the eigenvectors corresponding to the $k$ largest eigenvalues
4. Project the data onto $W$: $H = W^\top \cdot X$
5. If needed: reconstruct data by $X \approx \tilde{X} = WH$.
   This holds for all linear matrix factorization methods.

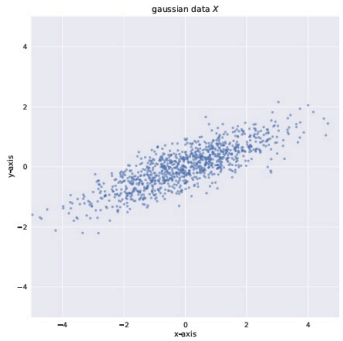
## Kernel PCA Algorithm

**Algorithm 2:** Kernel PCA

**Input:** Dataset $X \in \mathbb{R}^{d \times n}$ with $d \gg n$; Number of PCs $k$
**Output:** First $k$ PCs $W \in \mathbb{R}^{d \times k}$, Hidden causes $H \in \mathbb{R}^{k \times n}$ of the dataset

1 Compute Kernel $K = (X - \overline{X})^\top (X - \overline{X}) \in \mathbb{R}^{n \times n}$
2 Compute EVD $K = V \Lambda V^\top$
3 Take first $k$ eigenvectors corresponding to largest eigenvalues $\alpha = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$
4 Calculate $W = X\alpha \in \mathbb{R}^{d \times k}$
5 Project data $H = W^\top X$
6 **return** $W$ *and* $H$

## Principle Component Analysis



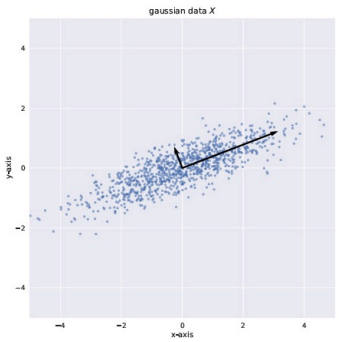

- Which dimensions contain the most information?

$\Rightarrow$ directions $\mathbf{w}$ with the highest variance

$$\underset{\mathbf{w}}{\text{argmax}}\, \text{var}(\mathbf{w}^\top X) = \underset{\mathbf{w}}{\text{argmax}}\, \mathbf{w}^\top \Sigma_X \mathbf{w}$$

$\Rightarrow$ directions $\mathbf{w}$ that minimizes the noise

$$\underset{\mathbf{w}}{\text{argmin}} \sum_{i=1}^{n} ||(\mathbf{w}^\top \mathbf{x}_i)\mathbf{w} - \mathbf{x}_i||^2 = \underset{\mathbf{w}}{\text{argmax}}\, \mathbf{w}^\top \Sigma_X \mathbf{w}$$




- $\mathbf{w}^T \Sigma_X \mathbf{w}$ is maximized by the eigenvector of $\Sigma_X$ with the largest eigenvalue
- The eigenvectors of $\Sigma_X$ are the principle components
- PCs can be constrained to unit length $||\mathbf{w}|| = 1$
- Variance along PC is given by the corresponding eigenvalue, since

$$\text{var}(\mathbf{w}^\top X) = \mathbf{w}^T \Sigma_X \mathbf{w} = \mathbf{w}^T \lambda \mathbf{w} = \lambda \mathbf{w}^T \mathbf{w} = \lambda$$

## Kernel PCA

- If dimensionality becomes larger than number of samples $d \gg n$
  - Only $\leq n$ non-zero eigenvalues
  - Covariance $\Sigma_X$ becomes singular, $\text{Rank}(\Sigma_X) \leq n$
  - Covariance is positive *semi* definite
  - Covariance $\Sigma_X \in \mathbb{R}^{d \times d}$ will be very large
  - High time and space complexity for calculation
- Kernel PCA should be used to estimate $n$ first Principal Components

## From PCA to NMF

- Given data $X \in \mathbb{R}^{d \times n}$ and PCs in the columns of a matrix $W \in \mathbb{R}^{d \times k}$
- Data can be projected onto PCs $H = W^\top X \Leftrightarrow X \approx WH$
- If $X > 0$, it should be possible to choose $W, H$ such that $W > 0, H > 0$
- Goal of NMF is to find $W \in \mathbb{R}^{d \times k}$ and $H \in \mathbb{R}^{k \times n}$, sucht that $||X - WH||^2_{\text{Fro}}$ is minimized
- Cannot be directly calculated $\Rightarrow$ iterative optimization with gradient descent