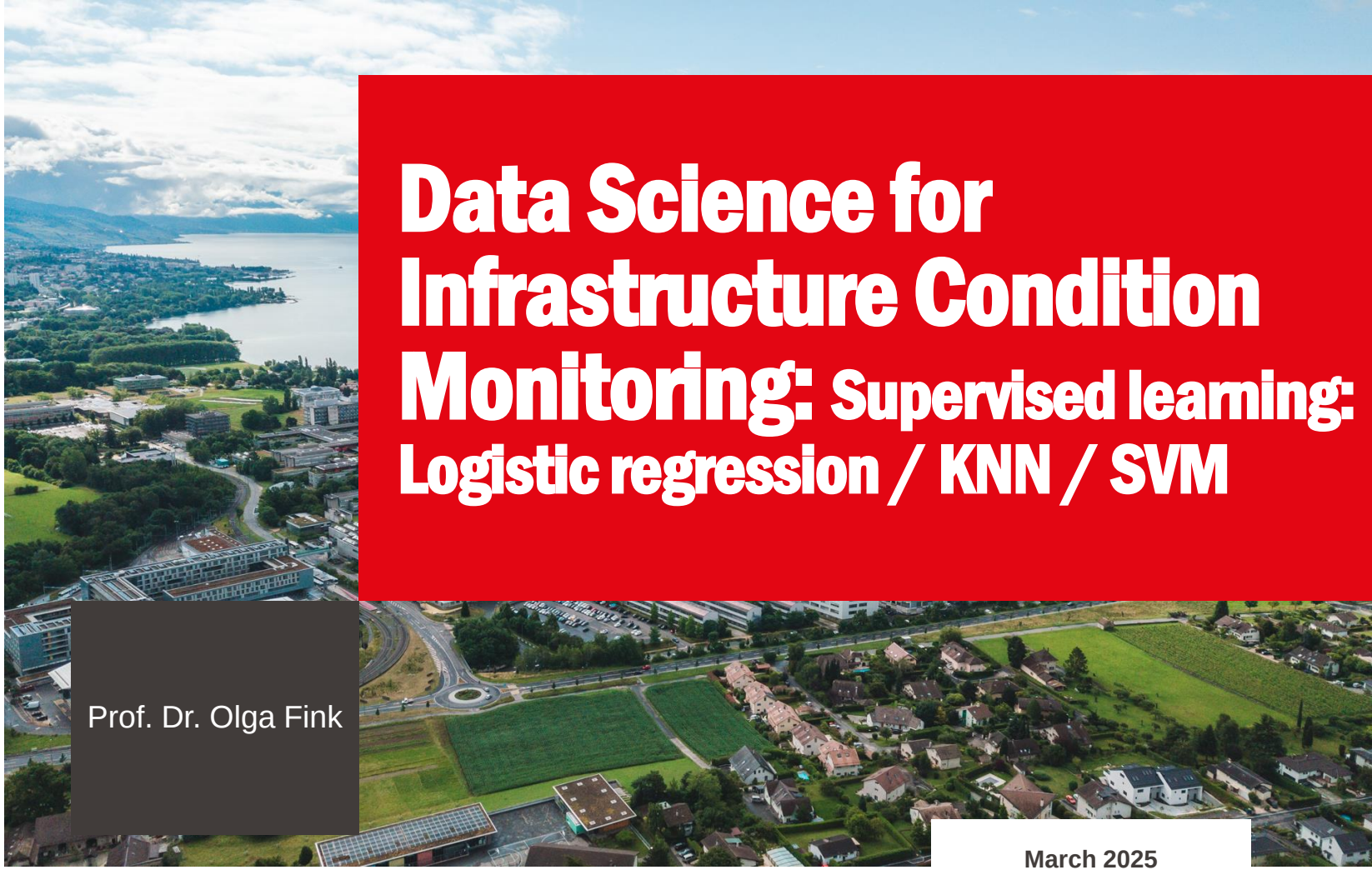


Data Science for Infrastructure Condition Monitoring: Supervised learning: Logistic regression / KNN / SVM

Prof. Dr. Olga Fink

École polytechnique fédérale de Lausanne

March 2025

Supervised Learning

Applications in Infrastructure Monitoring

1. **Damage Detection and Classification → Fault Diagnostics**

- **Objective**: Identify and classify damage types (e.g., cracks, corrosion, delamination).
- **Data**: Sensor measurements (strain, acceleration, temperature), images (vision-based inspections).
- **Example**: Classifying crack severity in bridge components using labeled inspection images.

2. **Anomaly/Fault Detection**

- **Objective**: Detect deviations from normal structural behavior.
- **Data**: Time-series data from sensors (vibration, strain gauges).
- **Example**: Identifying abnormal vibration patterns in wind turbines indicating early-stage failures.

3. **Remaining Useful Life (RUL) Prediction → Prognostics**

- **Objective**: Estimate how much time is left before maintenance is required or failure occurs.
- **Data**: Historical degradation data, maintenance records.
- **Example**: Predicting the lifespan of bridge cables under varying load conditions.

4. **Load and Stress Estimation**

- **Objective**: Predict loads or stress levels on structural components in real time.
- **Data**: Displacement, strain, temperature measurements.
- **Example**: Estimating traffic-induced stress on highway overpasses.

Regression in Infrastructure Monitoring

- Predicts **continuous numerical values** based on input data.
- **Examples in Infrastructure Monitoring**
- **Remaining Useful Life (RUL) Prediction**
  → Predict how many days/weeks a bridge component will last before maintenance is needed.
- **Stress / Strain Estimation**
  → Estimate stress at a specific point on a bridge deck not directly instrumented.
- **Deflection / Displacement Prediction**
  → Predict the displacement of a structure under load in real time.
- **Crack Length Growth Prediction**
  → Forecast how long a crack in concrete or steel will become over time.

Classification in Infrastructure Monitoring

- Assigns **categories** to input data

→Determine the state or condition of the structure by placing it into predefined classes.

- Fault Type Identification→ Identify the type of fault: corrosion, crack, delamination.
- Health State Classification→ Classify a bridge as "Healthy", "Warning", or "Critical".

Virtual Sensing in Infrastructure Monitoring

- **Definition**:
  Virtual sensing (also known as soft sensing) refers to the estimation or reconstruction of unmeasured physical quantities at specific locations or times using mathematical models and data from existing physical sensors.
- **How it works**:
  Virtual sensors use **machine learning**, **physics-based models**, or **data-driven algorithms** to infer measurements in locations where installing physical sensors is impractical or too costly.

How Does Virtual Sensing Work?

1. Collect data from existing sensors(e.g., strain, acceleration, temperature).
2. Train predictive models (e.g., Machine Learning).
3. Estimate unmeasured states or variables(e.g., stress at an unmonitored point, displacement, load).

When and Where Can Virtual Sensing Be Applied?

1. **Structural Response Estimation**

Use Case: Predict strain, displacement, or stress in regions of a structure where no sensors are installed.

Example: Estimating stress at mid-span of a bridge using data from sensors placed at supports.

2. **Damage Localization and Quantification**

Use Case: Detect and localize potential damage zones by reconstructing full-field responses.

Example: Virtual strain sensing to detect damage in wind turbine blades or offshore platforms.

3. **Cost-Effective Monitoring**

Use Case: Reduce the number of physical sensors required without compromising data coverage.

Example: Use fewer accelerometers on large buildings but infer vibrations in unmeasured areas.

4. **Hard-to-Access or Hazardous Areas**

Use Case: Monitor areas where deploying physical sensors is dangerous or infeasible.

Example: Virtual sensing inside pipelines, deep inside dams, or underwater structures.

5. **Enhancing Sparse Sensor Networks**

Use Case: Fill in gaps in sensor data, improving the spatial resolution of monitoring systems.

Example: Reconstructing full bridge deck deflection profiles from a limited number of displacement sensors.

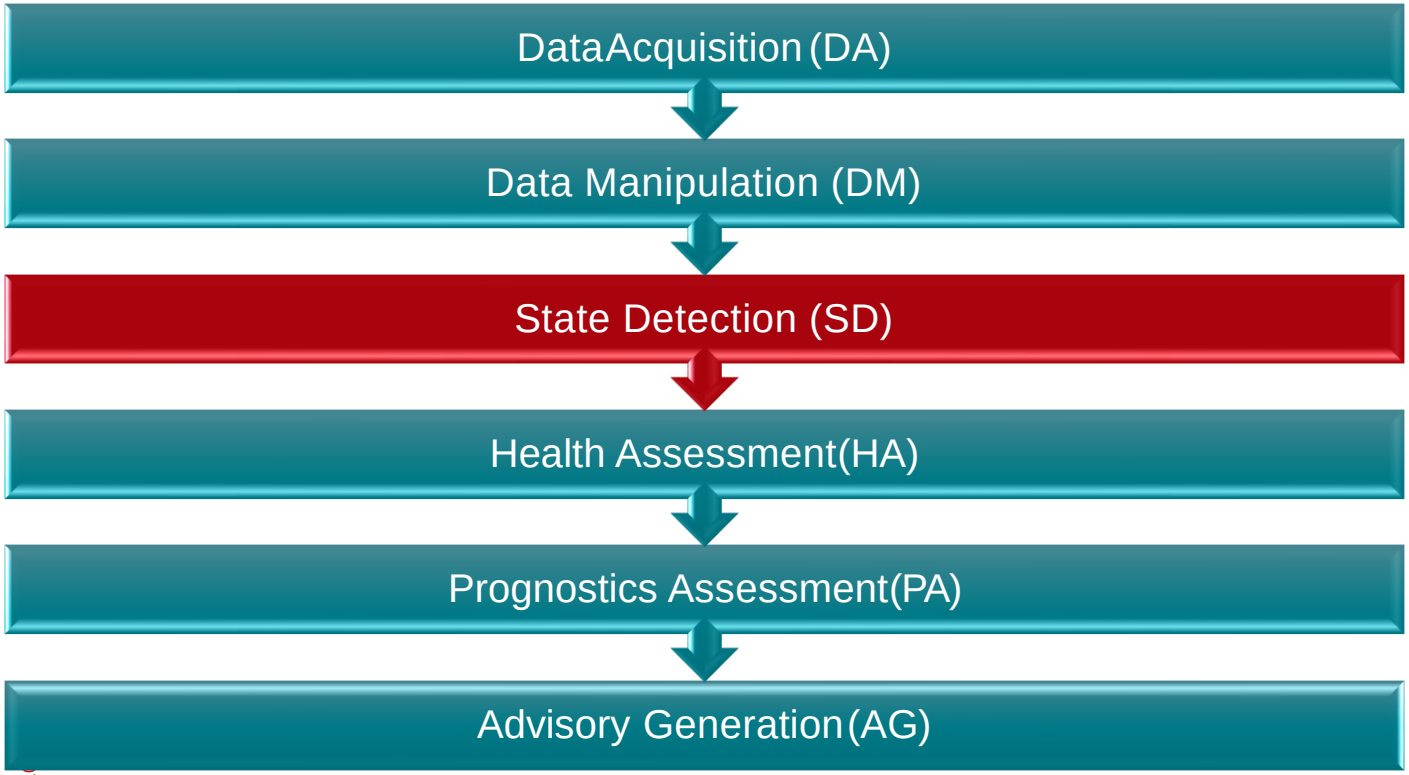
Benefits of Virtual Sensing

- Reduces hardware and maintenance costs
- Enables monitoring of inaccessible locations
- Increases spatial resolution of monitoring data
- Enhances understanding of structural behavior
- Supports proactive maintenance and decision-making

Why Supervised Learning Is Often a Bad Idea for Anomaly Detection

- Lack of Labeled Anomaly Data
  - Anomalies are rare by nature-collecting enough labeled examples is difficult.
  - Manual labeling of anomalies (e.g., structural damage) is time-consuming, expensive, and often requires expert domain knowledge.
  - Real-world systems (e.g., bridges, turbines) rarely fail or exhibit damage during normal operation, so anomaly datasets are typically highly imbalanced.
- Class Imbalance Problem
  - Anomalies are rare → massive imbalance between normal and abnormal examples.
  - Supervised algorithms can become biased toward the normal class, leading to:
    - High false negatives (missed anomalies)
    - Poor detection of rare but critical events (e.g., early signs of failure)
- Anomalies Are Not Always the Same
  - Anomalies can be unexpected, diverse, and unpredictable.
  - Supervised models require pre-defined classes of anomalies to train on.
  - New, unseen anomalies will not be detected well because the model has never encountered them during training.
- Data Drift and Changing Conditions
  - Infrastructure systems change over time:
    - Wear and tear
    - Environmental variations
    - Sensor noise/failures
  - Supervised models trained on historical data may fail to generalize to new operating conditions unless frequently retrained, which is impractical.

PHM Process


Data Acquisition (DA)
Data Manipulation (DM)
State Detection (SD)
Health Assessment(HA)
Prognostics Assessment(PA)
Advisory Generation (AG)

Performance Evaluation

Confusion matrix: missed alarms/ false alarms

True class → / Predicted class ↓	Fault	No fault
Positive (detected)	TP	FP
Negative (not detected)	FN	TN
	P=TP+FN	N=FP+TN

- Rate of missed alarms:
  FN/(TP+FN)
- Rate of false alarms:
  FP/(FP+TN)

Classification performance

$$Precision = \frac{True\ Positive}{True\ Postitive + False\ Positive}$$

$$Specificity = \frac{True\ Negative}{True\ Negative + False\ Postitive}$$

$$Recall(Sensitivity) = \frac{True\ Positive}{True\ Postitive + False\ Negative}$$

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total}$$

$$F_1 = \frac{2 \cdot (Recall \cdot Precesion)}{Recall + Precesion}$$

$$F_\beta = (1 + \beta^2)\frac{Recall \cdot Precesion}{Recall + \beta^2 Precesion}$$

True class → / Predicted class ↓	Fault	No fault
Positive (detected)	TP	FP
Negative (not detected)	FN	TN

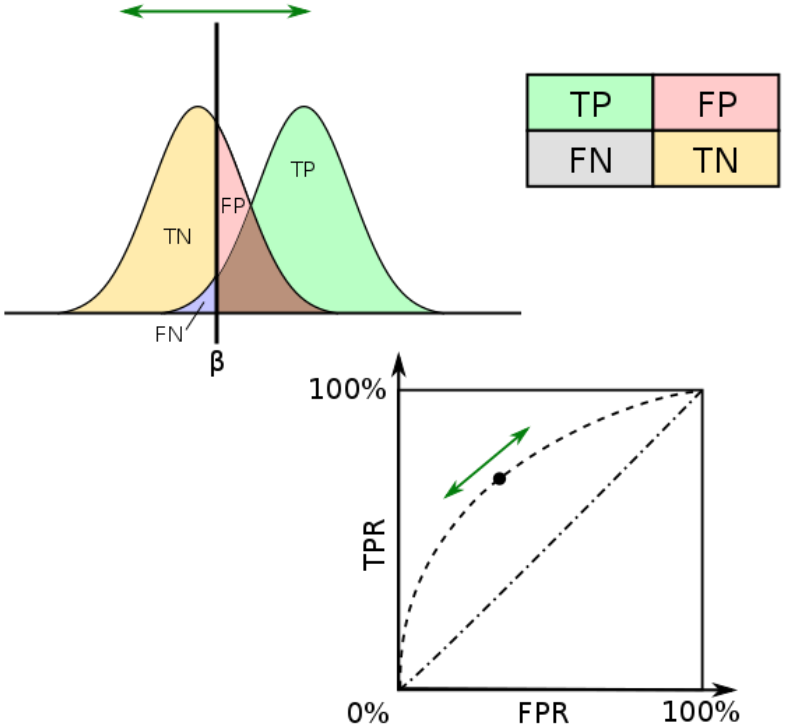
Precision

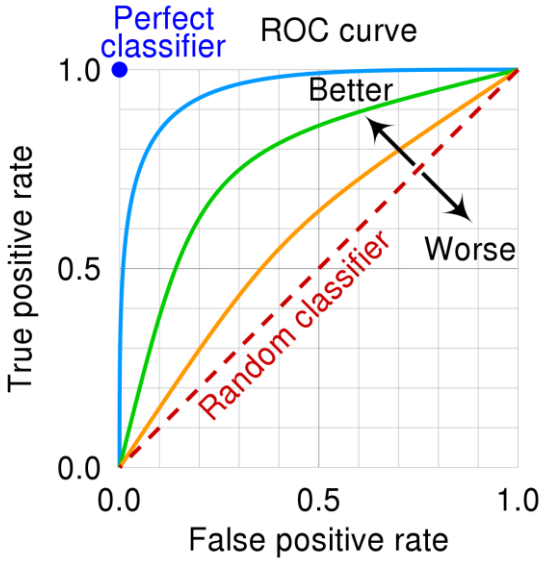
Recall

Specificity

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right)$$

ROC = Receiver Operating Characteristic
AUC = Area under the curve





Source: Wikimedia

Performance Metrics for Regression

Metric	Description	Equation	Interpretation
Mean Absolute Error (MAE)	Average of absolute differences between predicted and actual values.	$MAE = (1/n)\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$	Lower is better. Easy to interpret as average error in units of the target.
Mean Squared Error (MSE)	Average of squared differences between predicted and actual values.	$MSE = (1/n)\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$	Lower is better. Penalizes larger errors more heavily.
Root Mean Squared Error (RMSE)	Square root of the MSE. Provides error in same units as target.	$RMSE = \sqrt{(1/n)\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	Lower is better. Sensitive to large deviations.
R-squared (R^2)	Proportion of variance explained by the model (goodness-of-fit).	$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$	Closer to 1 is better. Measures explained variability.
Mean Absolute Percentage Error (MAPE)	Average absolute percentage errors between predicted and actual values.	$MAPE = (100/n)\sum_{i=1}^{n}\lvert\frac{y_i - \hat{y}_i}{y_i}\rvert$	Lower % is better. Intuitive relative error measure.
Symmetric MAPE (sMAPE)	Scale-independent error. Avoids division by zero.	$sMAPE = (100/n)\sum_{i=1}^{n}\frac{\lvert y_i - \hat{y}_i\rvert}{(\lvert y_i\rvert + \lvert\hat{y}_i\rvert)/2}$	Lower % is better. Useful near zero values.

Other considerations

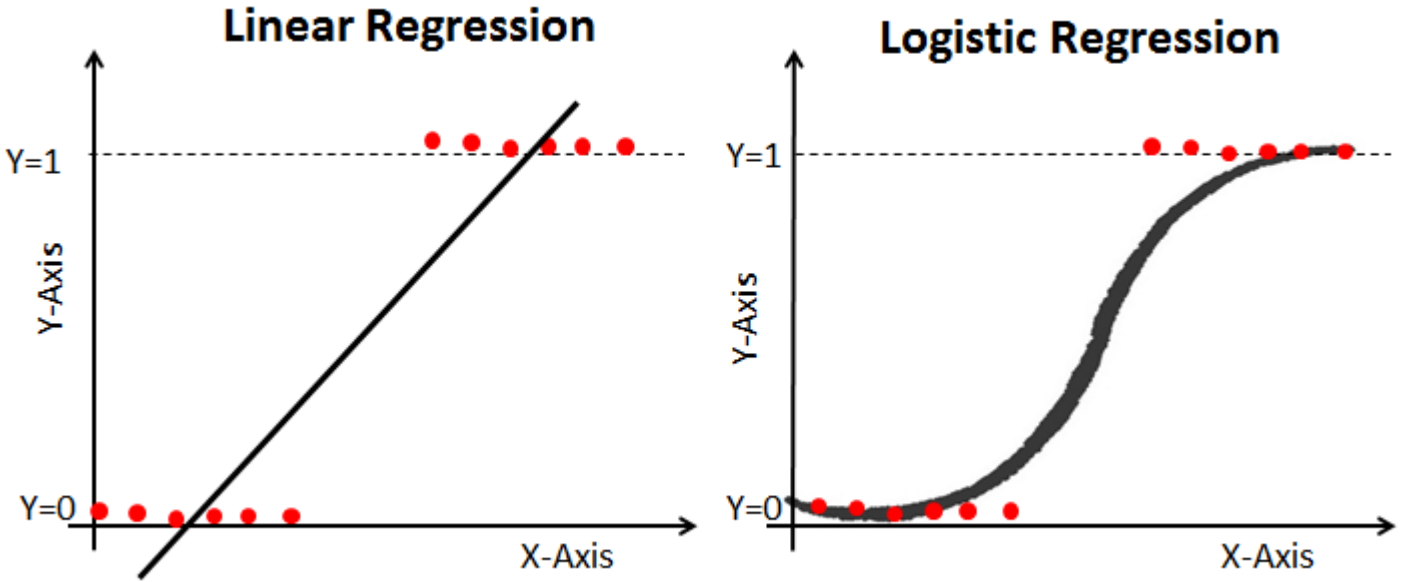
- Generalization Ability (How well does the model perform on unseen data?)
  - Overfitting vs. Underfitting
  - Performance gap between training and testing (or validation)
- Robustness (How stable is the model when inputs are noisy, incomplete, or perturbed)
  - Noise sensitivity analysis (injecting realistic noise into sensor data)
  - Outlier detection and performance under anomalous inputs
- Computational Efficiency (How resource-efficient is the model in terms of time and memory?)
  - Training time
  - Inference time (important for real-time monitoring)
  - Memory/CPU/GPU requirements
- Scalability (Can the model handle larger datasets or growing system complexity?)
  - Increase number of sensors/inputs (high-dimensional data)
  - Increase number of monitored structures
- Interpretability / Explainability (Can stakeholders understand and trust the model?)
- Robustness to Data Imbalance

Logistic Regression

Logistic regression

- Logistic regression is a statistical model used to predict the probability of a binary outcome (i.e., a "yes" or "no" answer) based on one or more predictor variables. It is a type of regression analysis commonly used in machine learning and statistics to model the relationship between the input features and the binary target variable.

Linear regression vs. Logistic regression



Source: www.towardsdatascience.com

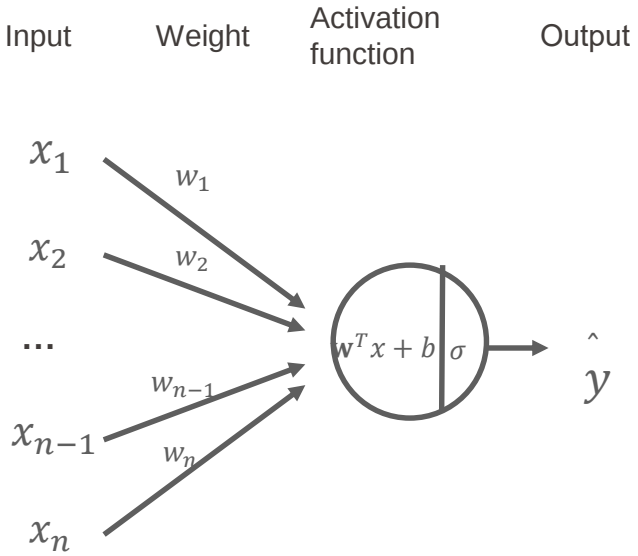
Logistic regression

$$\hat{y} = b_1 \cdot x_1 + b_2 \cdot x_2 + \ldots + b_k \cdot x_k + a$$

$$f(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(b_1 \cdot x_1 + \ldots + b_k \cdot x_k + a)}}$$

Logistic Regression

Visualisation



1) Linear Regression:

$$z^{(i)} = \mathbf{w}^T \boldsymbol{x}^{(i)} + b$$

2) Activation Function:

$$\hat{y}^{(i)} = \sigma(z^{(i)})$$

3) Classification:

$$if \hat{y}^{(i)} > 0.5 \Rightarrow label1$$

$$if \hat{y}^{(i)} \leq 0.5 \Rightarrow label0$$

Logistic Regression

Loss function

How to learn w?

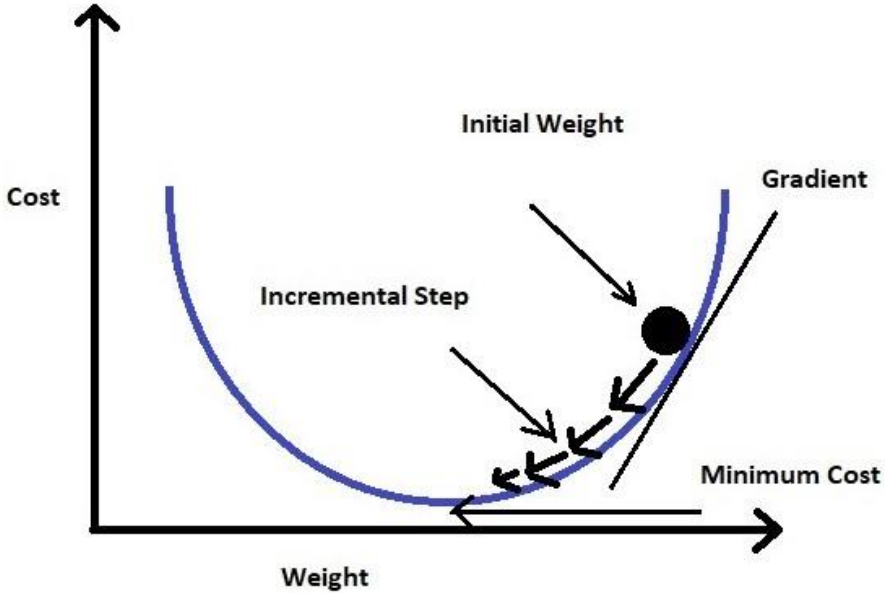
Binary cross-entropy loss

$$J = -\frac{1}{N}\sum_{i=1}^{N} y^{(i)}\log(h_{\mathbf{w}}(\boldsymbol{x}^{(i)})) + (1-y^{(i)})\log(1-h_{\mathbf{w}}(\boldsymbol{x}^{(i)}))$$

Active when y = 1
Penalize when $h_w(x) \to 0$

Active when y = 0
Penalize when $h_w(x) \to 1$

Gradient descent



$$\theta_{new} = \theta_{old} - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

Source: www.towardsdatascience.com

Logistic Regression

Softmax regression

Softmax function

$$\sigma(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^{K}\exp(z_j)}, i \in \{1,\dots,K\}$$

After applying softmax, each component will be in the interval $(0,1)$ and the component will add up to 1, so that they can be interpreted as probabilities

Softmax regression is a **generalization** of logistic regression to multi-class problems.

Note: For softmax regression, $\mathbf{z}$ is obtained as follows:

- $\mathbf{z} = \mathbf{W}^T\boldsymbol{x} + \mathbf{b}$
- $z_k = \mathbf{w}_k^T\boldsymbol{x} + b_k$ **where $\mathbf{w}_k$ is the k-th column of the D x K weight matrix $\mathbf{W}$.**

Logistic Regression

Categorical cross-entropy

Categorical Cross-Entropy Loss

$$J(\mathbf{w}, b) = -\sum_{i=1}^{N}\sum_{k=1}^{K} \mathbf{1}\{y^{(i)} = k\} \log\left(\frac{\exp(\mathbf{w}_k^T \boldsymbol{x}^{(i)} + b_k)}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^T \boldsymbol{x}^{(i)} + b_j)}\right)$$

where $\mathbf{1}\{y^i = k\}$ is “indicator function”, it works as:

$\mathbf{1}\{\textit{True statement}\} = 1$ and $\mathbf{1}\{\textit{False statement}\} = 0$

Example Logistic Regression

Problem Definition: detect open windows

Input:

- **Room Temperature**
- **Outside Temperature**
- **Past Room Temperature (T_{t-1}, T_{t-2})**
- **Room Temperature difference (ΔT_{t-1}, ΔT_{t-2})**
- **Presence Sensor**

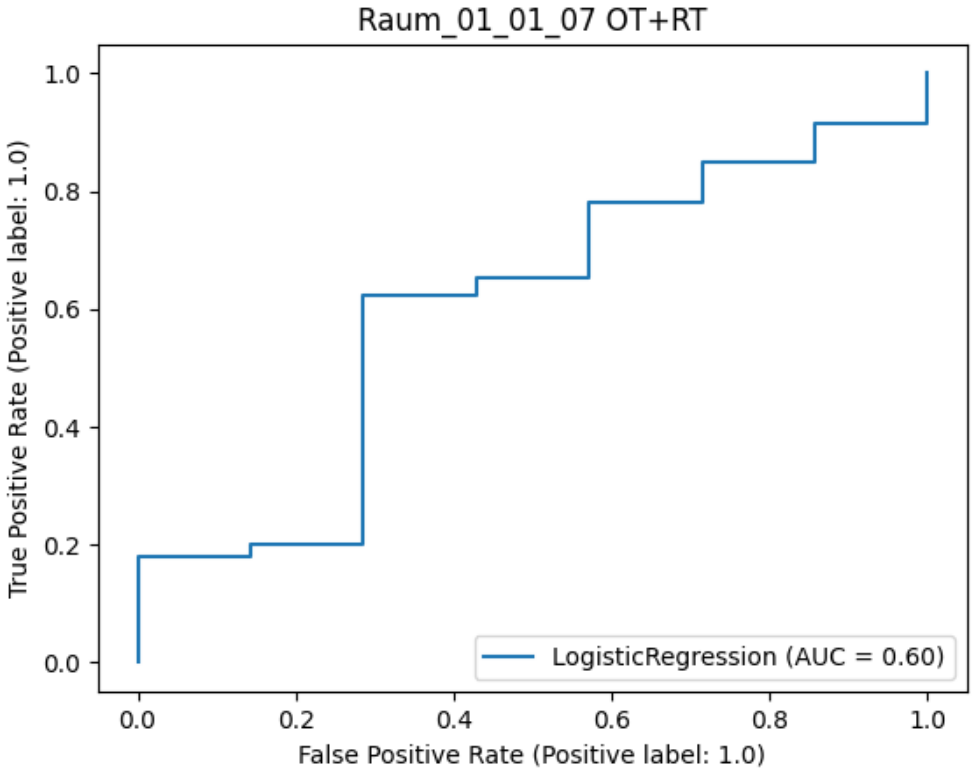
Timestep=**1min**.

Output
For each time t:
- $y_t = \mathbf{1}$ if **no opening** at time t
- $y_t = \mathbf{0}$ if **an opening** at time t

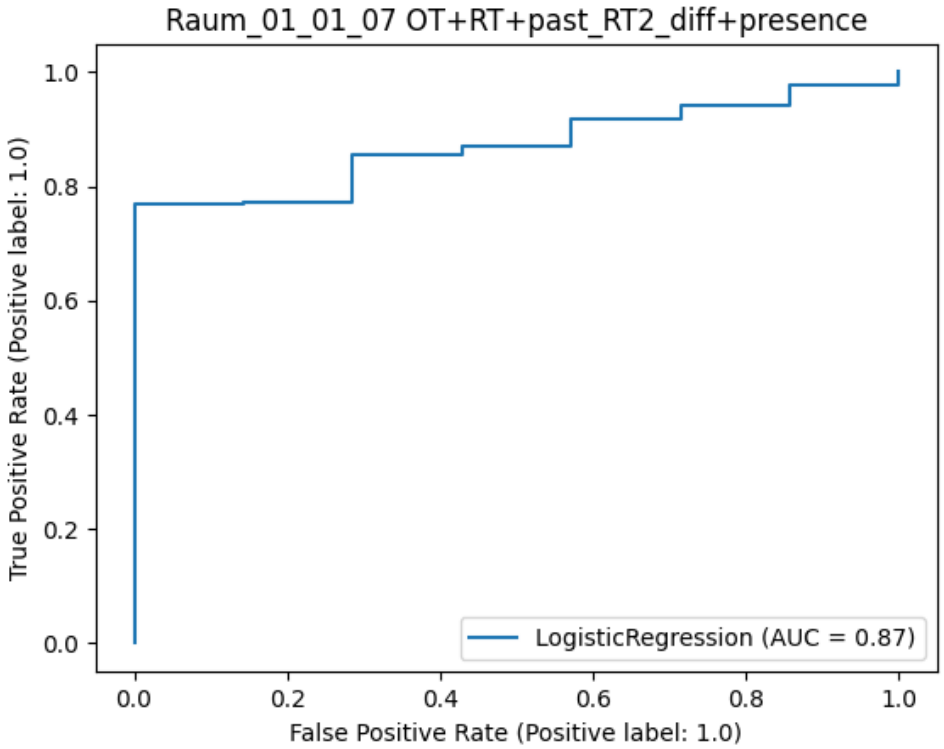
Results for different office rooms (AUC)

Room	OT+RT=B	B+RT_t-1	B+RT_t-1+RT_t-2=C	B+diff RT_t-1 + diff RT_t-2=D	B+ presence	C+presence	D+presence
01_01_07	0.60	0.60	0.61	0.62	0.87	0.87	0.87
01_01_27	0.67	0.67	0.67	0.67	0.78	0.78	0.78
01_01_56	0.58	0.58	0.59	0.59	0.71	0.71	0.71
01_01_57	0.80	0.78	0.76	0.71	0.81	0.80	0.79
03_02_26	0.69	0.69	0.69	0.69	0.79	0.79	0.79

Examples of ROC curves



Examples of ROC curves



K-Nearest Neighbor algorithm

Instance based learning

- Approximating real valued or discrete-valued target functions
- Learning in this algorithm consists of storing the presented training data
- When a new query instance is encountered, a set of similar related instances is retrieved from memory and used to classify the new query instance
- Construct only local approximation to the target function that applies in the neighborhood of the new query instance
- Instance-based methods can use vector or symbolic representation
- Appropriate definition of „neighboring“ instances
- Disadvantage of instance-based methods is that the costs of classifying new instances can be high
- Nearly all computation takes place at inference time rather than learning time

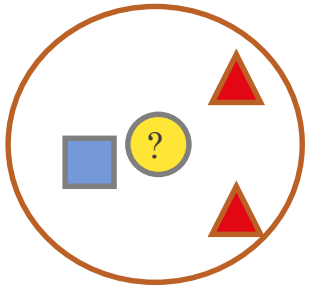
Source: Fenix, 2015

k-Nearest Neighbor algorithm

- Most basic instance-based method
- Data are represented in a vector space
- Supervised learning algorithm
- Distance measure required

Why k-NN?

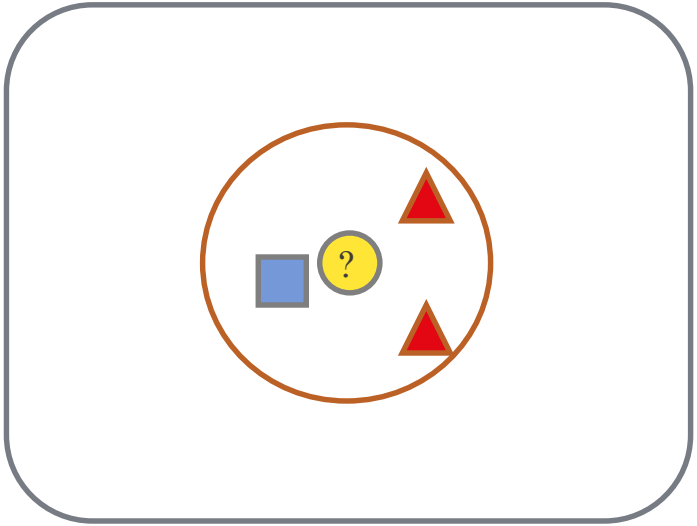
- Used to classify objects based on closest training examples in the feature space
  - Feature space: raw data transformed into sample vectors of fixed length using feature extraction (Training Data)
- Among the simplest classification algorithms
- Implementation of lazy learner
  - All computation deferred until classification



Source: Connor, 2006

k-NN Requirements

- Requires 3 things:
  - Feature Space (Training Data)
  - Distance metric
    - to compute distance between records
  - The value of *k*
    - the number of nearest neighbors to retrieve from which to get majority class
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record



Source: Connor, 2006

k-NN Distance measures

- Common Distance Metrics:
  - Euclidean distance (continuous distribution)

$$\|\vec{x} - \vec{y}\| = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

  - Cosine similarity

$$Sim(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|}$$

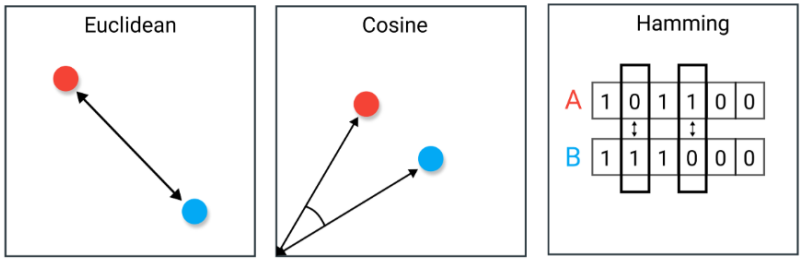
  - Hamming distance (overlap metric)

bat (distance = 1)
cat

toned (distance = 3)
roses

  - Discrete Metric(boolean metric)

if x = y then d(x,y) = 0. Otherwise, d(x,y) = 1



- Determine the class from ***k*** nearest neighbor list
  - Take the majority vote of class labels among the k-nearest neighbors
  - Weighted factor: w =1/d (generalized linear interpolation) or $1/d^2$

Source: Connor, 2006

Curse of Dimensionality

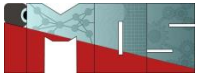
- Imagine instances described by 20 features (attributes) but only 3 are relevant to target function
- Curse of dimensionality: nearest neighbor is easily misled when instance space is high-dimensional
- Dominated by large number of irrelevant features

Possible solutions

- Stretch j-th axis by weight z_j, where $z_1,\ldots,z_n$ chosen to minimize prediction error (weight different features differently)
- Use cross-validation to automatically choose weights $z_1,\ldots,z_n$
- Feature subset selection if z_j set zero
- Dimensionality reduction

Source: Fenix, 2015

k-NN for Regression (similar to classification)



- **Requires same things as classification:**
  - **Feature Space**
  - **Distance Measure**
  - **The value of k**

- **Regression Prediction**: Unlike classification, where the most common class of the 'k' neighbors is typically returned, k-NN regression averages the target values of these neighbors to determine the prediction. The average can be a simple arithmetic mean or a weighted mean if some neighbors are closer than others.

When to Consider Nearest Neighbors

- Instances map to points in R^d
- Less than 20 features per instance, typically normalized
- Sufficient training data

Advantages:

- Training is very fast
- Learn complex target functions
- Do not loose information

Disadvantages:

- Slow at query time
  - Presorting and indexing training samples into search trees reduces time
- Easily fooled by irrelevant features (attributes)

Source: Fenix, 2015

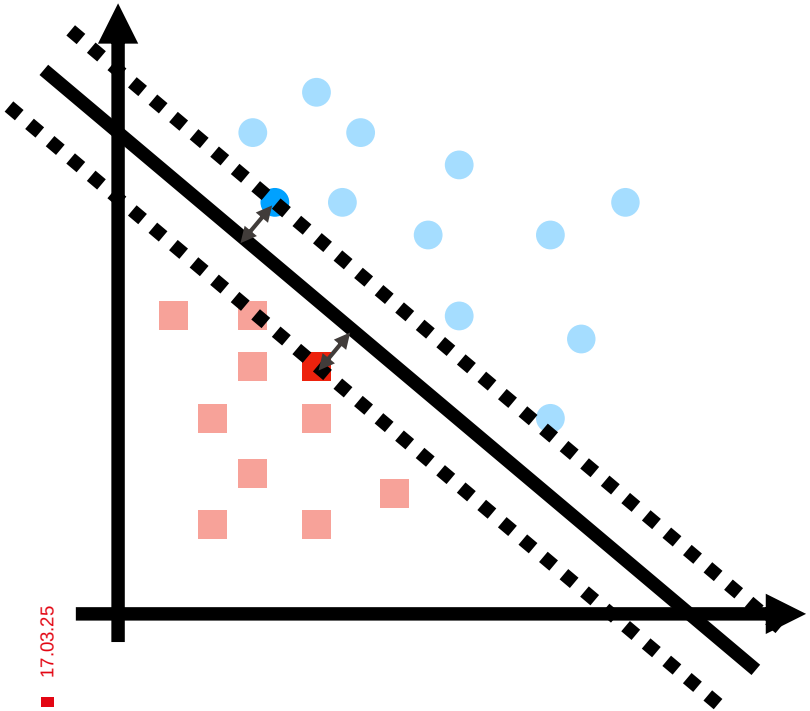
How unserupervised k-NN can work for anomaly detection

- **Data Collection & Preprocessing**
  - Collect sensor data (e.g., vibration, strain, displacement).
  - Extract meaningful features (RMS, peak amplitude, frequencies).
- **Build a Reference Dataset**
  - Use data from normal (healthy) operating conditions.
- **Apply k-NN Algorithm**
  - For each new data point, find its K nearest neighbors from the reference dataset.
  - Calculate the distance (Euclidean or other) to its neighbors.
- **Anomaly Detection Rule**
  - If the average distance is greater than a threshold, classify it as anomalous.
  - Threshold selection: Based on validation data or statistical methods.

Support Vector Machine

SVM

Optimal hyperplane separation



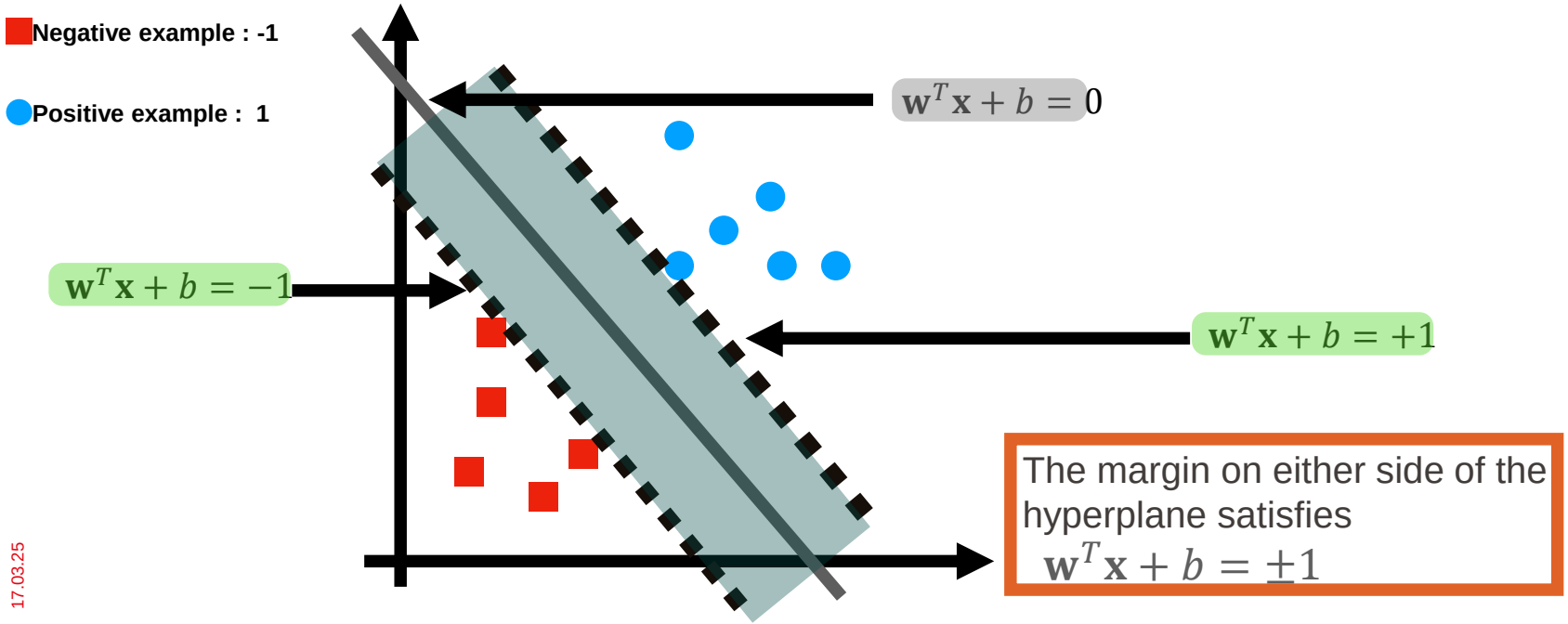
Classification Algorithm

Based on maximizing **margin**

Margin : Defined as the width that the boundary could increase by before hitting a datapoint

Optimal hyperplane separation

We obtain our constraints



SVM

Optimal hyperplane separation

We obtain an optimization problem:

$$\min_{\mathbf{w},b} \frac{\| \mathbf{w} \|^2}{2}$$

Objective

$$\left.\begin{array}{l} \mathbf{w}^T\mathbf{x}^{(i)} + b \geq 1 \text{ when } y^{(i)} = +1 \\ \mathbf{w}^T\mathbf{x}^{(i)} + b \leq -1 \text{ when } y^{(i)} = -1 \end{array}\right\} y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 \text{ when } i = 1,2,\ldots,M$$

Constraints

This is hard-margin SVM, and it work only for separable data

SVM

Optimal hyperplane separation

The margin between two classes is at least $\frac{2}{\|\mathbf{w}\|}$

We obtain our objective criteria

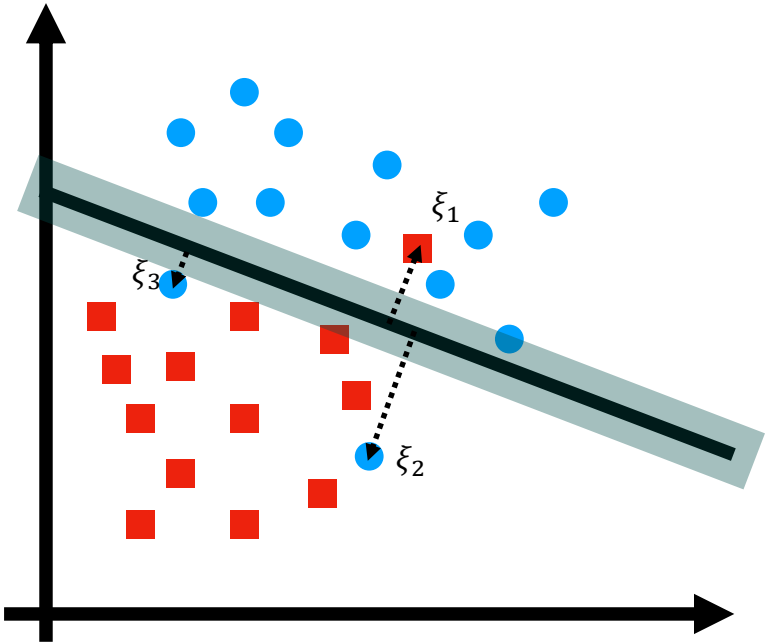
Maximizing this condition is equivalent to **minimizing** $\frac{\|\mathbf{w}\|}{2}$

Better to minimize a **square form** : $\frac{\|\mathbf{w}\|^2}{2}$

SVM

Dealing with non-separable data

What should we do ?



Constraints relaxed by slack variables ξ_i

$$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi_i s.t \xi_i \geq 0 \ \forall i = 1,2,\ldots,N$$

M : number of examples in the margin or misclassified

We need to add a penalty for too large slack variable

$$\min_{\mathbf{w},b}\left(\frac{\| \mathbf{w} \|^2}{2} + \frac{C}{N}\sum_{i=1}^{N}\xi_i\right)$$

C > 0 weight the influence of the penalty term

Find trade off between maximizing margin and minimizing the classification errors

SVM

Dealing with non-separable data

Control size of the margin

$$\min_{\mathbf{w},b}\left(\frac{\|\mathbf{w}\|^2}{2}+\frac{C}{N}\sum_{i=1}^{N}\xi_i\right)$$

Control number of misclassification

$$\left.\begin{aligned}\mathbf{w}^T\mathbf{x}^{(i)}+b &\geq 1-\xi_i \text{ when } y^{(i)}=+1\\ \mathbf{w}^T\mathbf{x}^{(i)}+b &\leq -1+\xi_i \text{ when } y^{(i)}=-1\end{aligned}\right\} y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b)\geq 1-\xi_i \text{ when } i=1,2,\ldots,M$$

Note that : $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b)\geq 1-\xi_i \Leftrightarrow \xi_i \geq 1-y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b)$

Because $\xi_i \geq 0$ **Note that :** $\xi_i = max(0,1-y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b))$

Dealing with non-separable data

Control size of the margin

$$\min_{\mathbf{w},b}\left(\frac{\|\mathbf{w}\|^2}{2}+\frac{C}{N}\sum_{i=1}^{N}\xi_i\right)$$

Control number of misclassification

We can reformulate into a form more adapted for learning as :

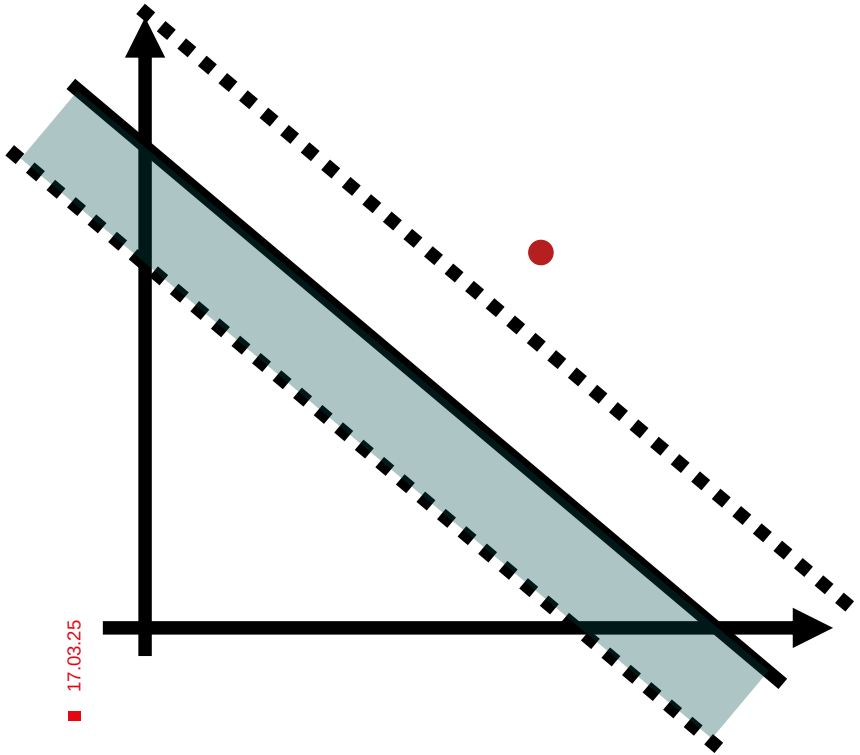
Hinge loss

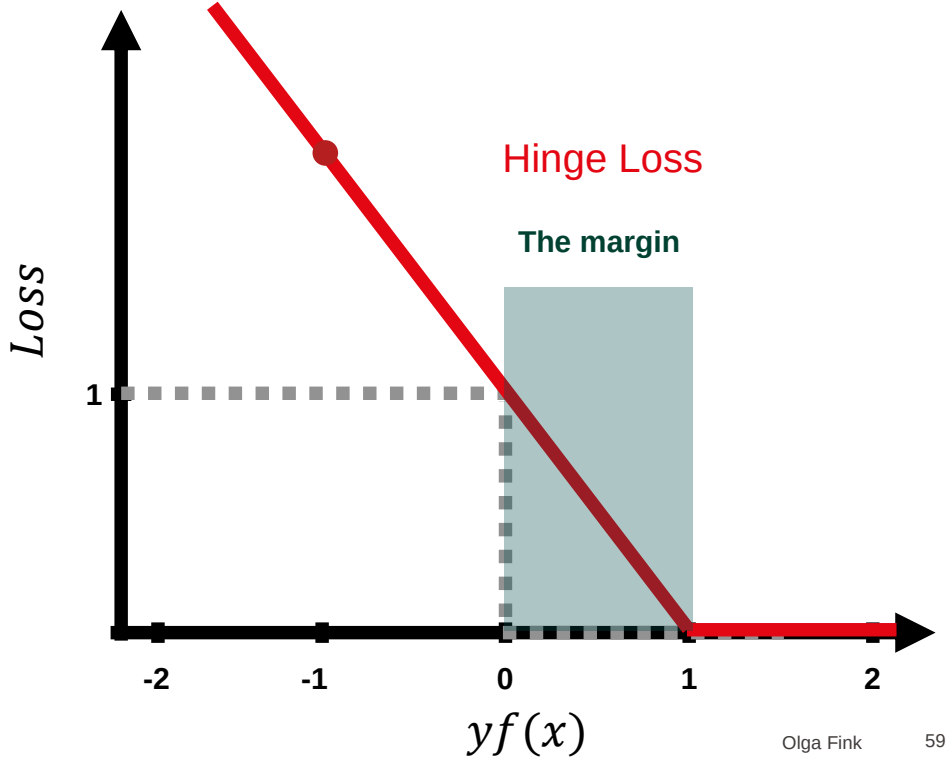
$$\min_{\mathbf{w}}\left(\sum_{n=1}^{N}max(0,1-y^{(n)}\mathbf{x}^{(n)^T}\mathbf{w})+\frac{\lambda}{2}\|\mathbf{w}\|^2\right)$$

Regularization term

Optimal hyperplane separation

Link with the Hinge function





SVM

Dealing with non-separable data

$$\min_{\mathbf{w}}\left(\sum_{n=1}^{N} max(0,1-y^{(n)}\mathbf{x}^{(n)^T}\mathbf{w}\right)$$

Leads to :

1. **A separating hypeplane**
2. **A scaling of $\mathbf{w}$ so that no point of the data is in the margin**

$$\min_{\mathbf{w}}\left(\frac{\lambda}{2}\parallel \mathbf{w} \parallel^2\right)$$

Guarantee that separating hyperplane and scaling for which the margin is the largest

SVM

Dealing with non-separable data

$$\min_{\mathbf{w}}\left(\sum_{n=1}^{N} max(0,1-y^{(n)}\mathbf{x}^{(n)^T}\mathbf{w}) + \frac{\lambda}{2}\parallel \mathbf{w} \parallel^2\right)$$

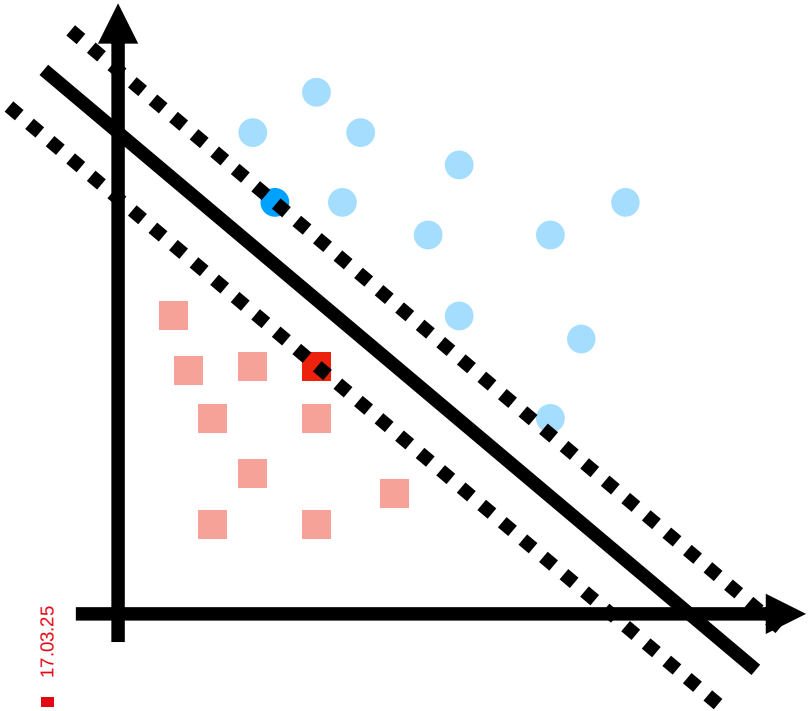
Note that this function is convex

So we can use SGD to optimize it efficiently

SVM

Dealing with non-separable data

$$\max_{\alpha \in \{0,1\}} \min_{\mathbf{w}} \left(\sum_{n=1}^{N} \alpha_n (1 - y^{(n)} \mathbf{x}^{(n)T} \mathbf{w}) + \frac{\lambda}{2} \| \mathbf{w} \|^2 \right)$$

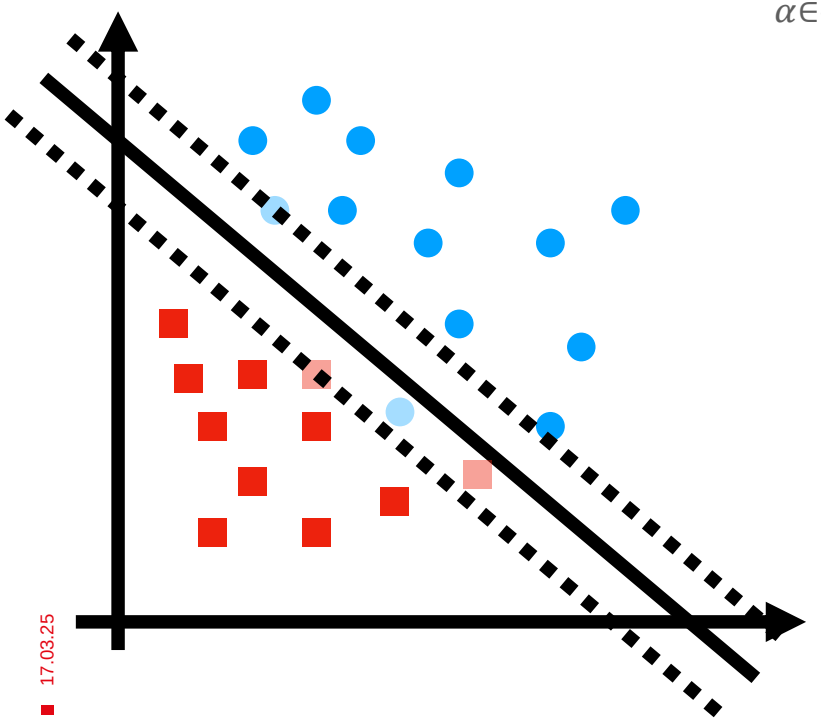


The α_n are the support vectors !

The support vectors are the **nearest samples** from the hyperplane

SVM

Dealing with non-separable data



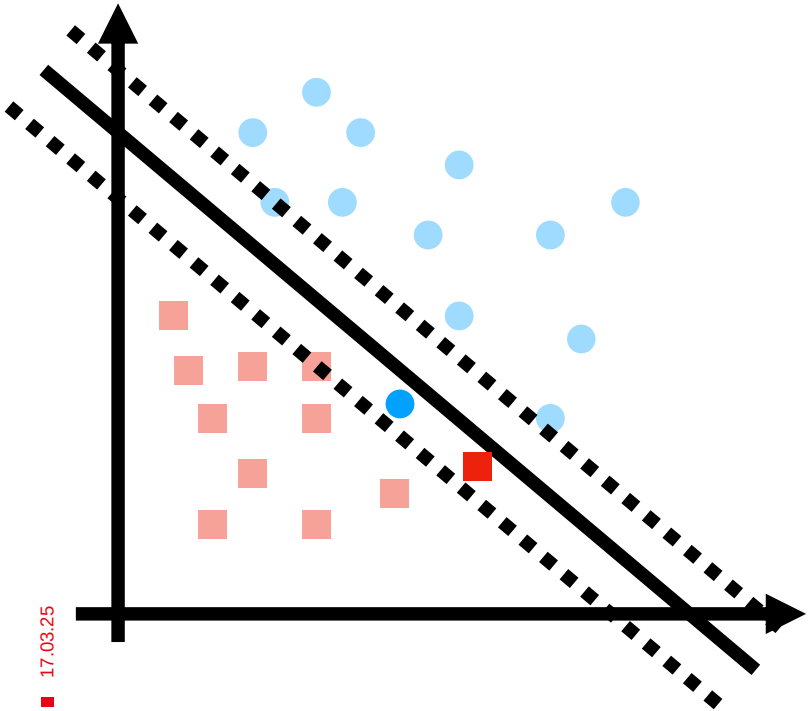
$$\max_{\alpha \in \{0,1\}} \min_{\mathbf{w}} \left(\sum_{n=1}^{N} \alpha_n (1 - y^{(n)} \mathbf{x}^{(n)T} \mathbf{w}) + \frac{\lambda}{2} \| \mathbf{w} \|^2 \right)$$

α_n = 0 when $1 - y^{(n)} \mathbf{x}^{(n)T} \mathbf{w} < 0$

Examples that lie on the correct side and outside of the margin

SVM

Dealing with non-separable data



$$\max_{\alpha \in \{0,1\}} \min_{\mathbf{w}} \left(\sum_{n=1}^{N} \alpha_n [1 - y^{(n)} \mathbf{x}^{(n)T} \mathbf{w}]_+ + \frac{\lambda}{2} \| \mathbf{w} \|^2 \right)$$

$$\alpha_n = 1 \text{ when } 1 - y^{(n)} \mathbf{x}^{(n)T} \mathbf{w} > 0$$

Examples that lie strictly inside the margin, or on the wrong side

The "Kernel Trick"

- The linear classifier relies on dot product between vectors $K(x_i,x_j)=x_i^T x_j$
- If every data point is mapped into high-dimensional space via some transformation Φ: $x \rightarrow \varphi(x)$, the dot product becomes:

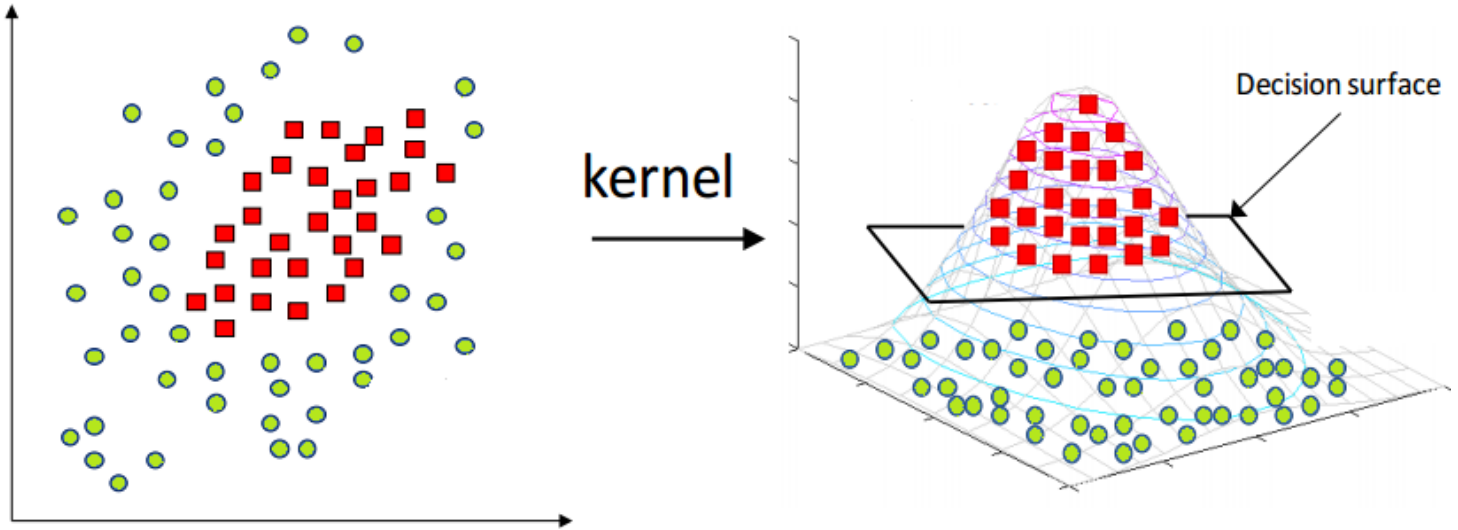
$$K(x_i,x_j)= \varphi(x_i)^T \varphi(x_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

SVM

Dealing with non-linear classification



SVM

Dealing with non-linear classification

$$\max_{\alpha \in [0,1]^N} \alpha^T \mathbf{1} - \frac{1}{2\lambda} \alpha^T \mathbf{Y}\mathbf{X}\mathbf{X}^T\mathbf{Y}\alpha$$

We saw that in this alternative formulation the data only enters in the form of a ''kernel'' $\mathbf{K} = \mathbf{X}\mathbf{X}^T$

$X: [N \times D]$ with N the number of samples and D the number of features

By definition, a ''kernel'' $\mathbf{K} = \mathbf{X}\mathbf{X}^T$ is equivalent to an **inner product**

There exists an **infinity of inner products** dependent to the **geometry of spaces** where we use it

SVM - Kernels

Dealing with non-linear classification

Kernel	Equation
Linear	$K(x,y) = xy$
Sigmoid	$K(x,y) = \tanh(axy + b)$
Polynomial	$K(x,y) = (1 + xy)^d$
RBF	$K(x,y) = a[\exp(\frac{\gamma}{\|\|x-y\|\|^2 + sigma^2}) - 1]$
KMOD	$K(x,y) = \exp(-a\|\|x-y\|\|^2)$
Exponential RBF	$K(x,y) = \exp(-a\|\|x-y\|\|)$

Constructing the Decision Function

Once the optimal α_i values are found, the decision function for a new point $\mathbf{x}$ can be constructed without explicitly computing $\mathbf{w}$. Instead, we use:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

The bias b can be computed using the support vectors, and conditions derived from the KKT conditions.

Properties of SVM

- Flexibility in choosing a similarity function
- Sparseness of solution when dealing with large data sets
  - only support vectors are used to specify the separating hyperplane
- Ability to handle large feature spaces
  - complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach
- Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution
- Feature Selection

SVM for multi-class classification

- Support Vector Machine (SVM) is fundamentally a binary classifier, which means that it's naturally suited for distinguishing between two classes. However, it can be extended for use in multi-class classification through several strategies:
- **One-vs-Rest (OvR):** Also known as one-vs-all, this strategy involves training a single SVM for each class, with the samples of that class as positive samples and all other samples as negatives. This results in as many classifiers as there are classes. For prediction, the classifier with the highest output function (distance from the decision boundary) is typically chosen as the class label.
- **One-vs-One (OvO):** This approach trains an SVM for every pair of classes. If there are N classes, this means training $\frac{N(N-1)}{2}$ classifiers. For a new input instance, each SVM votes for a class, and the class with the most votes determines the instance's label. Although OvO requires training more classifiers than OvR, each classifier only needs to be trained on the part of the training set for the two classes that it needs to separate.

Support Vector Regression (SVR)

- Support Vector Regression (SVR) applies the principles of Support Vector Machines (SVM) to regression problems. While SVM is used for classification tasks, SVR is designed to predict continuous values.
- The main idea behind SVR is similar to SVM, which is to find a function that has at most ε deviation from the actual target values of the training data and at the same time is as flat as possible.

Support Vector Regression (SVR)

- **Fitting the Best Line**: In SVR, instead of looking for a separating hyperplane as in SVM classification, the algorithm tries to fit the best line (in 2D) or hyperplane (in higher dimensions) within a threshold value, ε, which defines a tube around the estimated function.
- **Epsilon Tube (ε-Tube)**: This tube is the acceptable error margin; points inside this tube are not penalized, as their predicted values fall within a tolerable range of the actual values.
- **Objective**: The goal is to find the flattest tube so that for the training points outside this ε-tube, the distances to the tube boundaries are minimized. This is achieved by minimizing a function that represents the flatness of the tube, which is often the norm of the weights vector w**w**.
- **Loss Function**: SVR uses a special type of loss function called the ε-insensitive loss function which ignores errors that are within the distance ε of the true value. It only penalizes data points that fall outside of the ε-tube.
- **Support Vectors**: The data points that lie at the boundary of the ε-tube or outside it are the Support Vectors. They are the only points that influence the shape of the regression function.
- **Dual Form and Kernel Trick**: As with SVM, SVR can be formulated in a dual form which allows it to take advantage of the kernel trick for dealing with nonlinear data. This means that the linear regression function is mapped into a higher-dimensional space using a kernel function (like the radial basis function, polynomial, or sigmoid), where it is assumed that the data can be fitted with a linear function.
- **Optimization**: The fitting process involves a constrained optimization problem where you maximize the margin while penalizing points that fall outside the ε-tube.