Data-driven Decision Making Introduction to Machine Learning

João Pedro Pedroso

2024/2025

João Pedro Pedroso

Data-driven Decision Making

2024/2025

- Matching problems
- Kidney exchange problems

Today's class:

- Machine learning:
 - motivation
 - basics
- Fundamental algorithms
- Building Blocks of a Learning Algorithm:
 - loss function;
 - optimization criterion based on the loss function
 - optimization method to find a solution
 - basic practice

João Pedro Pedroso

æ

Introduction to Machine learning

Bibliography

- Machine Learning: essential notions We'll follow closely The Hundred-Page Machine Learning Book Andriy Burkov http://themlbook.com/wiki/doku.php
- Machine Learning in action: we'll pick some topics from Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Aurélien Géron O'Reilly Media, Inc
- Software:
 - Python
 - https://www.python.org
 - Scikit-learn https://scikit-learn.org
 - Google colab notebooks https://colab.research.google.com

The Hundred-Page Machine Learning Book: contents:

- Introduction
- 2 Notation and Definitions
- Fundamental Algorithms
- Anatomy of a Learning Algorithm
- Sasic Practice
- Neural Networks and Deep Learning
- Problems and Solutions
- Advanced Practice
- Onsupervised Learning
- Other Forms of Learning
- Conclusion

Machine Learning

- Algorithms that rely on a collection of examples of some phenomenon
 - nature
 - handcrafted by humans
 - generated by another algorithm
- Used for solving practical problems by
 - gathering a dataset
 - algorithmically building a statistical model based on that dataset
 - using the model to solve the problem

• Supervised learning

- based on a dataset with labeled examples
- these are used to train (learn) the model
- Semi-supervised learning
 - dataset has labeled and unlabeled examples
- Unsupervised learning
 - dataset has no labels
 - somehow based on *similarity* among examples
 - learn patterns or data groupings from (unlabeled) data
- Reinforcement learning
 - data is acquired from the environment
 - rewards for actions in given states

Supervised Learning

Dataset: collection of labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$

- each element $\mathbf{x}_i \rightarrow \text{feature vector}$
 - vector of dimension $j = 1, \dots, D$
 - each element \rightarrow value that somehow describes the example • feature $x^{(j)}$
- e.g., if each example x in our collection represents a person:
 - feature $x^{(1)}
 ightarrow$ height in cm
 - feature $x^{(2)}
 ightarrow$ weight in kg
 - ...
 - (position (j) represents the same feature in all examples)
- each label y_i can be:
 - element belonging to a finite set of classes $\{1, 2, \dots, C\}$
 - e.g., mail message spam or not_spam
 - real number
 - e.g., credit limit to a customer
 - more complex structure (vector, matrix, tree, graph)

< 回 > < 三 > < 三 > -

use dataset to produce a model

- input: feature vector **x**
- output: its label y
- example:
 - input: vector with elements of a person's health
 - weight, BMI, age, ...
 - output: probability that the person has cancer

Unsupervised Learning

- Dataset: collection of unlabeled examples $\{\mathbf{x}_i\}_{i=1}^N$
- each element $x_i \rightarrow \text{feature vector}$
- goal of unsupervised learning:
 - use dataset to produce a model
 - input: feature vector x
 - output: a value/vector
- examples:
 - clustering:
 - output: ID of cluster to which each x_i belongs
 - dimensionality reduction:
 - $\bullet\,$ output: selection of features from x
 - outlier detection:
 - output: real number indicating how x is different from a "typical" example in the dataset

- Dataset: collection of labeled and unlabeled examples
- Much more unlabeled than labeled
- Goal:
 - same as the goal of the supervised learning algorithm
 - hope: using many unlabeled examples helps finding a better model
 - leverage on additional information given by larger sample

- Machine "lives" in an environment
- State of environment: vector of features
- Machine can execute actions in every state
- Different actions \rightarrow different rewards \rightarrow may move machine to another state of the environment
- Goal: learn a policy
 - $\bullet~\mbox{policy} \to \mbox{function that}$
 - takes feature vector of a state as input
 - outputs an optimal action to execute in that state
 - action is optimal if it maximizes the expected average reward
- sequential decision making
- Iong-term goal:
 - game playing, robotics, resource management, logistics, ...

Gather data

- collection of pairs (input, output)
- input: email messages, pictures, sensor measurements, ...
- output:
 - (most common:) real numbers or labels (spam/not_spam, cat/dog/mouse, ...)
 - vectors (e.g., four coordinates of rectangle person on a picture)
 - sequences (e.g., ["adjective", "adjective", "noun"] for input "big beautiful car"
 - some other structure
- Decide on a learning algorithm
- Create a model

Supervised Learning At Work: example

- Our problem: spam detection
- Data: *e.g.*, 10,000 email messages, each with either label spam or not_spam
- Convert each email message into a feature vector e.g.: bag of words
 - to take dictionary of English words
 - say, 20,000 alphabetically sorted words
 - stipulate that each feature vector has 20,000 elements
 - element index j is
 - 1 if word index j in the dictionary is in message
 - 0 otherwise
 - compute this to each message \rightarrow
 - 10,000 feature vectors (each with dimension 20,000)
 - 10,000 labels <code>spam/not_spam</code> \rightarrow 1/0, or +1/-1
- Dataset now ready to use with a learning algorithm to compute a model
- *e.g.*, let the algorithm be *support vector machine* (SVM)

Supervised Learning At Work: example: SVM

- $\bullet\,$ feature vector $\to\,$ point in a high-dimensional space
 - in our example: 20,000-dimensional space
 - algorithm determines 19,999-dimensional hyperplane
 - separate examples with positive labels from examples with negative labels
 - boundary separating examples of different classes \rightarrow decision boundary
- equation of the hyperplane is given by two parameters:
 - real-valued vector \mathbf{w} (same dimensionality as \mathbf{x})
 - real number b; equation:

$$\mathbf{w}\mathbf{x} - b = 0$$

•
$$\mathbf{wx} \to w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + \ldots + w^{(D)}x^{(D)}$$

- $\bullet \ D \to {\rm dimension} \ {\rm of} \ {\rm feature} \ {\rm vector} \ {\bf x}$
- using our model: predict label of some input feature vector x

$$y = \operatorname{sign}(\mathbf{wx} - b)$$

• learning \Leftrightarrow determining **w** and b

- Objective of the SVM: find optimal values **w**^{*} and *b*^{*}
- SVM model:

$$f(\mathbf{x}) = \operatorname{sign}(\mathbf{w}^*\mathbf{x} - b^*)$$

- To predict whether an email message is spam or not:
 - take the text of the message
 - convert it into a feature vector
 - multiply this vector by **w**^{*}, subtract b^{*}
 - take the sign of the result
 - $+1
 ightarrow { t spam}$
 - $-1
 ightarrow \texttt{not_spam}$

Goal of the learning algorithm (SVM)

- How to find \mathbf{w}^*, b^* ?
 - solve an optimization problem
 - constraints: correctly predict labels of our i = 1, ..., 10,000 examples

$$\mathbf{wx}_i - b \ge +1$$
 if $y_i = +1$
 $\mathbf{wx}_i - b \le -1$ if $y_i = -1$

• objective: maximize margin \Leftrightarrow minimize $||\mathbf{w}|| = \sqrt{\sum_{j=1}^{D} (w^{(j)})^2}$

• Optimization model (*training*):

minimize $||\mathbf{w}||$ subject to $y_i(\mathbf{wx}_i - b) \ge +1$ for i = 1, ...



• can also incorporate kernels

- make the decision boundary arbitrarily non-linear
- usual version of SVM incorporates a *penalty hyperparameter*
 - what to pay in the objective for the misclassification of training examples

• hyperparameter:

- property of a learning algorithm
- often, a numerical value
- influence the way the algorithm works
- hyperparameter aren't learned by the algorithm
 - must be set by data analyst before running the algorithm

- Any classification learning algorithm that builds a model creates a decision boundary
 - implicitly or explicitly
- Can be straight, curved, have a complex form, ...
- Form of the decision boundary determines the accuracy of the model
 - ratio of examples whose labels are predicted correctly
- Different learning algorithms:
 - different decision boundaries
 - different speed of model building and prediction processing time

We'll follow notation used in

The Hundred-Page Machine Learning Book:

- Data structures:
 - scalars: italic letters x, y, z
 - vectors: boldface letters **x**, **y**, **z**
 - $\mathbf{x}_i \rightarrow \text{vector index } i \text{ (on dataset)}$
 - elements of a vector $\mathbf{x} : (x^{(1)}, x^{(2)}, \ldots)$
 - matrices: bold capital letters X, Y, Z
 - sets: calligrafic letters $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$
- 2 Random variables
 - denoted with capital letter X, Y, Z
 - $f_X \rightarrow$ probability density function of X
 - "examples" \rightarrow observations of X
 - "dataset" \rightarrow collection of examples (*i.e.*, a sample) $S_X = \{x_i\}_{i=1}^N$

- hyperparameters \rightarrow properties of a learning algorithm
 - influence the way the algorithm works
 - not learned by the algorithm from data
 - set by the data analyst before running the algorithm
- $parameters \rightarrow$ define the model learned
 - directly modified by the learning algorithm based on the training data
 - goal of learning: finding parameters that make the model "optimal"

Classification vs. Regression

- Classification: problem of automatically assigning a label to an unlabeled example
 - e.g., spam detection
 - learning algorithm:
 - takes a collection of labeled examples as inputs
 - produces a model:
 - \rightarrow take as input an unlabeled example
 - \rightarrow output a label (or a label identifier)
 - binary classification: two classes
 - multiclass classification: 3 or more classes
- Regression: problem of predicting a real-valued label for an unlabeled example
 - e.g., estimating house price valuation based on house features
 - area, number of bedrooms, location, ...
 - learning algorithm:
 - takes a collection of labeled examples as inputs
 - produces a model:
 - \rightarrow take as input an unlabeled example
 - \rightarrow output a target

- Most supervised learning algorithms are model-based
 - *e.g.*, SVM
 - use the training data to create a model that has parameters learned from training data
 - in SVM: \mathbf{w}^*, b^*
 - after the model is built, training data can be discarded
- Instance-based learning algorithms:
 - use the whole dataset as the model
 - e.g., k-Nearest Neighbors (kNN)
 - look at neighborhood of input example (in the space of feature vectors)
 - output:
 - \rightarrow classification: label most often seen
 - \rightarrow regression: average target of neighbors



2024/2025

ヘロト 人間 とくほとくほど

æ

• Shallow learning algorithm:

- learns the parameters of the model directly from the features of the training examples
- most supervised learning algorithms are shallow
- exceptions: are neural networks
- Deep learning algorithm:
 - deep neural networks
 - more than one layer between input and output
 - most model parameters:
 - not directly learned from the features of training examples
 - learned from the outputs of the preceding layers



2024/2025

イロト 不得下 イヨト イヨト

æ

Model is a linear combination of input features

- Problem statement:
 - input: labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$
 - $N \rightarrow$ size of the collection
 - $\mathbf{x}_i \in \mathbb{R}^D \to D$ -dimensional vector i
 - $y_i \in \mathbb{R} \to \text{target value}$
 - build model $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}\mathbf{x} + b$
 - $\mathbf{w} \in \mathbb{R}^D \to D$ -dimensional vector of parameters
 - $b \in \mathbb{R}$
 - f is parametrized by **w** and b
 - model used to predict the unknown y for a given x

• $y \leftarrow f_{\mathbf{w},b}(\mathbf{x})$

- find optimal values \mathbf{w}^*, b^*
 - make the most accurate predictions on data



João Pedro Pedroso

Data-driven Decision Making

Linear Regression

• Minimize sum of squared errors:

$$\frac{1}{N}\sum_{i=1,\ldots,N}(f_{\mathbf{w},b}(\mathbf{x}_i)-y_i)^2$$

- objective function/cost function
- loss function: $(f_{\mathbf{w},b}(\mathbf{x}_i) y_i)^2$
 - squared error loss/mean squared error (MSE)
- Model-based learning: minimize loss function
- Linear regression: cost function is average loss
 - empirical risk
 - average of all penalties obtained by applying the model to the training data
 - quadratic vs absolute value \rightarrow different models
 - quadratic is convenient: derivatives
 - optimum \rightarrow gradient set to zero
- Extensions:
 - use a higher degree polynomial instead of linear combination
 - overfitting:
 - model predicts very well labels of examples used for training 2024/2025

João Pedro Pedroso

Data-driven Decision Making

Overfitting



more powerful models ⇒ higher risk of overfitting

João Pedro Pedroso

Data-driven Decision Making

2024/2025 31/61

- Actually, logistic regression is a *classification learning algorithm*
- Name comes from statistics
 - mathematical formulation is similar to that of linear regression
- Sigmoid/Standard logistic function:

$$\frac{1}{1+e^{-x}}$$

• Logistic Regression model

$$f_{\mathbf{w},b}(\mathbf{x}) \stackrel{\mathrm{def}}{=} rac{1}{1+e^{-(\mathbf{w}\mathbf{x}+b)}}$$

Standard logistic function



João Pedro Pedroso

Data-driven Decision Making

2024/2025

Logistic Regression

- Interpretation: $f(\mathbf{x})$ as the probability of y_i being positive
- How to find optimal **w**^{*}, b^{*} ?
 - maximize likelihood of our training set according to the model
 - how likely an observation (example) is
- For instance:
 - labeled example (\mathbf{x}_i, y_i) in training data
 - model: arbitrary $\hat{\mathbf{w}}, \hat{b}$
 - applying to $\mathbf{x}_i : \ f_{\hat{\mathbf{w}},\hat{b}}(\mathbf{x}_i) \rightarrow \text{output } 0$
 - if y_i is the positive class:
 - likelihood of y_i being the positive class given by p
 - if y_i is the negative class:
 - likelihood of y_i being the negative class given by 1 p

Optimization criterion in logistic regression: maximum likelihood

• Likelihood of the training data:

$$L_{\mathbf{w},b} \stackrel{\text{def}}{=} \prod_{i=1...N} f_{\mathbf{w},b}(\mathbf{x}_i)^{y_i} (1 - f_{\mathbf{w},b}(\mathbf{x}_i))^{1-y_i}$$

- each term in the product:
 - $f_{\mathbf{w},b}(\mathbf{x}_i)$ when $y_i = 1$
 - $1 f_{\mathbf{w},b}(\mathbf{x}_i)$ otherwise
- in practice, to avoid overflow in the exponential, it's more convenient to maximize log-likelihood:

$$\begin{array}{l} \text{LogL}_{\mathbf{w},b} \stackrel{\text{def}}{=\!\!=} \log L_{\mathbf{w},b} \\ \stackrel{\text{def}}{=\!\!=} \sum_{i=1\dots N} [y_i \log f_{\mathbf{w},b}(\mathbf{x}_i) + (1-y_i) \log (1-f_{\mathbf{w},b}(\mathbf{x}_i))] \end{array}$$

• no closed form solution \rightarrow use, e.g., gradient descent to find optimum

- Classification learning algorithm
- Logistic Regression model

$$f_{\mathbf{w},b}(\mathbf{x}) \stackrel{\mathrm{def}}{=} rac{1}{1+e^{-(\mathbf{w}\mathbf{x}+b)}}$$

f_{w,b}(x) → likelihood of x being on positive class
 1 - f_{w,b}(x) → likelihood of x being on negative class

Decision Tree Learning

æ

Decision Tree Learning

- Decision tree: acyclic graph used to make decisions
 - each branching node examines a specific feature j
 - $\bullet\,$ if value below specific threshold $\Rightarrow\,$ follow left branch
 - $\bullet \ \ {\rm otherwise} \Rightarrow {\rm follow} \ {\rm right} \ {\rm branch}$
 - leaf node \rightarrow decision: class to which the example belongs
- Problem statement:
 - input: labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$
 - labels $y_i \in \{0, 1\}$
- Solution: several possibilities, choosing ID3
 - average log-likelihood:

$$\frac{1}{N}\sum_{i=1...N} [y_i \log f_{\mathsf{ID3}}(\mathbf{x}_i) + (1-y_i) \log (1-f_{\mathsf{ID3}}(\mathbf{x}_i))]$$

- $f_{\text{ID3}}(\mathbf{x}_i) \rightarrow \text{decision tree model}$
- ID3 constructs approximate nonparametric model

$$f_{\mathsf{ID3}}(\mathbf{x}) \stackrel{\mathrm{def}}{=\!\!=} \mathsf{Pr}(y=1|\mathbf{x})$$

• Initially: decision tree only has a start node containing all examples

$$\mathcal{S} \stackrel{\mathrm{def}}{=\!\!=} \{(\mathbf{x}_i, y_i)\}_{i=1}^N$$

• Initial (constant) model:

$$f_{\mathsf{ID3}}^{\mathcal{S}} \stackrel{\mathrm{def}}{=\!\!=} \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x}, y) \in \mathcal{S}} y$$

• same prediction $f_{\text{ID3}}^{\mathcal{S}}(\mathbf{x})$ for any input \mathbf{x}

ID3 steps (cont.)

- Then, search through:
 - all features $j = 1, \ldots, D$
 - all thresholds t
 - and split the set S into two subsets

$$\begin{split} \mathcal{S}_{-} & \stackrel{ ext{def}}{=} \{(\mathbf{x},y) | (\mathbf{x},y) \in \mathcal{S}, x^{(j)} < t\} \ \mathcal{S}_{+} & \stackrel{ ext{def}}{=} \{(\mathbf{x},y) | (\mathbf{x},y) \in \mathcal{S}, x^{(j)} \geq t\} \end{split}$$

- $\bullet \ \ {\rm new \ subsets} \to {\rm two \ new \ leaf \ nodes}$
- evaluate for all possible pairs (j, t) "how good" split with pieces S₋ and S₊ is
- finally:
 - ightarrow pick the best such values (j, t)
 - \rightarrow split ${\mathcal S}$ into ${\mathcal S}_+$ and ${\mathcal S}_-$
 - \rightarrow form two new leaf nodes
 - \rightarrow continue recursively on \mathcal{S}_+ and \mathcal{S}_-
- Quit when no split produces a model that's sufficiently better than the current one

João Pedro Pedroso

2024/2025

$$\mathbf{x}$$

$$S = \{ (\mathbf{x}_{1}, y_{1}), (\mathbf{x}_{2}, y_{2}), (\mathbf{x}_{3}, y_{3}), (\mathbf{x}_{4}, y_{4}), (\mathbf{x}_{5}, y_{5}), (\mathbf{x}_{6}, y_{6}), (\mathbf{x}_{7}, y_{7}), (\mathbf{x}_{8}, y_{8}), (\mathbf{x}_{9}, y_{9}), (\mathbf{x}_{10}, y_{10}), (\mathbf{x}_{11}, y_{11}), (\mathbf{x}_{12}, y_{12}) \}$$

$$Pr(y = 1 | \mathbf{x}) = (y_{1} + y_{2} + y_{3} + y_{4} + y_{5} + y_{6} + y_{7} + y_{8} + y_{9} + y_{10} + y_{11} + y_{12})/12$$

$$Pr(y = 1 | \mathbf{x})$$



2024/2025

(日)

3

• Goodness of a split: estimated by using entropy criterion

- measures uncertainty about a random variable
- reaches maximum when all values of the random variables are equiprobable
- reaches minimum when random variable can have only one value
- Entropy of a set of examples S:

$$H(\mathcal{S}) \stackrel{\text{def}}{=} -f_{\mathsf{ID3}}^{\mathcal{S}} \mathsf{log}(f_{\mathsf{ID3}}^{\mathcal{S}}) - (1 - f_{\mathsf{ID3}}^{\mathcal{S}}) \operatorname{log}(1 - f_{\mathsf{ID3}}^{\mathcal{S}})$$

• Entropy of a split $H(S_-, S_+)$: weighted sum of two entropies:

$$H(\mathcal{S}_{-},\mathcal{S}_{+}) \stackrel{\text{def}}{=} rac{|\mathcal{S}_{-}|}{|\mathcal{S}|} H(\mathcal{S}_{-}) + rac{|\mathcal{S}_{+}|}{|\mathcal{S}|} H(\mathcal{S}_{+})$$

ID3: stopping criteria

- At each step, at each leaf node:
 - find a split that minimizes entropy
 - or stop at this leaf node
- Criteria for stopping at a leaf node:
 - all examples there are correctly classified
 - we cannot find an attribute to split upon
 - $\bullet\,$ best split reduces the entropy less than some given $\epsilon\,$
 - tree reaches some given maximum depth d
- Hyperparameters ϵ , d to be found experimentally
- Widely used formulation improving ID3: C4.5
 - accepts both continuous and discrete features
 - handles incomplete examples
 - solves overfitting problem by using a bottom-up technique known as "pruning"
 - go through the tree once it's been created
 - $\bullet\,$ branches that don't contribute significantly to the error reduction $\rightarrow\,$ replaced by leaf nodes





Decision surface of a decision tree using paired features

∃ →

< 4 ₽ >

More about SVM:

- What if there's noise in the data and no hyperplane can perfectly separate positive examples from negative ones?
- What if the data cannot be separated using a plane, but could be separated by a higher-order polynomial?





For convenience, objective is squared:

minimize
$$\frac{1}{2} ||\mathbf{w}||^2$$

subject to: $\mathbf{w}\mathbf{x}_i - b \ge +1$ if $y_i = +1$
 $\mathbf{w}\mathbf{x}_i - b \le -1$ if $y_i = -1i = 1, \dots, N$

• Constraints can be written as $y_i(\mathbf{wx}_i - b) \geq +1$

- hard margin SVM
- We want to make them "soft": allow violations, at a cost in the objective
 - for dealing with noise
- Hinge loss function: $max(0, 1 y_i(wx_i b))$
 - zero if constraints are satisfied
 - proportional to the distance to decision boundary if not

• New objective: soft-margin SVM

minimize
$$C||\mathbf{w}||^2 + \frac{1}{N}\sum_{i=1}^N \max(0, 1 - y_i(\mathbf{wx}_i - b))$$

- Hyperparameter C: tradeoff between
 - increasing the size of the decision boundary
 - ensuring that each \mathbf{x}_i lies on the correct side of the decision boundary
 - value for C:
 - $\bullet \ \ high \rightarrow ignore \ misclassification$
 - $\bullet~\mbox{low}$ \rightarrow classification errors more costly
 - must be chosen experimentally



SVM: dealing with inherent non-linearity



- transform the original space into a space of higher dimensionality
- hope: examples will become linearly separable in this space
- SVMs can use a function to implicitly doing this: kernel trick



SVM: kernel trick

• Transform *feature space* into a higher dimentionality space: $\phi : \mathbf{x} \mapsto \phi(\mathbf{x})$

• e.g.,
$$\phi([q,p]) \stackrel{\text{def}}{=} (q^2, \sqrt{2}qp, p^2)$$

- Problem: we don't know a priori which mapping would work for our data
- Kernel functions:
 - tool to efficiently work in higher-dimensional spaces
 - no need to do this transformation explicitly
- Application in SVMs:
 - using the method of Lagrange multipliers
 - $\alpha_i \rightarrow$ multiplier associated to example *i*
 - optimization problem becomes:

$$\max_{\alpha_1,\ldots,\alpha_N} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N y_i \alpha_i (\mathbf{x}_i \mathbf{x}_k) y_k \alpha_k$$

subject to:
$$\sum_{i=1}^N \alpha_i y_i = 0, \alpha_i \ge 0 \qquad \qquad i = 1, \ldots, N$$

SVM: kernel trick

$$\max_{\alpha_1,...,\alpha_N} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N y_i \alpha_i (\mathbf{x}_i \mathbf{x}_k) y_k \alpha_k$$

subject to:
$$\sum_{i=1}^N \alpha_i y_i = 0, \alpha_i \ge 0 \qquad \qquad i = 1, \dots, N$$

- convex quadratic optimization problem
 - efficiently solvable
- term $(\mathbf{x}_i \mathbf{x}_k)$
 - only place where the feature vectors are used
 - transform vector space into higher dimensional space:
 - transform \mathbf{x}_i into $\phi(\mathbf{x}_i)$ and \mathbf{x}_k into $\phi(\mathbf{x}_k)$
 - then multiply $\phi(\mathbf{x}_i)$ by $\phi(\mathbf{x}_k)$
 - $\bullet \ \rightarrow \textit{would be very costly}$
- we are only interested in the result of the dot-product $(x_i x_k)$
 - real number
 - replace that by operation on original feature vectors with the same result

João Pedro Pedroso

Data-driven Decision Making

2024/2025

SVM: kernel examples

• previous example: quadratic kernel

- transforming q_1, p_1 into $\left(q_1^2, \sqrt{2}q_1p_1, p_1^2\right)$
- transforming q_2, p_2 into $(q_2^2, \sqrt{2}q_2p_2, p_2^2)$
- computing dot-product $(q_1^2, \sqrt{2}q_1p_1, p_1^2) \cdot (q_2^2, \sqrt{2}q_2p_2, p_2^2)$
- yields the same result as:
 - dot-product between (q_1,p_1) and $(q_2,p_2)
 ightarrow (q_1q_2+p_1p_2)$
 - then square it
 - same result $(q_1^2q_2^2 + 2q_1q_2p_1p_2 + p_1^2p_2^2)$
- quadratic kernel: $k(\mathbf{x}_1, \mathbf{x}_j) \stackrel{\text{def}}{=} (\mathbf{x}_i \mathbf{x}_k)^2$

most widely used: radial basis function, RBF kernel

$$k(\mathbf{x}, \mathbf{x}') \stackrel{\text{def}}{=} \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right)$$

 $\bullet \ ||\textbf{x}-\textbf{x}'||^2 \rightarrow \textit{Euclidean distance}$

- infinite number of dimensions (series expansion of exp)
- hyperparameter $\sigma \to {\rm controls}$ how smooth or curvy decision boundary is in the original space

João Pedro Pedroso

k-Nearest Neighbors

- Non-parametric learning algorithm
- kNN keeps all training examples in memory
- Once a previously unseen example x comes in:
 - algorithm finds k training examples closest to x
 - returns:
 - $\bullet \ \ classification \rightarrow majority \ label$
 - regression \rightarrow average label
- Closeness of two examples: given by a distance function
- Popular choices:
 - Euclidean distance
 - Negative cosine similarity
 - angle between two vectors
 - Chebychev distance
 - Hamming distance
 - ...
- Distance metric could also be *learned from data*



5 basic algorithms:

- Linear Regression
- Logistic Regression
- Decision Tree Learning
- Support Vector Machine
- k-Nearest Neighbors