# Introduction to Machine Learning

## Lecture 4 - Kernel Methods & Support Vector Machine (SVM)

GuangBing Yang, PhD

yguangbing@gmail.com, Guang.B@chula.ac.th February 12th, 2021 



### What is kernel method and Why

- Why do we need the kernel method?
  - the training data {x, t} are discarded after training a model, which learns parameter vector w or posterior distribution p(w|t)
  - this kind of approach is called **Parametric** methods (linear/nonlinear, classification/regression, etc.)
- Another approach, namely **Non-parametric** methods, do use the training data or partial training data during the prediction. For example,
  - Parzen probability density model: set of kernel functions centred on training data points.
  - Nearest neighbours techniques: use closest examples from the training set
  - Memory-based methods: similar examples from the training data.
- Kernel method: it is a kind of technique that uses some of data points as examples in predicting the new data points. These examples consist of a kernel function, which are evaluated during training.



### Kernel Function and Dual Representation

- The kernel function brings up the concept of dual representation,
  - which is in linear parametric models about the predictions are also based on linear combinations of a kernel function evaluated at the training data.
  - For example, a model based on a fixed nonlinear feature space mapping $\Phi(x)$, its kernel function is given as $k(x, x') = \phi(x)^T\phi(x')$.
- Kernel function examples:
  - for models based on feature space mapping $\Phi(x)$: $k(x, x') = \phi(x)^T\phi(x')$
  - symmetric function: $k(x, x') = k(x', x)$
  - linear kernel: $k(x, x') = (x)^T(x')$
  - stationary kernel: $k(x, x') = k(x - x')$
  - homogeneous kernel: $k(x, x') = k(||x - x'||)$



### Kernel Function and Dual Representation

- More Kernel function examples:
- Polynomial kernels:, $k(x, x') = (x^T x')^2$, $k(x, x') = (x^T x' + c)^2, c > 2$, $k(x, x) = (x^T x1)^M$, $k(x, x') = (x^T x' + c)^M, c > 0$
- Gaussian kernel: $k(x, \tilde{x}) = \exp(-||x - x'||^2/2\sigma^2)$
- Sigmoidal kernel: $k(x, \tilde{x}) = \tanh(ax^T x' + b)$

## Kernel Function and Dual Representation

- Dual representation for linear regression and classification
  - Given an linear regression model ($\lambda \geq 0$):
    $J(w) = \frac{1}{2}\sum_{n=1}^{N}\{w^T\phi(x_n) - t_n\}^2 + \frac{\lambda}{2}w^T w$
  - let gradient to zero for w calculation:
    $w = -\frac{1}{\lambda}\sum_{n=1}^{N}\{w^T\phi(x_n) - t_n\}\phi(x_n) = \sum_{n=1}^{N} a_n\phi(x_n) = \Phi^T \mathrm{a},$
  - where $\Phi$ is the design matrix, and $\mathrm{a} = (a_1, \ldots, a_N)^T$ can be represented as:
    $a_n = -\frac{1}{\lambda}\{w^T\phi(x_n) - t_n\}$




## Kernel Function and Dual Representation

- Dual representation for linear regression and classification
  - Instead of working with the parameter vector w, using dual representation for a:
    $J(a) = \frac{1}{2}a^T\Phi\Phi^T\Phi\Phi^T a - a^T\Phi\Phi^T\mathrm{t} + \frac{1}{2}\mathrm{t}^T\mathrm{t} + \frac{\lambda}{2}a^T\Phi\Phi^T a$, where
    $\mathrm{t} = (t_1, \ldots, t_N)^T$.
  - Define the *Gram* matrix $K = \Phi\Phi^T$, a $N \times N$ symmetric matrix with elements
    $K_{nm} = \phi(x_n)^T\phi(x_m) = k(x_n, x_m)$, this is the kernel function defined previously.
  - in terms of the Gram matrix, the
    $J(a) = \frac{1}{2}a^T\Phi\Phi^T\Phi\Phi^T a - a^T\Phi\Phi^T\mathrm{t} + \frac{1}{2}\mathrm{t}^T\mathrm{t} + \frac{\lambda}{2}a^T\Phi\Phi^T a,$
  - the error function can be: $J(a) = \frac{1}{2}a^T K K a - a^T K\mathrm{t} + \frac{1}{2}\mathrm{t}^T\mathrm{t} + \frac{\lambda}{2}a^T K a$




## Kernel Function and Dual Representation

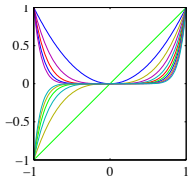
- Solution for Dual representation for linear regression and classification
  - with w, we can get $a = (K + \lambda I_N)^{-1}\mathrm{t}$
  - Substitute back this to the linear regression model, the prediction for a new input x is given: $y(\mathrm{x}) = w^T\phi(x) = a^T\Phi\phi(x) = k(x)^T(K + \lambda I_N)^{-1}\mathrm{t}$, where
    $k(x) = (k(x_1, x), \ldots, k(x_N, x))$
  - Where in the original parameter space, we have to invert an $M \times M$ matrix, instead in kernel function to insert a $N \times N$ matrix.
  - Since N is larger than M, it seems the dual representation not to be particularly useful.
  - However, the advantage of the dual representation is that it is expressed entirely in terms of the kernel function. One can work directly in kernels and avoid the explicit feature spaces $\phi(x)$. This allows to use feature spaces of high, even infinite, dimensionality.

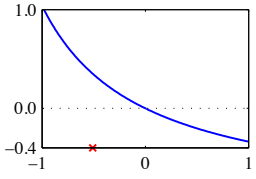



## Construct Kernels

- Solution for construct kernels
  - approach one: choose feature space mapping $\phi(x)$:
    - $k(x, x') = \phi(x)^T\phi(x') = \sum_{i=1}^{M}\phi_i(x)\phi_i(x')$
  -




feature space mapping

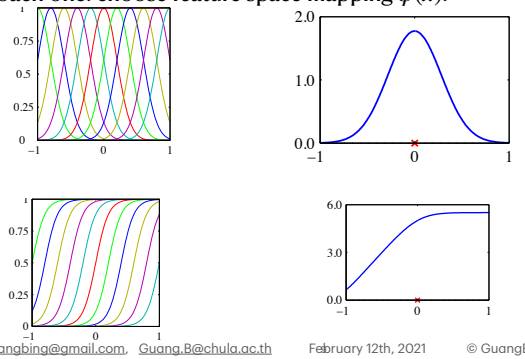



kernels

## Construct Kernels

- Solution for construct kernels
  - approach one: choose feature space mapping $\phi(x)$:
  -



9

## Construct Kernels

- Solution for construct kernels
  - approach two: construct kernel function directly and verify its validity:
  - for example, construct a kernel function: $k(x,z) = (x^T z)^2$
  - in 2-D case corresponds to: $k(x,z) = \phi(x)^T\phi(z)$, with $\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$
  - To verify its validity using the condition: $k(x,z) = \phi(x)^T\phi(z)$ is a valid kernel as long as $K \geq 0 \quad \forall\{x_n\}$


10

## Techniques for Constructing New Kernels

- Solution for construct new kernels given valid kernels $k_1(x,x')$ and $k_2(x,x')$, the following new kernels are also valid:
  - $k_1(x,x') = ck_1(x,x')$, c is a constant
  - $k(x,x') = f(x)k_1(x,x')f(x')$
  - $k(x,x') = q(k_1(x,x'))$
  - $k(x,x') = \exp(k_1(x,x'))$
  - $k(x,x') = k_1(x,x') + k_2(x,x')$
  - $k(x,x') = k_1(x,x')k_2(x,x')$
  - $k(x,x') = k_3(\phi(x),\phi(x'))$


11

## Techniques for Constructing New Kernels

- For example:
  - Kernel for generative model p(x):
    - $k_1(x,x') = p(x)p(x')$,
    - $k_1(x,x') = \sum_i p(x\,|\,i)p(x'\,|\,i)p(i)$, or $k_1(x,x') = \int p(x\,|\,z)p(x'\,|\,z)p(z)dz$
  - Kernel for Hidden Markov Model (HMM): $k(X,X') = \sum_Z p(X\,|\,Z)p(X'\,|\,Z)p(Z)$, where X are observations, and Z are hidden states.


12

## Kernels Connect to Neural Networks

- In Neural Networks, for sufficiently large dimensions of the hidden layers, a two-layer network can approximate any given function with arbitrary accuracy.
- However, in the framework of maximum likelihood, the number of hidden units needs to be limited to the number of training data points in order to avoid the over-fitting problem.
- In a Bayesian Neural Network,
  - the prior distribution over the parameter w in conjunction with the network function f(x, w) produces a prior distribution over functions y(x)
  - the distribution of functions will tend to a gaussian process in the limit M -> ∞
  - the outputs of neural works share hidden units and so they 'borrow statistical strength' from each other, that is the weights associated with each hidden unit are influenced by all of the output variables not just one of them. This property makes Gaussian process lost its limit.
  - Gaussian process is determined by its covariance (kernel) function.




## Gaussian Processes

- In Neural Networks,
  - A kernel function can represent the covariance of Gaussian process.
  - So, that is why kernel methods and corresponding algorithms and models are important to study.
- Kernel to probabilistic discriminative models leads to the framework of Gaussian processes.
- "Gaussian process is defined as a probability distribution over functions y(x) such that the set of values of y(x) evaluated at an arbitrary set of points $x_1, \ldots, x_N$, jointly have a Gaussian distribution." — Simply say a Gaussian process is a mixture (joint) of a set of Gaussian distributions.




## Gaussian Processes

- The key point in Gaussian processes is: the joint distribution is defined completely by second-order statistics (which are mean and covariance).
- Note that usually the mean is take to zero, then we only need the covariance, which is a kernel function:
  - $\mathbb{E}[y(x_n), y(x_m)] = k(x_n, x_m)$




## Gaussian Process for Regression

- To use Gaussian processes for regression, we need to model noise:
  - $t_n = y_n + \epsilon_n$, with $y_n = y(x_n)$, $y_n$ is the regression model.
- For noise processes with a Gaussian distribution we obtain:
  - $p(t_n \mid y_n) = \mathbb{N}(t_n \mid y_n, \beta^{-1})$
- Because the noise is independent for each data point, the joint distribution for $\mathrm{t} = (t_1, \ldots, t_N)^T$ and $\mathrm{y} = (y_1, \ldots, y_N)^T$, is given by $p(\mathrm{t} \mid \mathrm{y}) = \mathbb{N}(\mathrm{t} \mid \mathrm{y}, \beta^{-1}\mathrm{I}_N)$, where $\mathrm{I}_N \in N \times N$, unit matrix.
- From the definition of the Gaussian process, the marginal distribution p(**y**) is given by a Gaussian whose mean is zero and covariance is a kernel function defined by a Gram matrix **K**,

## Gaussian Process for Regression

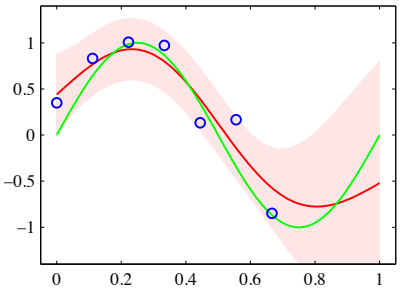
- So that, $p(\mathrm{y}) = \mathbb{N}(\mathrm{y}\,|\,0,K)$
- The whole point is that the kernel function that determines K is typically chosen to express the property that for points $x_n$ and $x_m$ that are similar, the corresponding values $y(x_n)$ and $y(x_m)$ will be more strongly correlated than for dissimilar points.
- For the marginal distribution p(**t**), integrate over **y**, we have:
  - $p(\mathrm{t}) = \int p(\mathrm{t}\,|\,\mathrm{y})p(\mathrm{y})d\mathrm{y} = \mathbb{N}(\mathrm{t}\,|\,0,\mathrm{C})$, with $\mathrm{C} = K + \beta^{-1}\mathrm{I}$
- To predict by partitioning the joint Gaussian distributions over $x_1, \ldots, x_N, x_{N+1}$, obtain $p(t_{N+1}\,|\,\mathrm{t})$ given by its mean and covariance:
  - $m(x_{N+1}) = K^T C^{-1}\mathrm{t}, \;\; K = (k(x_1, X_{N+1}), \ldots, k(x_N, x_{N+1}))^T$
  - $\sigma^2(x_{N+1}) = c - K^T C^{-1} K, \;\; c = k(x_{N+1}, x_{N+1}) + \beta^{-1}$




## Gaussian Process for Regression

- For example, a prediction
- Green curve: original sinusoidal function; blue points: sampled training data points with additional noise.
- read line: mean estimate;
- shaded regions: +/- 2σ

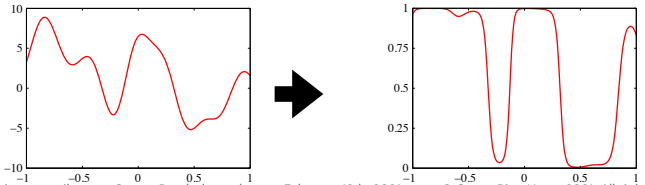





## Gaussian Process for Classification

- Again, the classification needs a decision making to map regression values to class values. For a binary classification, use sigmoid
- The goal is to model the posterior probabilities of the target variables for a new input x.
- Need to map values to interval (0; 1)
- Use a Gaussian process together with a non-linear activation function for decision surface (or boundary).






## Gaussian Process for Classification

- Again, need defining a Gaussian process over a function $a(x)$ and transform a using the logistic sigmoid to $y = \sigma(a(x))$ for a binary classification.
- Similar to the regression, to predict the target variable given new input x
- $p(t_{N+1} = 1\,|\,\mathrm{t}) = \int p(t_{N+1} = 1\,|\,a_{N+1})p(a_{N+1}\,|\,\mathrm{t})da_{N+1} = \int \sigma(a_{N+1})p(a_{N+1}\,|\,\mathrm{t})da_{N+1}$
- This integral is intractable due to the online sigmoid function. Approximations need to apply for numerical or analytical results.
- Use the Laplace approximation to approximate the posterior over $a_{N+1}$.
- With Bayes' theorem, we have:
  $p(a_{N+1}\,|\,\mathrm{t}) = \int p(a_{N+1}, \mathrm{a}\,|\,\mathrm{t})d\mathrm{a} = \ldots = \int p(a_{N+1}\,|\,\mathrm{a})p(\mathrm{a}\,|\,\mathrm{t})d\mathrm{a}$

## Support Vector Machine (SVM)

- In kernel methods, one of the significant limitations is that the kernel function must be evaluated for all possible pairs $x_n$ and $x_m$ of training points. This process is very computational expensive.
- SVM comes out a solution as a kernel-based algorithm that has sparse solutions—which means the kernel function evaluated at a subset of the training data points.
- An important property of SVM is that the determination of the model parameters corresponds to a convex optimization problem—which means there is a global optimum.
- SVM uses Lagrange multipliers as optimization constraints to optimize the parameters to find the global optimums.
- One of limitations of SVM is that it is a decision machine and so does not provide posterior probabilities.







## Maximum Margin Classifiers (SVM)

- For a binary classification, the linear model is given as: $y(x) = w^T\phi(x) + w_0$
- The classifier separates the data points based on $y(x_n) > 0, \forall x$, having $t_n = +1$ and $y(x_n) < 0, \forall x$, having $t_n = -1$, so $t_n y(x_n) > 0$ for all data points.
- The problem of this approach is that there are multiple ways to separate the data. We need to find the one that gives the smallest generalization error.
- SVM approaches this problem via a concept of the margin—defines the smallest distance between the decision boundary and any of the samples.
- To keep the generalization, SVM choses the decision boundary to be the one for which the margin is maximized.

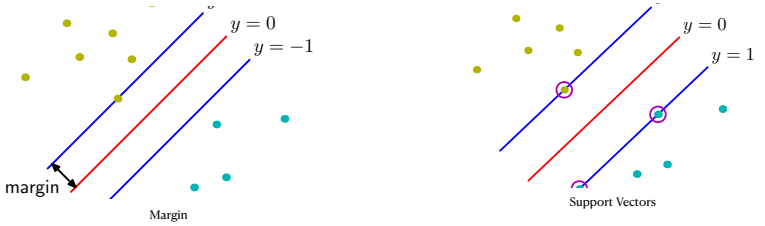






## Maximum Margin Classifiers (SVM)

- The diagram in the left side shows the margin is defined as the distance between decision boundary and the closest of the data points.
- The right side shows that maximizing the margin leads to a particular choice of decision boundary. The location of this boundary is determined by a subset of the data points, which known as ***support vectors***, indicated by the circles.









## Maximum Margin Classifiers (SVM)

- So, the intuition of the SVM approach is to find this subset of the data points, which known as ***support vectors***.
- Remember the Gaussian kernels having a covariance $\Sigma$ or variance $\sigma^2$. In the limit $\sigma^2$ ->o, the optimal hyperplane is shown to be one having maximum margin.
- The intuition behind this result is that as $\sigma^2$ is reduced, the hyperplane is increased dominated by nearby data points relative to more distance ones. In the limit, the hyperplane becomes independent of data points that are not support vectors.
- Based on this, we find such small set of data points that can independently determine the decision boundary, such a subset of the data points are called ***support vectors***
- Other data points that do not belong to ***support vectors*** will not participate in the prediction process.

## Maximum Margin Classifiers (SVM)

- The Lagrange multipliers $a_n \geq 0$ is used to find the optimum of parameters.
- $L(\mathrm{w}, b, \mathrm{a}) = \frac{1}{2}||\mathrm{w}||^2 - \sum_{n=1}^{N} a_n\{t_n(\mathrm{w}^T\Phi(\mathrm{x}_n) + b) - 1\}$,
- minimize L w.r.t w and b equal to zero, obtain following two conditions:
  - $\mathrm{w} = \sum_{n=1}^{N} a_n t_n \Phi(\mathrm{x}_n)$
  - $0 = \sum_{n=1}^{N} a_n t_n$
- Substitute w and b from L(w, b, a) using these conditions then gives the dual representation of the maximum margin problem in which we maximize,
- $\hat{L}(\mathrm{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathrm{x}_n, \mathrm{x}_m)$, w.r.t **a** subject to the constraints




## Maximum Margin Classifiers (SVM)

- $\hat{L}(\mathrm{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathrm{x}_n, \mathrm{x}_m)$, w.r.t **a** subject to the constraints
- $a_n \geq 0$, $n = 1,...,N$, and $\sum_{n=1}^{N} a_n t_n = 0$. \$. Here the kernel function is defined by $k(\mathrm{x}, \mathrm{x}') = \Phi(\mathrm{x})^T\Phi(\mathrm{x}')$.
- For binary classification $y(\mathrm{x}) = \mathrm{w}^T\Phi(\mathrm{x}) + b$, the $y(\mathrm{x}) = \sum_{n=1}^{N} a_n t_n k(\mathrm{x}, \mathrm{x}_n) + b$, and the Karush-Kuhn-Tucker (KKT) condition gives the constrained optimization of this form:
  - $a_n \geq 0$
  - $t_n y(\mathrm{x}_n) - 1 \geq 0$
  - $a_n\{t_n y(\mathrm{x}_n) - 1\} = 0$




## Maximum Margin Classifiers (SVM)

- Thus, for each data point, either $a_n = 0$ or $t_n y(\mathrm{x}_n) = 1$.
- Any data point for which $a_n = 0$ will not appear in the sum $y(\mathrm{x}) = \sum_{n=1}^{N} a_n t_n k(\mathrm{x}, \mathrm{x}_n) + b$ and hence plays no role in making predictions for new data points.
- The remaining data points are ***support vectors***, and they satisfy $t_n y(\mathrm{x}_n) = 1$, they correspond to points that lie on the maximum margin hyperplanes in feature space.
- This is the central property of the SVMs in practice. Once the model is trained, a significant proportion of the data points can be discarded and only the support vectors retained.




## Recap

- Kernel method is a kind of technique that uses some of data points as examples in predicting the new data points.
- The kernel function brings up the concept of dual representation, which is evaluated at the training data.
- The dual representation is based on linear combinations of a kernel function.
- The advantage of the dual representation is that it is expressed entirely in terms of the kernel function. This allows to use feature spaces of high, even infinite, dimensionality.
- To construct kernels, there are two approaches: one is to choose feature space mapping $\phi(x)$, another one is construct kernel function directly and verify its validity.
- There are many techniques to construct new kernels from valid kernels.

## Recap

- Kernel methods connect to the Neural Networks.
- In a Bayesian Neural Network, the prior distribution over the parameter w in conjunction with the network function f(x, w) produces a prior distribution over functions y(x).
- The distribution of functions will tend to a gaussian process in the limit M -> ∞,
- which is determined by its covariance (kernel) function.
- So, that is why kernel methods and corresponding algorithms and models are important to study.
- Kernel to probabilistic discriminative models leads to the framework of Gaussian processes.
- Gaussian process is a mixture (joint) of a set of Gaussian distributions.




## Recap

- In kernel methods are very computational expensive.
- SVM comes out a solution as a kernel-based algorithm that has sparse solutions—which means the kernel function evaluated at a subset of the training data points.
- An important property of SVM is that the determination of the model parameters corresponds to a convex optimization problem—which means there is a global optimum. Once SVM finds the optimal parameters, it may perform better than other classification algorithms.
- SVM uses Lagrange multipliers as optimization constraints to optimize the parameters to find the global optimums.
- One of limitations of SVM is that it is a decision machine and so does not provide posterior probabilities. This limitation is solved by Relevance Vector Machine (RVM).
- SVM maximizes the margin leads to a particular choice of decision boundary. The location of this boundary is determined by a subset of the data points, which known as ***support vectors***. That is SVM name comes from.




## Questions?

## Lab4 is about the measurement of model performance