CS:E4830 Kernel Methods in Machine Learning

Lecture 3 : Representer Theorem and Polynomial Kernel Example

17th March, 2021

Some Announcements

- Lecture slides of this (3rd) lecture - uploaded to Mycourses
- Assignment 1 has been released, deadline to finish by 30th March
- Next week Thursday (25th March) - tutorial session (Zoom) for assignment 1
- No tutorial session this week on Thursday

Outline For Today

1. Recap and Motivation
2. Representer Theorem
   - Proof
   - Kernel Least Square Regression
3. Explicit Feature Maps with Kernels
   - Polynomial kernel with example
   - LibShorttext

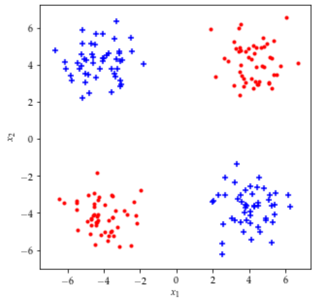
Recap and Motivation

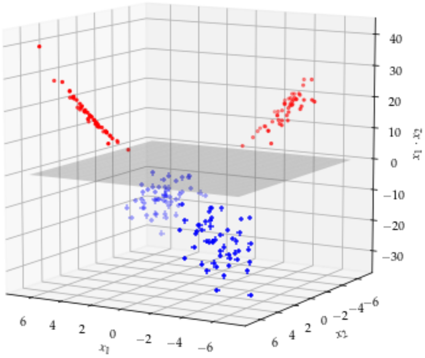
So far...

- Lecture 1
  - Linear and Non-linear classification
  - Kernel functions $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$
  - Properties of kernel functions
- Lecture 2
  - Positive definite functions
  - Equivalence between kernel and positive definite functions
  - Moore-Aronszjan theorem and Reproducing Kernel Hilbert Space
- This lecture - Representer Theorem

Non-linear Classification Example

- Dataset in 2-D (left), which is not linearly separable
- can be separated by a plane in 3-D (third feature is the product $x_1 x_2$)





Non-linear classification

Prediction function can involve non-linear combination of features

- For the classification function f_2 below, which is linear in weights and **non-linear in input features**

$$f_2(x) = w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + w^{(3)}x^{(1)}x^{(2)} + w^{(4)}x^{(2)}x^{(1)}$$

- Here, the decision function $f_2(x)$ is trying to capture **non-linear combination** of the input components as well such as $x^{(1)}x^{(2)}, x^{(2)}x^{(1)}$
- Non-linear feature map $\phi_2 : \mathbb{R}^2 \mapsto \mathbb{R}^4$, and is given by $\phi_2(x) = (x^{(1)}, x^{(2)}, x^{(1)}x^{(2)}, x^{(2)}x^{(1)})^T$
  - $\phi_2(x) \in \mathcal{H}$, which is referred to as the feature space
- Note that the decision function $f_2(x)$ **is still linear in the weight vector co-efficients** $w^{(j)}$'s and is parameterized by $(w^{(1)}, w^{(2)}, w^{(3)}, w^{(4)})^T$

Implcit vs Explicit Computation

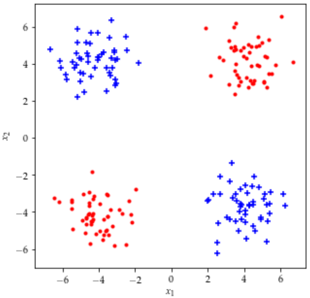
- Example - Polynomial kernel of degree 2. Assuming inputs $x, z \in \mathbb{R}^d$, i.e. $x = (x^{(1)}, x^{(2)}, \ldots, x^{(d)})$

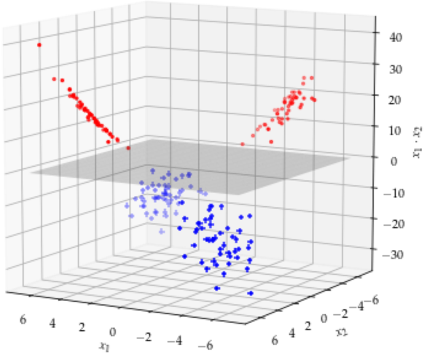
$$k_{poly}(x, z) = (\langle x, z \rangle)^2 = \langle \phi(x), \phi(z) \rangle$$

- What is the dimensionality of the input space and feature space, $\phi(.)$ (keeping order into account. i.e., $x^{(1)}x^{(2)}$ is different from $x^{(2)}x^{(1)}$)?
  - $\phi : \mathbb{R}^d \mapsto \mathbb{R}^{d^2}$
  - $\phi(x) = \{x^{(1)}x^{(1)}, x^{(1)}x^{(2)}, \ldots x^{(1)}x^{(d)}, \ldots, x^{(d)}x^{(1)}, x^{(d)}x^{(2)}, \ldots, x^{(d)}x^{(d)}\}$
- Computational complexity of
  - $\langle \phi(x), \phi(z) \rangle$? $O(d^2)$
  - $k_{poly}(x, z)$? $O(d)$
- What if we consider another kernel with a higher degree such as $k_{poly}(x, z) = (\langle x, z \rangle)^{10}$

Decision Boundary in Non-linear Case

- How to represent the decision boundary in non-linear classification
- In Gaussian kernel case - feature space is infinite dimensional





Linear classification $\phi(x) = x$

Dual representation of the optimal hyperplane

- Consequently, the functional margin $y\mathbf{w}^T\mathbf{x}$ also can be expressed using the support vectors:

$$y\mathbf{w}^T\mathbf{x} = y\sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i^T\mathbf{x}$$

- The norm of the weight vector can be expressed as

$$\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i^T \sum_{j=1}^{m}\alpha_j y_j \mathbf{x}_j = \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

- Note that the training data appears in pairwise inner products: $\mathbf{x}_i^T\mathbf{x}_j$

24

Representation Learning

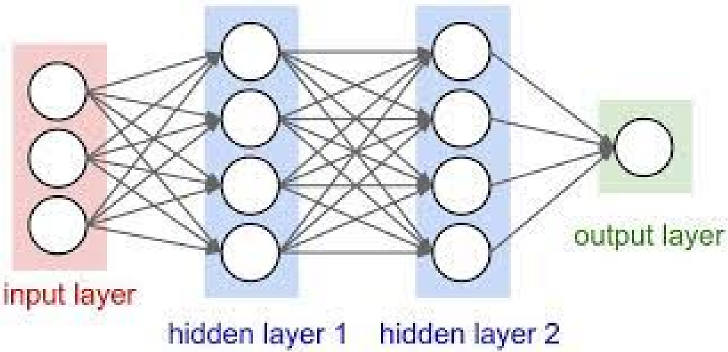


Figure: Representation learning in deep networks

In kernel Methods

- Feature mapping $\phi(x)$ is also a representation but that is not really learnt
- Given a kernel it is a fixed representation

Representer Theorem

Empirical Loss

- For a loss function L, and dataset D of finite size N, Empirical loss $\hat{\mathcal{R}}_{L,D}$ of a classifier f is defined as

$$\hat{\mathcal{R}}_{L,D} := \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i))$$

- The classifier obtained by minimizing the empirical loss is given by

$$\hat{f}_D := \arg \min_{f:\mathcal{X} \mapsto \mathbb{R}} \hat{\mathcal{R}}_{L,D}$$

- Minimizing Empirical risk over an arbitrary function class can lead to overfitting (as seen earlier in the lecture)
- Empirical risk minimization (ERM) is, therefore, performed over a smaller function class of function, which are typically smooth functions. This is given by

$$\hat{f}_{\mathcal{H}} := \arg \min_{f \in \mathcal{H}} \hat{\mathcal{R}}_{L,D}$$

- Pick a function class with bounded RKHS norm i.e. $\{f : ||f||_{\mathcal{H}} \leq \lambda\}$

Constrained versus Penalized Problem

- Contrained formulation (from previous slide)

$$\hat{f}_{\mathcal{H}} := \arg \min_{\{f:||f||_{\mathcal{H}} \leq \lambda\}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i))$$

- Lagrangian formulation

$$\hat{f}_{\mathcal{H}} := \arg \min_{f} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda ||f||_{\mathcal{H}}^2$$

  for $\lambda > 0$

- The above is optimization problem over potentially an infinite dimensional space, since $\mathcal{H}$ can be an infinite dimensional as in the case of Gaussian kernel.

Representer Theorem

- For the following objective function

$$\hat{f}_{\mathcal{H}} := \arg\min_{f} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda\theta(||f||_{\mathcal{H}}) \tag{1}$$

  where $\theta : [0, \infty) \mapsto \mathbb{R}$ is non-decreasing function

- Even though the above problem is potentially an infinite dimensional optimization problem, **Representer Theorem** states its solution can be expressed in the following form

$$\hat{f}_{\mathcal{H}}(.) = \sum_{i=1}^{N} \alpha_i k(., x_i)$$

  where $\alpha_i \in \mathbb{R}$

- Infinite to finite dimensional problem - instead of finding potentially infinite coefficients of $\hat{f}$, we need to find N coefficients of the above linear combination α_i

Representer Theorem - Proof

- Decompose the RKHS $\mathcal{H}$ into the following :

$$\mathcal{H} = \mathcal{H}_0 \oplus \mathcal{H}^{\perp}$$

  - where $\mathcal{H}_0 = \{f \in \mathcal{H} : f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i), (\alpha_i)_{i=1}^{N}\}$. It can also been seen as a *finite dimensional sub-space* spanned by the kernel evaluations at the points x_i, i.e. $\mathcal{H}_0 = \text{span}\{k(., x_1), k(., x_2), \ldots, k(., x_N)\}$
  - $\mathcal{H}^{\perp}$ is the component of $\mathcal{H}$, which is orthogonal to $\mathcal{H}_0$.
- Therefore, the function $f \in \mathcal{H}$ is decomposed as

$$f = f_0 + f^{\perp}$$

- For the loss term in the objective function (in previous slide)

$$f(x_i) = \langle f, k(., x_i) \rangle_{\mathcal{H}} = \langle f_0, k(., x_i) \rangle_{\mathcal{H}_0} + \langle f^{\perp}, k(., x_i) \rangle_{\mathcal{H}^{\perp}}$$

Representer Theorem - Proof

- Decompose the RKHS $\mathcal{H}$ into the following :

$$\mathcal{H} = \mathcal{H}_0 \oplus \mathcal{H}^\perp$$

  - where $\mathcal{H}_0 = \{f \in \mathcal{H} : f(x) = \sum_{i=1}^N \alpha_i k(x, x_i), (\alpha_i)_{i=1}^N\}$. It can also been seen as a *finite dimensional sub-space* spanned by the kernel evaluations at the points x_i, i.e. $\mathcal{H}_0 = \text{span}\{k(., x_1), k(., x_2), \dots, k(., x_N)\}$
  - $\mathcal{H}^\perp$ is the component of $\mathcal{H}$, which is orthogonal to $\mathcal{H}_0$.
- Therefore, the function $f \in \mathcal{H}$ is decomposed as

$$f = f_0 + f^\perp$$

- For the loss term in the objective function (in previous slide)

$$f(x_i) = \langle f, k(., x_i)\rangle_{\mathcal{H}} = \langle f_0, k(., x_i)\rangle_{\mathcal{H}_0} + \langle f^\perp, k(., x_i)\rangle_{\mathcal{H}^\perp}$$

The second term in the loss term $L(.,.)$ expansion (on the previous slide) is 0.

Representer Theorem - Proof

- For the regularization term use the Pythagoras theorem, i.e.

$$||f||^2_{\mathcal{H}} = ||f^{\perp}||^2_{\mathcal{H}^{\perp}} + ||f_0||^2_{\mathcal{H}_0}$$

  This implies that $||f_0||_{\mathcal{H}_0} \leq ||f||_{\mathcal{H}}$. Since $\theta(.)$ is a non-decreasing function $\theta(||f_0||_{\mathcal{H}_0}) \leq \theta(||f||_{\mathcal{H}})$
- Therefore, the optimal solution has no component in $\mathcal{H}^{\perp}$, and has the form

$$f_{\mathcal{H}}(.) = \sum_{i=1}^{N} \alpha_i k(., x_i)$$

Practical Implications of Representer Theorem

- It allows us to look for the solutions of the following form :

$$f_{\mathcal{H}}(.) = \sum_{i=1}^{N} \alpha_i k(., x_i)$$

  which are easier finite dimensional optimization problems as against the equivalent infinite dimensional original problems.

- For $j = 1 \dots N$

$$f_{\mathcal{H}}(x_j) = \sum_{i=1}^{N} \alpha_i k(x_i, x_j) = [K\boldsymbol{\alpha}]_j$$

  which is the j-th element of the matrix-vector product $K\boldsymbol{\alpha}$

- Also,

$$||f||^2_{\mathcal{H}} = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j k(x_i, x_j) = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

Least Square Regression

- Let f be the prediction function, then squared error is given by

$$\ell(f(x), y) = (y - f(x))^2$$

- Let $\mathcal{F}$ be a function class (not necessarily an RKHS) from which we are choosing our function
- Least Square regresion find a function with smallest sqared error

$$\hat{f} \in \arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i))^2$$

- Possible problems :
  - Can be unstable in high dimensions
  - Overfit if the function space $\mathcal{F}$ is too large

Kernel Ridge Regression

- Finding f in an RKHS with a kernel $k(x, x')$
- $$\hat{f} \in \arg\min_{f \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda ||f||_{\mathcal{H}}$$
- The above formulation has two advantages :
  - Prevents overfitting
  - Representer theorem enables an efficient solution of the form

$$\hat{f}(.) = \sum_{i=1}^{N} \alpha_i k(., x_i)$$

Solving Kernel Ridge Regression

- Lets denote by
  - The label vector $\mathbf{y} \in \mathbb{R}^N$ denoting the true values for the inputs
  - The kernel matrix K, where $K_{i,j} = K(x_i, x_j)$
  - $\boldsymbol{\alpha} \in \mathbb{R}^N$, the co-efficients we want to find
- For the input instance, the prediction by the desired function can be written as follows :

$$(\hat{f}(x_1), \ldots, \hat{f}(x_N))^T = K\boldsymbol{\alpha}$$

- We also know that

$$||f||_{\mathcal{H}}^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j k(x_i, x_j) = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

- Solving Kernel Ridge Regression involves solving

$$\arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \frac{1}{N} (K\boldsymbol{\alpha} - \mathbf{y})^T (K\boldsymbol{\alpha} - \mathbf{y}) + \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

Kernel Ridge Regression - Solution

- Desired optimization problem

$$\arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \frac{1}{N}(K\boldsymbol{\alpha}-\mathbf{y})^T(K\boldsymbol{\alpha}-\mathbf{y})+\lambda\boldsymbol{\alpha}^T K\boldsymbol{\alpha}$$

- The above is convex and differntiable w.r.t to $\boldsymbol{\alpha}$, and can be analytically found by setting the gradient

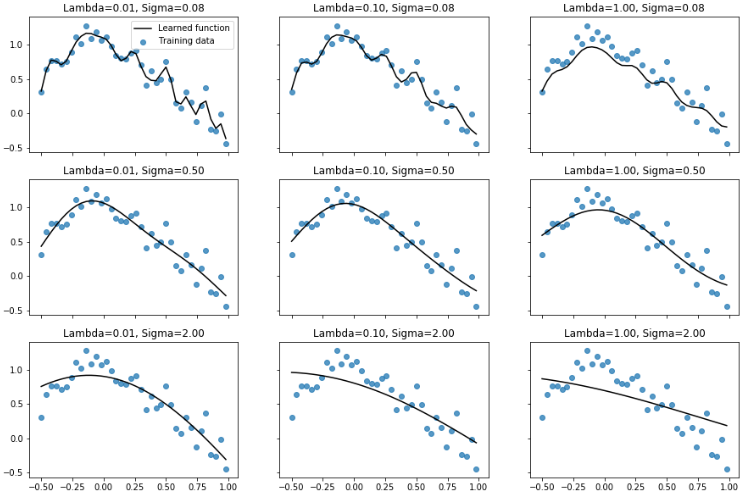
$$\frac{2}{N}K(K\alpha-\mathbf{y})+2\lambda K\alpha=\mathbf{0}$$

- Since K is positive definite (from previous lecture), we can invert $K+\lambda NI$, and hence the solution is given by

$$\boldsymbol{\alpha}=(K+\lambda NI)^{-1}\mathbf{y}$$

where I is the identity matrix

Kernel Ridge Regression with Gaussian Kernel



Recall - Linear classification

Assuming the input data x is 2-dimensional (i.e. in $\mathbb{R}^2$)

Linear classification

- Consider the classification function f_1 below, which is linear in both the input features and weights

$$f_1(x) = w^{(1)}x^{(1)} + w^{(2)}x^{(2)}$$

- In this case, the decision function $f_1(x)$ is trying to capture only **linear combination** of the input components x_1, x_2
- Linear feature map $\phi : \mathbb{R}^2 \mapsto \mathbb{R}^2$, and is given by, $\phi_1(x) = (x_1, x_2)^T$
- $f_1(.)$ is parameterized by three co-efficients $(w_1, w_2)^T$

Recall - Non-linear classification

Prediction function can involve non-linear combination of features

- For the classification function f_2 below, which is linear in weights and **non-linear in input features**

$$f_2(x) = w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + w^{(3)}x^{(1)}x^{(2)} + w^{(4)}x^{(2)}x^{(1)}$$

- Here, the decision function $f_2(x)$ is trying to capture **non-linear combination** of the input components as well such as $x^{(1)}x^{(2)}, x^{(2)}x^{(1)}$
- Non-linear feature map $\phi_2 : \mathbb{R}^2 \mapsto \mathbb{R}^4$, and is given by $\phi_2(x) = (x^{(1)}, x^{(2)}, x^{(1)}x^{(2)}, x^{(2)}x^{(1)})^T$
  - $\phi_2(x) \in \mathcal{H}$, which is referred to as the feature space
- Note that the decision function $f_2(x)$ **is still linear in the weight vector co-efficients** $w^{(j)}$'s and is parameterized by $(w^{(1)}, w^{(2)}, w^{(3)}, w^{(4)})^T$

Kernel Matrix Bottleneck

In many problem scenarios in modern day big data setup, it is not difficult to get millions (or even bigger) of training samples :

- Computing a million $\times$ million kernel matrix is not trivial
  - For instance, one needs to invert the kernel matrix for kernel ridge regression (as we will see later) - complexity $O(N^3)$ for N training data points
- The computational bottle-neck also limits hyper-parameter tuning

Linear Classification with Linear kernel

- In some cases, we may have **high dimensional input features** such as classification problems involving **textual data** :
  - Classification of data as in Wikipedia, News Articles, Research articles
  - Ranking of queries/Documents in Searching Engines
  - Web-advertising

In these cases, data may exhibit *linear separability*, it might just be enough to use linear classifiers, i.e., no feature mapping $\phi(x)$ is required

- $f(x) = \mathbf{w}^T\mathbf{x}$ is the decision function, i.e. $\phi(\mathbf{x}) = \mathbf{x}$ (aka linear kernel)

[1] https://www.csie.ntu.edu.tw/ cjlin/liblinear/, which is also the underlying solver in scikit-learn for Linear classification problems

Linear Classification with Linear kernel

- In some cases, we may have **high dimensional input features** such as classification problems involving **textual data** :
  - Classification of data as in Wikipedia, News Articles, Research articles
  - Ranking of queries/Documents in Searching Engines
  - Web-advertising

In these cases, data may exhibit *linear separability*, it might just be enough to use linear classifiers, i.e., no feature mapping $\phi(x)$ is required

- $f(x) = \mathbf{w}^T\mathbf{x}$ is the decision function, i.e. $\phi(\mathbf{x}) = \mathbf{x}$ (aka linear kernel)
- How we might check linear separability of the given dataset in high dimensions ?

[1] https://www.csie.ntu.edu.tw/ cjlin/liblinear/, which is also the underlying solver in scikit-learn for Linear classification problems

Linear Classification with Linear kernel

- In some cases, we may have **high dimensional input features** such as classification problems involving **textual data** :
  - Classification of data as in Wikipedia, News Articles, Research articles
  - Ranking of queries/Documents in Searching Engines
  - Web-advertising

In these cases, data may exhibit *linear separability*, it might just be enough to use linear classifiers, i.e., no feature mapping $\phi(x)$ is required

- $f(x) = \mathbf{w}^T\mathbf{x}$ is the decision function, i.e. $\phi(\mathbf{x}) = \mathbf{x}$ (aka linear kernel)
- How we might check linear separability of the given dataset in high dimensions ?
- A very nice and scalable C++ implementation (upto millions of training samples) of linear classification and regression comes in Liblinear solver [1]

[1] https://www.csie.ntu.edu.tw/ cjlin/liblinear/, which is also the underlying solver in scikit-learn for Linear classification problems

Linear SVM

Soft-margin SVM as a regularised learning problem

- We can rewrite the soft-margin SVM problem

$$
\begin{aligned}
\textit{Minimize} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + \tfrac{C}{m}\textstyle\sum_{i=1}^{m} \xi_i \\
\text{Subject to} \quad & \xi_i \geq \max(1 - y_i \mathbf{w}^T \mathbf{x}_i, 0) \\
& \text{for all } i = 1, \ldots, N. \\
& \xi_i \geq 0
\end{aligned}
$$

  equivalently in terms of Hinge loss as

$$
\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_{Hinge}(\mathbf{w}^T \mathbf{x}_i, y_i) + \frac{\lambda}{2}||\mathbf{w}||^2
$$

- This is a so called **regularized learning problem**
  - First term minimizes a loss function on training data
  - Second term, called the **regularizer**, controls the complexity of the model
  - The parameter $\lambda = \frac{1}{C}$ controls the balance between the two terms

Figure: Linear SVM from Machine Learning Basic Principles (2019)

Short text classification - Example from eBay

```
Tickets calgary flames vancouver canucks 2 tickets sept 24 2012 row 2
Stamps  stamps italy kingdom used high value f 9608
Music   brahms j hungarian dances cd new
Jewelry & Watches       5 colors optional hotaru fashion silicone date calendar jelly candy sport watch
Tickets free legoland ticket for friend w legoland pass member expires 8 31 12
Books   specimen book of monotype printing types vol 2 c 1971 garamond to othello
Tickets 4 new york jets vs new england patriots tickets 11 22 sec 336 row 17 10 yd
Art     lowe 1858 antique fern botanical print polypodium lycopodioides
Art     m c escher black white print house of stairs
Books   days in the lives of social workers 54 professionals tell real life stories
Music   bizet g carmen sung in english cd new
Stamps  parliament house cent fdc perth fdi pmk 2
Jewelry & Watches       faceted multigem sterling silver bracelet by silverrushstyle
Tickets michael jackson the immortal tour 8 15 12 staples 2 lower level
```

- Each row is training example in the form of LABEL $\langle$ TAB $\rangle$ INPUT-VECTOR
- In the example above, *Tickets*, *Stamps*, *Music* etc are labels or classes
- The vector representation of the INPUT-VECTOR consists of frequency of each word in the vocabulary (set of all words) or as a sparse representation
- Consider thousands or hundreds of thousand such examples

Polynomial kernel and n-grams

In short text [2] classification (as in previous slide), it might be useful to consider bigrams/trigrams also as features along with normal uni-gram features (this is similar to projecting data to high dimensions)

- bigrams - pairs of words (such as 'calgary flames', 'flames vancouver') in a window of size 2
- trigrams - triplets of words in a window of size 3

[2] in contrast, long text documents might refer to instance with hundreds of words as classification in Wikipedia documents or News stories

Polynomial kernel and n-grams

In short text [2] classification (as in previous slide), it might be useful to consider bigrams/trigrams also as features along with normal uni-gram features (this is similar to projecting data to high dimensions)

- bigrams - pairs of words (such as 'calgary flames', 'flames vancouver') in a window of size 2
- trigrams - triplets of words in a window of size 3

How might we achieve this in the context of kernels ?

[2] in contrast, long text documents might refer to instance with hundreds of words as classification in Wikipedia documents or News stories

Polynomial kernel and n-grams

In short text [2] classification (as in previous slide), it might be useful to consider bigrams/trigrams also as features along with normal uni-gram features (this is similar to projecting data to high dimensions)

- bigrams - pairs of words (such as 'calgary flames', 'flames vancouver') in a window of size 2
- trigrams - triplets of words in a window of size 3

How might we achieve this in the context of kernels ?

- Use a polynomial kernel of degree 2 (bigrams) or 3 (trigrams)

[2] in contrast, long text documents might refer to instance with hundreds of words as classification in Wikipedia documents or News stories

Polynomial kernel and n-grams

In short text [2] classification (as in previous slide), it might be useful to consider bigrams/trigrams also as features along with normal uni-gram features (this is similar to projecting data to high dimensions)

- bigrams - pairs of words (such as 'calgary flames', 'flames vancouver') in a window of size 2
- trigrams - triplets of words in a window of size 3

How might we achieve this in the context of kernels ?

- Use a polynomial kernel of degree 2 (bigrams) or 3 (trigrams)

[2] in contrast, long text documents might refer to instance with hundreds of words as classification in Wikipedia documents or News stories

Explicit feature map for Polynomial kernel with unigram + bi-grams features

Non-linear classification

For the classification function f_3 below :

$$\begin{aligned} f_3(x) = & \, w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + w^{(3)}x^{(3)} \\ & + w^{(4)}x^{(1)}x^{(2)} + w^{(5)}x^{(1)}x^{(3)} \\ & + w^{(6)}x^{(2)}x^{(1)} + w^{(7)}x^{(2)}x^{(3)} \\ & + w^{(8)}x^{(3)}x^{(1)} + w^{(9)}x^{(3)}x^{(2)} \end{aligned}$$

- Here, the decision function $f_3(x)$ maps a vector in $\mathbb{R}^3$ to $\mathbb{R}$
- Non-linear feature map $\phi_3 : \mathbb{R}^3 \mapsto \mathbb{R}^9$, and is given by $\phi_3(x) = (x^{(1)}, x^{(2)}, x^{(3)}, x^{(1)}x^{(2)}, x^{(1)}x^{(3)}, x^{(2)}x^{(1)}, x^{(2)}x^{(3)}, x^{(3)}x^{(1)}, x^{(3)}x^{(2)})^T$
- The learning process is to learn a linear decision boundary in $\mathbb{R}^9$, namely the 9 co-efficients $(w_1, \ldots, w_9)^T$
- **Importantly**, it is much easier to solve a linear system without having to do pairwise computation as require for the kernel matrix

Feature map for Polynomial kernel - Large N, Large D

Two challenges :

- Using kernels i.e. implicit feature mapping - there might be hundreds of thousand training examples N, so computing the kernel matrix of size N^2 becomes challenging.
- Without kernel i.e. explicit feature mapping - there will be D^2 bigrams and D^3 trigrams. Therefore, one has to learn co-efficients which are quadratic/cubic in the data dimensionality

Feature map for Polynomial kernel - Large N, Large D

Two challenges :

- Using kernels i.e. implicit feature mapping - there might be hundreds of thousand training examples N, so computing the kernel matrix of size N^2 becomes challenging.
- Without kernel i.e. explicit feature mapping - there will be D^2 bigrams and D^3 trigrams. Therefore, one has to learn co-efficients which are quadratic/cubic in the data dimensionality

Solution -

- Exploit input data sparsity - create bigrams/trigrams on-demand
- Main idea - create bigram features for only those pairs of words which appear in the training data
- As an example for ebay data, for D=1,000, and sparsity $s = 10$, computation $O(s^2)$ is much smaller than $O(D^2)$
- Then, *learn a linear classifier which is a much easier problem* than non-linear method involving kernel especially with large sample size N.

Feature map for Polynomial kernel - Large N, Large D

Two challenges :

- Using kernels i.e. implicit feature mapping - there might be hundreds of thousand training examples N, so computing the kernel matrix of size N^2 becomes challenging.
- Without kernel i.e. explicit feature mapping - there will be D^2 bigrams and D^3 trigrams. Therefore, one has to learn co-efficients which are quadratic/cubic in the data dimensionality

Solution -

- Exploit input data sparsity - create bigrams/trigrams on-demand
- Main idea - create bigram features for only those pairs of words which appear in the training data
- As an example for ebay data, for D=1,000, and sparsity $s = 10$, computation $O(s^2)$ is much smaller than $O(D^2)$
- Then, *learn a linear classifier which is a much easier problem* than non-linear method involving kernel especially with large sample size N.

Libshorttext (`https://www.csie.ntu.edu.tw/~cjlin/libshorttext/`) python wrapper over Liblinear provides a scalable implemetation.

Term Frequency and Inverse Document Frequency

TfIDF based data representation for sparse text data

- Bag of Words
  - Converts **sequence of words** representation of a document to a **set of words**
- Tf-IDF weighting of every term in the document
  - Term-Frequency $\times$ Inverse Document Frequency
  - Term-Frequency (log normalized) of term t in document d

    $$tf(t, d) = \log(1 + f_{t,d})$$

    where $f_{t,d}$ is the number of times t appears in document d
  - Inverse Document Frequency - How frequent is the term across documents

    $$idf(t) = \log(1 + \frac{N}{n_t})$$

    where n_t is the number of times term t appears from the training set of N documents

Linear vs Kernel Methods

Linear Classification	**Kernel Classification**
Example - LinearSVM using LibLinear	Example - KernelSVM such as LibSVM [3]
Linear Regression	Kernel Regression with Linear Kernel
Linear SVM + Bigram feature	LibSVM + Polynomial Kernel
Con - Requires new explicit features	Pros - Just change the kernel
Pros - Suitable for large sample size	Con - Suitable for small size

Table: Linear vs Non-linear (Kernel) Classification

- With appropriate explicit features for Linear classification, both are statistically equivalent

[3] https://www.csie.ntu.edu.tw/ cjlin/libsvm/

Linear vs Kernel Methods

Linear Classification	Kernel Classification
Example - LinearSVM using LibLinear	Example - KernelSVM such as LibSVM [3]
Linear Regression	Kernel Regression with Linear Kernel
Linear SVM + Bigram feature	LibSVM + Polynomial Kernel
Con - Requires new explicit features	Pros - Just change the kernel
Pros - Suitable for large sample size	Con - Suitable for small size

Table: Linear vs Non-linear (Kernel) Classification

- With appropriate explicit features for Linear classification, both are statistically equivalent
- Try a linear SVM using LibLinear, and LibSVM with Linear Kernel on the same problem for increasing sample sizes

[3] https://www.csie.ntu.edu.tw/ cjlin/libsvm/

Conclusion

- Revisited the idea of linear and non-linear classification
- Representer theorem
  - Proof
  - Implications
- Kernel Ridge Regression
- Kernel matrix bottleneck for large sample size
  - Explicit feature maps for polynomial kernel
  - Sparse data representation

- For more detailed material on Gaussian RKHS and other examples
  - Lecture notes by Arthur Gretton - http://www.gatsby.ucl.ac.uk/~gretton/coursefiles/lecture4_introToRKHS.pdf

Books for further study

- Learning with kernels - Schoelkopf and Smola
- Kernel Methods for Pattern Analysis - Shawe-Taylor and Christianini