50.021 – AI

Alex

Week 01: Ordinary Least Squares (Linear regression)

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

Learning goals

- See how the linear model can be used for regression and classification
- be able to reproduce the explicit solution for linear regression and ridge regression
- be able to explain how the set of constant points for $\{x : f(x) = w \cdot x = c\}$ looks like for a linear model depending on w and b as parameters
- l2-loss for regression, 0-1-loss and hinge-loss for classification
- a first insight into generalization as one goal of training machine learning models
- extending linear regression by basis functions (polynomial, RBF): you should be able to reproduce their formulas and the reasons for using them

1 Ordinary Least Squares (Linear regression)

Lets go through the following information for regression:

- I. Input and output space?
- II. the prediction mapping / model
- III. the loss function used to train the model

- IV. the goal in regression - what means generalization
- V. what possible functions does the model represent ?
- VI. a method to select a prediction mapping $g_{w,b}$ / or its parameters from a training dataset
- VII. from linear regression to ridge regression

1.1 I. Input and output space?

Some examples:
A. `http://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set`

$X1$ = the transaction date

$X2$ = the house age (unit: year)

$X3$ = the distance to the nearest MRT station (unit: meter)

$X4$ = the number of convenience stores in the living circle on foot (integer)

$X5$ = the geographic coordinate, latitude. (unit: degree)

$X6$ = the geographic coordinate, longitude. (unit: degree)

- The output is as follows Y = house price per unit of area

B. `http://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data`

$X1$ = Dew Point

$X2$ = Temperature

$X3$ = Pressure

$X4$ = Combined wind direction

$X5$ = Cumulated wind speed

$X6$ = Cumulated hours of snow

$X7$ = Cumulated hours of rain

- The output is as follows Y = PM2.5 concentration ($\mu g/m^3$)

C. `http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime`
(`http://archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test` `https://en.wikipedia.org/wiki/Concrete_slump_test`)
(`http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset`)

Input and output space in regression problems?

$\mathcal{X} =?, \mathcal{Y} =?$

1.2 II. the prediction mapping?

Linear function without/with a bias

$$f_w(x) = x \cdot w \qquad = \sum_{d=1}^{D} x_d w_d \qquad , \; w \in \mathbb{R}^{D \times 1}$$

$$g_{w,b}(x) = x \cdot w + b \qquad = \sum_{d=1}^{D} x_d w_d + b \qquad , \; w \in \mathbb{R}^{D \times 1}, b \in \mathbb{R}^1$$

1.3 III. Loss function for regression

Loss function for a pair (x, y):

$$L(f(x), y) = (f(x) - y)^2$$

Reasons:

- capture deviations on both sides of the real ground truth value y
- simple derivative

1.4 IV. the goal in regression

Find parameters w, b which generalize well to new, unseen data points. What does Generalize well to new, unseen data points mean????

The idea is the following: you assume that you can draw from your data source test data sets T_n. A test data set consists of n pairs (x_i, y_i). x_i is the input feature, y_i is the ground truth label to it. That is y_i is the regression value which x_i is expected to have.

You cannot predict which samples (x_i, y_i) you will obtain from your data source for new, unseen data points. Therefore: We can model the uncertainty by drawing from a probability for the (x_i, y_i):

$$(x_i, y_i) \sim P_{test}.$$

An Example for such a generating probability
for input samples $x \in [0, 1] \times [0, 1] \subset \mathbb{R}^2$:

$$\begin{aligned} p(x) &= Unif([0, 1] \times [0, 1]) \\ p(y|x) &= Normal(\mu(x), \sigma^2 = 0.1) \\ \mu(x) &= 2x_1 - 3x_2 \\ P_{test}(x, y) &= p(x)p(y|x) \end{aligned}$$

Drawing in practice

- draw $x \sim p(x)$
- draw $y \sim p(y|x)$
- this implies: have $(x, y) \sim P_{test}$

By drawing n times in this way, one can obtain a training dataset D_n or a test dataset T_n of independent samples.

Generalization for regression with l2-loss

Generalize well $\leftrightarrow$??? Suppose you can draw from your data source K times test data sets $T_n^{(k)}, k = 1, \ldots, K$, each of them has n samples . Then good generalization means:

1. we find parameters on a training set,

$$(w^*, b^*) = \text{argmin}_{(w,b)} \hat{L}(f, D_n) = \text{argmin}_{(w,b)} \frac{1}{n} \sum_{(x_i,y_i)\in D_n} L(f(x_i), y_i)$$

$$= \text{argmin}_{(w,b)} \frac{1}{n} \sum_{(x_i,y_i)\in D_n} (f(x_i) - y_i)^2$$

2. with the goal that for most of K test data sets (more formally: with high probability over the set of all test datasets of size n), the averaged loss $aL(f, T_n^{(k)})$ on the test dataset is low.

$$\hat{L}(f, T_n^{(k)}) = \frac{1}{n} \sum_{(x_i,y_i)\in T_n^{(k)}} L(f(x_i), y_i)$$

$$= \frac{1}{n} \sum_{(x_i,y_i)\in T_n^{(k)}} (f(x_i) - y_i)^2$$

We will formalize this two lectures later on – to generalize means to find a mapping f^* (or parameters (w^*, b^*) which define a mapping f^*) which achieves a low expected loss under the probability of the test samples.

$$f^* \approx \text{argmin}_{f\in\mathcal{F}} E_{(x,y)\sim P_{test}}[L(f^*(x), y)]$$

If this expectation is low, then by the law of large numbers, for most datasets of size n the averaged loss on the dataset will be close to that expectation, and therefore low as well:

$$E_{(x,y)\sim P_{test}}[L(f^*(x), y)] \approx \frac{1}{n} \sum_{(x_i,y_i)\in T_n^{(k)}} L(f(x_i), y_i)$$

Question: Why the goal is not: to have low loss on a single test dataset $T_{50} = \{(x_i, y_i), i = 1, \ldots, 50\}$?

1.5 V. What does a linear mapping represent?

Goal: understand what the mapping does.

A. What is the set of points x: $g_{w,b}(x) = 0$?

B. What is the set of points x the prediction is a constant c, that is $g_{w,b}(x) = c$?

1.5.1 What is the set of points x: $g_{w,b}(x) = 0$?

$$\begin{aligned} & g_{w,b}(x) && = 0 \\ \Leftrightarrow & x \cdot w = -b \end{aligned}$$

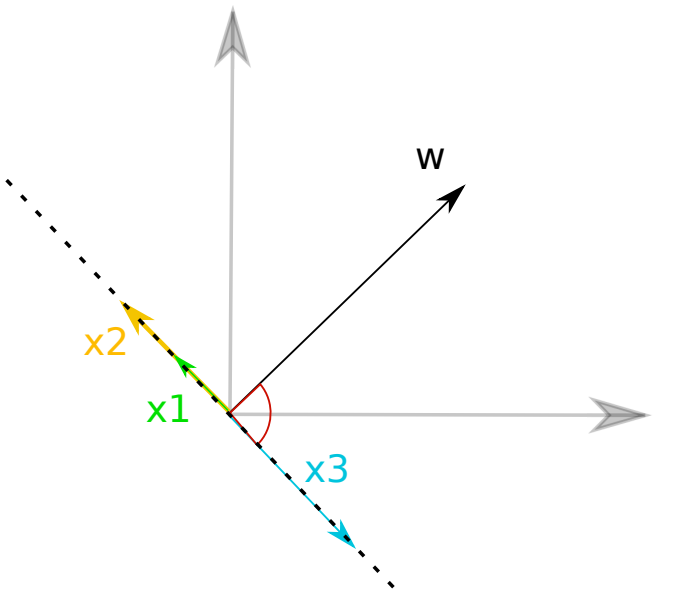
In order to understand how the bias b influences the zero set, lets consider three cases:

- $b = 0$
- $b > 0$
- $b < 0$

The case $b = 0$ We know that for $b = 0$: $g_{w,0}(x) = w \cdot x = 0$ holds for the zero vector $x = 0$.

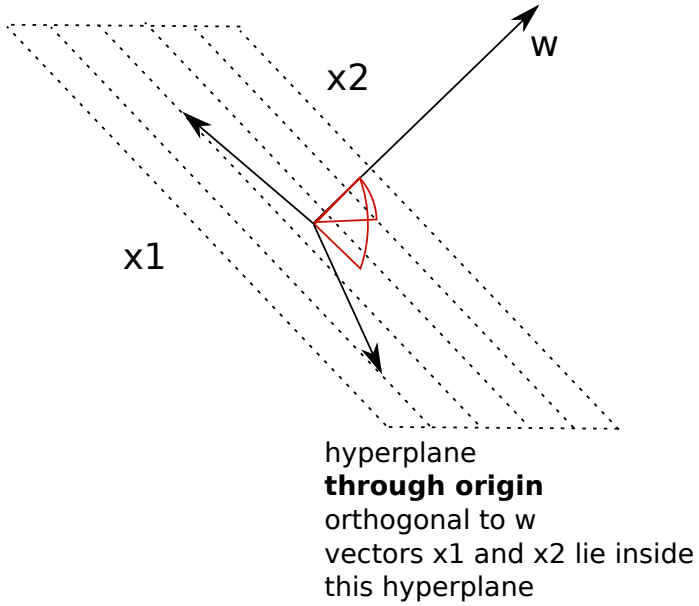
$$x \cdot w = 0$$

The set of points x such that the inner product $x \cdot w = 0$ is in 2 dims a one-dimensional line, which goes through the origin $(x_1, x_2) = (0, 0)$, and which is orthogonal to w.



x_1,x_2,x_3 are all orthogonal to the vector w

The analogy also holds for 3 or more dimensions. So for 3 dims it is a two-dimensional plane, which goes through the origin $(x_1, x_2, x_3) = (0, 0, 0)$.

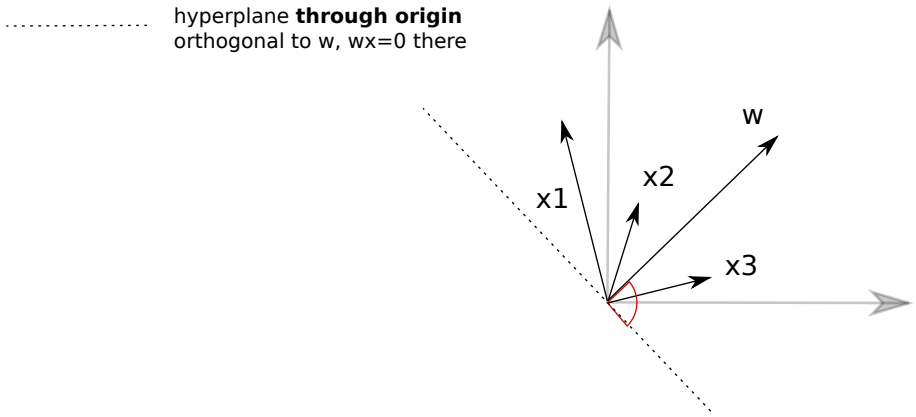


For n dimensions the plane of orthogonal vectors has $n-1$ dimensions (+ goes through the origin), but is still a hyperplane (that is if $x_1 \in P, x_2 \in P \Rightarrow a_1 x_1 + a_2 x_2 \in P$, and P can be represented as a linear combination of $n-1$ basis vectors).

The case $b < 0$

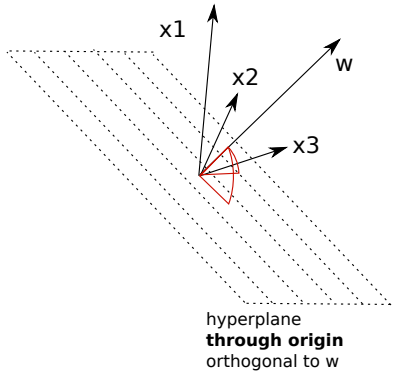
$$b < 0,\ w \cdot x + b = 0 \Rightarrow w \cdot x = -b > 0$$

We know: $w \cdot x > 0$ for all points x that are on that side of the **hyperplane through the origin**, in which w points to.



x_1,x_2,x_3 all have $w \cdot x_i > 0$ because relative to the hyperplane orthogonal to w which goes through the origin, they all point into the direction of w

The same also holds for 3 or more dimensions.



x_1,x_2,x_3 all have $w \cdot x_i > 0$ because relative to the hyperplane orthogonal to w which goes through the origin, they all point into the direction of w

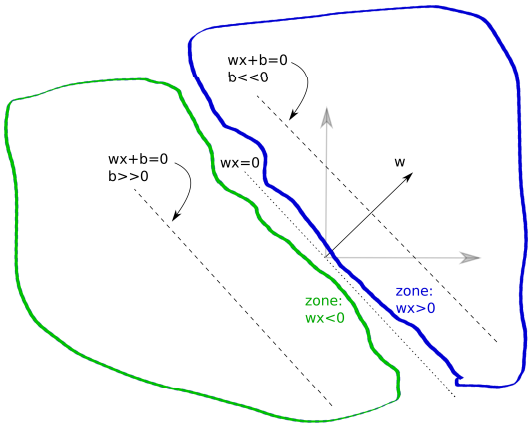
What does that mean? All the above vectors solve $wx + b = 0$ for some bias $b < 0$!

The case $b > 0$
The analogous thought results in the understanding, that set of points x such that

$$\{x : \ wx + b = 0\}$$

is a hyperplane which is parallel to the hyperplane $\{x : x \cdot u = 0\}$ orthogonal to w going through the origin, and which is shifted opposite to the direction of w

As the next graphic shows, the bias b shifts the zero set corresponding to $w \cdot x + b = 0$ anti-parallel to the direction of w.



hyperplane dependency on bias b

- Negative $b < 0$ shift the hyperplane $\{x : wx + b = 0\}$ into the direction of w,
- positive $b > 0$ shift the hyperplane $\{x : wx + b = 0\}$ against the direction of w.
- Large values of $|b|$ shift it away quite far.

hyperplane explicit

As a summary: the linear mapping $g(x) = w \cdot x + b$ has a zero set which is the plane of points

$$\{x : x = u + -b\frac{w}{\|w\|^2}, u \text{ such that } w \cdot u = 0\}$$

In this representation:

- u such that $w \cdot u = 0$ is the hyperplane of vectors u orthogonal to w.
- the vector $-b\frac{w}{\|w\|^2}$ shifts the hyperplane antiparallel to direction of w.

That holds because

$$\begin{aligned} w \cdot x + b &= w \cdot (u + -b\frac{w}{\|w\|^2}) + b \\ &= w \cdot u + w \cdot (-b\frac{w}{\|w\|^2}) + b \\ &= 0 + -b\frac{w \cdot w}{\|w\|^2} + b \\ &= 0 + -b\frac{\|w\|^2}{\|w\|^2} + b = 0 \end{aligned}$$

1.5.2 What is the set of points x where the prediction is a constant, that is $g_{w,b}(x) = c$?

This can be answered by reducing it to a zero set:

$$\begin{aligned} g_{w,b}(x) = wx + b &= c \\ wx + (b - c) &= 0 \end{aligned}$$

So the set of points x such that $g_{w,b}(x) = c$ is just the zero-set of $g_{w,b-c}(x)$.

1.6 VI. a method to select a prediction mapping $g_{w,b}$ / or its parameters from a training dataset

First of all, we assume here no bias for this lecture (to get an explicit solution). Parameters are w.
We want to find parameters which minimize the loss on this dataset, that is:

$$(w^*) = \text{argmin}_w \sum_{i=1}^{n} L(f(x_i), y_i) = \text{argmin}_w \sum_{i=1}^{n} (x_i \cdot w - y_i)^2$$

for a given dataset $D_n = \{(x_i, y_i)\}$. Then $f_{w^*}(x) = x \cdot w^*$ is the selected mapping.

Rare case: Can be solved explicitly for w. Write in matrix form:

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1^{(1)}, \ldots, x_1^{(D)} \\ x_2^{(1)}, \ldots, x_2^{(D)} \\ \vdots \\ x_n^{(1)}, \ldots, x_n^{(D)} \end{pmatrix} \in \mathbb{R}^{n \times D}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

then:

$$\sum_{i=1}^{n} (x_i \cdot w - y_i)^2 = (X \cdot w - Y)^T \cdot (X \cdot w - Y)$$

Solve the minimization problem by computing the gradient for w and setting it to zero.

$$\begin{aligned} D_w((X \cdot w - Y)^T \cdot (X \cdot w - Y)) = 2X^T \cdot (X \cdot w - Y) &= 0 \\ \Rightarrow (X^T \cdot X) \cdot w &= X^T \cdot Y \\ w &= (X^T \cdot X)^{-1} X^T \cdot Y \end{aligned}$$

if the matrix inverse $(X^T \cdot X)^{-1}$ exists.

solution without bias

Let X be the training data matrix.

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1^{(1)}, \ldots, x_1^{(D)} \\ x_2^{(1)}, \ldots, x_2^{(D)} \\ \vdots \\ x_n^{(1)}, \ldots, x_n^{(D)} \end{pmatrix} \in \mathbb{R}^{n \times D}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^{n \times 1}$$

The model is $f(x) = w \cdot x$. Then a solution is given as

$$w = (X^T \cdot X)^{-1} X^T \cdot Y$$

if the matrix inverse $(X^T \cdot X)^{-1}$ exists.

How to extend this to a bias?

Extend each feature by an additional dimension set to 1:

$$x = \left(x^{(1)}, \ldots, x^{(D)}\right) \to \hat{x} = \left(x^{(1)}, \ldots, x^{(D)}, 1\right)$$

Then the parameter w also gets an additional dimension

$$\hat{w} = \left(w^{(1)}, \ldots, w^{(D)}, w^{(D+1)}\right)$$

Demo: `t1()` shows a solution. You have `t1()` also in your code for playing with it!

1.7 VII. From Linear to Ridge regression

Overfitting: low training error, high test error. Reason: during learning one picks up too much of the noise in the training data, and learns weights that listen to noise signals.

One way to deal with it: avoiding weights w getting too large: add a penalty on the euclidean length of w

$$\text{argmin}_w \sum_{i=1}^{n} (x_i \cdot w - y_i)^2 + \lambda \|w\|^2$$

Ridge regression

Ridge regression is Linear regression with added penalty term on weights

$$\lambda \|w\|^2$$

λ is a hyperparameter in this approach. The solution changes to

$$w = (X^T \cdot X + \lambda I)^{-1} X^T \cdot Y$$

In practice, one needs to find a good value for the hyperparameter λ on a validation set, before measuring the performance on the test set. The effect of this regularization will be discussed later.

2 Linear model for classification (hingeloss)

- I. Input and output space?
- II. the prediction mapping / model
- III. the loss function used to train the model
- IV. the goal in classification - what means generalization
- V. from classification with hingeloss to SVM
- IV. how to obtain parameters is deferred to after the lecture on gradient

2.1 I. Input and output space

Input and output space in classification problems?

2 classes: $\mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \{0, 1\}$ or $\mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \{-1, 1\}$

C classes: $\mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \{0, \ldots, C-1\}$

2.2 II. the prediction mapping?

Consider the case of two classes. Linear function without/with a bias. Thresholded by a sign

$$f_w(x) = x \cdot w \qquad = \sum_{d=1}^{D} x_d w_d \qquad , \; w \in \mathbb{R}^{D \times 1}$$

$$g_{w,b}(x) = x \cdot w + b \qquad = \sum_{d=1}^{D} x_d w_d + b \qquad , \; w \in \mathbb{R}^{D \times 1}, b \in \mathbb{R}^1$$

$$h(x) = sign g_{w,b}(x)$$

2.3 III. Loss function for classification

First loss function for a pair (x, y):

0-1-loss

$$L(f(x), y) = 1[\text{sign}(g_{w,b}(x)) \neq y]$$

Problem: sign is unsuitable for gradient optimization. better:

hinge-loss

$$L(f(x), y) = \max(0, 1 - g_{w,b}(x)y)$$

The hinge-loss is an upper bound on the zero-one-loss. Therefore, if the upper bound gets minimized, then the 0-1-loss would get minimized as well.

2.4 IV. the goal in classification

Find parameters w, b which generalize well to new, unseen data points. Generalize well means here again the same as for regression:

generalization for classification with the hinge-loss

1. we find parameters on a training set,

$$(w^*, b^*) = \text{argmin}_{(w,b)} \hat{L}(f, D_n) = \text{argmin}_{(w,b)} \frac{1}{n} \sum_{(x_i,y_i)\in D_n} L(f(x_i), y_i)$$
$$= \text{argmin}_{(w,b)} \frac{1}{n} \sum_{(x_i,y_i)\in D_n} \max(0, 1 - g_{w,b}(x_i)y_i)$$

2. with the goal that for most of these K test data sets (more formally: with high probability over the set of all draws of test datasets of size n), the averaged loss $aL(f, T_n^{(k)})$ on the test dataset is low.

$$\hat{L}(f, T_n^{(k)}) = \frac{1}{n} \sum_{(x_i,y_i)\in T_n^{(k)}} L(f(x_i), y_i)$$
$$= \frac{1}{n} \sum_{(x_i,y_i)\in T_n^{(k)}} \max(0, 1 - g_{w,b}(x_i)y_i)$$

Take away I: generalization

Generalization is the same idea for many different setups: learn parameters on training data, so that the loss on newly drawn test datasets will be low on average – for most draws of such datasets from a data source.

The difference between the regression part and the classification part is only the choice of loss function.

Take away II

See also that the linear model is suitable for classification or regression – the loss function makes what the linear model (and any model in general, be it tree-classifiers or deep neural nets!) will predict after training.

2.5 V. from classification with hingeloss to SVM

now add again a quadratic penalty on the weights when learning parameters over a training set

$$\text{argmin}_{(w,b)} \frac{1}{n} \sum_{(x_i,y_i)\in D_n} \max(0, 1 - g_{w,b}(x_i)y_i) + \lambda\|w\|^2$$

SVM:

- averaged hinge loss $\max(0, 1 - f(x_i)y_i)$

- with a linear/affine model $f(x_i) = w \cdot x_i + b$
- with quadratic penalty $\lambda \|w\|^2$ on the weights.

2.6 VI. a method to select a prediction mapping $g_{w,b}$ / or its parameters from a dataset

deferred to the gradient lecture !

3 Basis functions

Deep learning $\approx$ learning basis functions from data. Here we discuss the usage of fixed basis functions as a prelude.

Linear functions try to fit a hyperplane - can be too restricted for fitting many datasets. Alternative: Map data $x_1, \ldots, x_n$ into some feature space by some mapping ϕ, then do a linear regression on $\phi(x_1), \ldots, \phi(x_n)$
Example: polynomial basis functions

$$\phi(x) = \begin{pmatrix} 1 \\ x^1 \\ x^2 \\ x^3 \\ \ldots \\ x^F \end{pmatrix}$$

(for a vector x do this on each dimension). By that:

$$w \cdot x = w_0 + w_1 x^1 + w_2 x^2 + w_3 x^3 + \ldots$$

Now the output is a polynomial in x of degree F with learnable coefficients w.

What is the relevance of basis functions beyond direct usage in linear regression?

- Radial basis functions are able to represent very general functions
- A layer in a neural network can be interpreted as: being a set of (learned, not fixed) basis functions that get used in the next layer.

Demo: `t2()` shows a solution for varying values of F

Demo: `t3()` shows train and test error for varying values of F

Problem: for n data points a polynomial with $F = n - 1$ achieves zero training error, but test error is too high – overfitting

Example: radial basis functions
Given a set of points $P = (x_1, \ldots, x_S)$, the gaussian radial basis function mapping is defined for any sample x as:

$$\phi(x) = (rbf(x, x_1), rbf(x, x_2), \ldots, rbf(x, x_S)) \in \mathbb{R}^S$$
$$rbf(u, v) = \exp(-\frac{\|u - v\|^2}{\lambda^2})$$

Idea: $\|x - x_i\|^2$ is a distance to the representative $x_i \in P$.
$exp(-\|x - x_i\|^2)$ is a kind of similarity.
The feature vector is a vector of similarities with respect to the samples in the set P.

Note here: x is a vector with the same dimensionality as the x_i. $\|u - v\|^2$ computes a real number from 2 vectors. So $rbf(x, x_i) = \exp(-\frac{\|x - x_i\|^2}{\lambda^2})$ is a real number. And $\phi(x)$ is a vector with S dimensions - which is the number of samples in the S.

Note also: this feature mapping has two parameters: the set of points P and the kernel width λ

How to obtain P ? Examples:

- sample at randomly S elements from the input features of the training data set $\{x_1, \ldots, x_n\}$
- perform k-means clustering on $(x_1, x_2, \ldots, x_n)$ with S centroids, P will be the centroids
- $P = D_n$ (use the whole training data, not always a good idea, locks feature dimensionality to sample size)

$$w \cdot \phi(x) = \sum_r w_r \exp(-\frac{\|x - x_r\|^2}{\gamma})$$

is a weighted sum of similarities between x and the set of points P

Implement it, and see its impact for different choices of kernel width γ in `learn.py`
Call `tRBF([val1,val2,val3])` with different values of the kernel width. For a too small kernel width one will have train error zero.