بسم الله الرحمن الرحيم

الدكتور انهارده هيتكلم علي ال kernel methods .. بتسمحلك انك ت scale شوية حاجات لما بتشتغل مع linear models فاحنا لما كنا بنتكلم علي linear models و كان في طريقه عشان نشتغل علي non-linear models تعال نشتغل علي mapping phi تودينا من ال original space to a new space وهناك اعمل ال linear algebra بتاعتك ...

تعال نفتكر شوية حاجات ..

Generalized linear models:

- Work with fixed non-linear basis functions
- Hypothesis space is limited
- Optimization is easier (usually convex)

Neural Networks:

- Adaptive non-linear basis functions
- Richer hypothesis space
- Harder to optimize (Draw back) (Usually optimization is non-convex)

ده كان ملخص كدا لل 2 important paradigms في الماشين ليرننج ..

طيب في ال generalized linear models احنا بنحتاج نجيب fixed basis functions .. المشكله في ال cost omputation .. في trick اسمها ال dual ... وديه بتقول ان ال complexity of the computation doesn't have to depend on #of basis functions we have but depends on amount of data we have

فلو كان عندنا low data .. يبقا التكنيكس ديه ممكن تبقا فعّاله عشان انت هتقدر ت map the data .. والمابينج هيعتمد علي حجم الداتا .. وده بيتم بال dual trick .... في معظم الألجورذمز لما بنروح ل new space هي دايماً من نوع phi(x)T * phi(x prime) اللي هو ان ال computation بيبقا dot product ... فلو هو دايماً dot product فانت ممكن ت accelerate الموضوع ده ... فال kernel function هتبقا هي ال dot product between x and x prime في ال new space اللي عندنا .. فلو انت عارف الأوتبوت بتاع ال kernel function لكل pair of points انت مش محتاج تعرف ايه هي ال underlying feature space defined by phi(x) . فالمفتاح هو انك تبين ان الألجورذمز بتعمل ال computation باستخدام الدوت بروضكت .. اللي هي يعني k(x, xprime) .. وبعدها هنبين ان ممكن نعرّف kernel functions اللي بتبقا سريعه جداً عشان ت evaluate و مش محتاجه explicit ..... computation with phi(x)

كمثال تعال نشوف ال linear regression و تعال نشوف ايه اللي بيحصل (in terms of computation) ... فالأول هنشوف ان كل ال computation بتبقا phi T phi وده ال kernel .. وبعدين هنشوف ان ممكن نعرّف ال kernel بحيث ان ال computation is independent of the dimensionality of the space .... فبالنسبه لل linear regression ده كان الهدف بتاعنا .. احنا بنقلل ال squared loss و بنضيف ترم ال regularization ... فاحنا خدنا المشتقه وساويناها بالصفر .. وفي الآخر بدل ما تعزل الw وتجيب الحل .. اكتب الحل بطريقه مختلفه ...

Dual Representations

- Recall linear regression objective

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}[\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - y_n]^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

- Solution: set gradient to 0

$$\nabla E(\boldsymbol{w}) = \sum_n(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - y_n)\phi(\boldsymbol{x}_n) + \lambda\boldsymbol{w} = 0$$

$$\boldsymbol{w} = -\frac{1}{\lambda}\sum_n(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - y_n)\phi(\boldsymbol{x}_n)$$

$\therefore$ **$\boldsymbol{w}$ is a linear combination of inputs in feature space**

$$\{\phi(\boldsymbol{x}_n)|1 \leq n \leq N\}$$

newname
10/9/2021 7:29:42 PM

لما هنيجي نعوض عن الدابليو ب الإكسبرشن اللي قدامك ده .. و هنقول ان الفاي هي الماتركس بتاعت كل النقط اللي عندك ... فالفاي بتاعت إكس 1 .. هي ال 1st data point .. فالفاي ماتركس ديه كلها هي ال concatenation of data points
اللي نقدر نعملو ناو اننا نعامل ال a علي اساس انو variable بدل W .. عشان الفاي هي الداتا في النيو سبيس .. ديه حاجه عندنا اصلا .. فلما هت vary w هو فعلياً كإنك بت vary a .. فانت كمان ممكن تحط في اعتبارك ال a كإنها new variable .. قال dual trick هنا إنك تعمل ال optimization into a different space فاللي بيحصل ان عندنا داتا في السبيس و عندنا مابينج فاي بتودينا للسبيس التاني .. بس كمان انت عندك فاريبل W هتعملو mapping ل dual space قال a هنا هي مجرد equivalent of w بس في سبيس مختلف

- Substitute $\mathbf{w} = \boldsymbol{\Phi}\boldsymbol{a}$
- Where $\boldsymbol{\Phi} = [\phi(\boldsymbol{x}_1)\ \phi(\boldsymbol{x}_2)\ \ldots\ \phi(\boldsymbol{x}_N)]$

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \quad \text{and } a_n = -\frac{1}{\lambda}(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - y_n)$$

- Dual objective: minimize E with respect to $\boldsymbol{a}$

$$E(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{a} - \boldsymbol{a}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{y} + \frac{\boldsymbol{y}^T\boldsymbol{y}}{2} + \frac{\lambda}{2}\boldsymbol{a}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{a}$$

$\boldsymbol{K}$

newname
10/9/2021 7:48:18 PM

الدكتور بيقولك ان لما بتعمل ال computation to the dual space احنا عندنا different complexit for the computation ... ال original optimization في المسأله الأصليه اللي عندنا
بالنسبه لل w كانت بتعتمد علي عدد ال basis functions .. فلما بعمل ال optimization with respect to w انا عندي variable W for every basis function .. بس لما هتشتغل في ال dual space .. هتشتغل علي ال a بدل ال w ..فالكوميلكستي هتعتمد علي كمية الداتا مش ال dimensionality of space ...

- Prediction: $y_* = \phi(\boldsymbol{x}_*)^T\boldsymbol{\Phi}\boldsymbol{a}$
  $= k(\boldsymbol{x}_*, \boldsymbol{X})(\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}$
- Linear regression where we find dual solution $\boldsymbol{a}$ instead of primal solution $\mathbf{w}$.
- Complexity:
  - Primal solution: depends on # of basis functions
  - Dual solution: depends on amount of data
    - Advantage: can use very large # of basis functions
    - Just need to know kernel k

ازاي نوصل لل complexit بتعتمد علي حجم الداتا هيعتمد علي طريقه نحسب بيها ال kernels اللي هم ال gram matrix من غير ما نضطر اننا ن refer to the points into the new space .. فانت محتاج بيقا عندك kernel K .. ديه بتسمحلك انك تلاقي ال dot products الي انت عاوزهم دول من غير ما تحسب ال dot product ... في طرق كتيره تقدر بيها تعرف kernel functions اللي هت implicitly correspond to dot products

..... انت مش محتاج تعمل ال dot products هو في طريقه مباشره اللي هي ال kernel ... فانت عاوز ال Gram matrix K وانت عارف انها الفاي ترانسبوز فاي .. فبدل ما تروح تحسب الدوت برودكت .. علي طول specify K .. فانا لو عملت كدا هنا في شوية فانكشنز تمام هيقدرو يعملو كدا ..بس في فانكشنز مش هينفع .. فهي مش هتبقا arbitrary function .. لازم تختار phi.T phi functions that have an implicit correspondence with phi

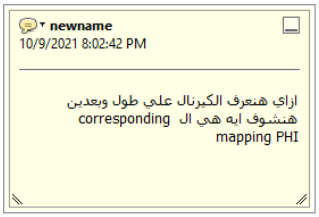
..... فالفانكشنز اللي ليها implicit correspondence هيبقا لازم يكونوا positive semi-definite ... الماتركس بتبقا positive semi-definite لما تكون تقدر ت factor الماتركس ديه ل one matrix phi و تضربها في نفسها هيديلك ال K تاني .. فعشان كدا ال Gram matrix or kernel function هتحتاج تبقا positive semi defininte .. معناها بس ان لازم يبقا في ماتركس فاي تقدر تضربها ف نفسها .. فلو بنتكلم linear algebra هي هي انك تقول ان ليها eigenvalues أكبر من او بتساوي الصفر ..

Constructing Kernels

- Two possibilities:
  - Find mapping $\boldsymbol{\phi}$ to feature space and let $\boldsymbol{K} = \boldsymbol{\phi}^T\boldsymbol{\phi}$
  - Directly specify $\boldsymbol{K}$

- Can any function that takes two arguments serve as a kernel?
- No, a valid kernel must be positive semi-definite
  - In other words, k must factor into the product of a transposed matrix by itself (e.g., $\boldsymbol{K} = \boldsymbol{\phi}^T\boldsymbol{\phi}$)
  - Or, all eigenvalues must be greater than or equal to 0.

Example



- Let $k(\boldsymbol{x}, \boldsymbol{z}) = \left(\boldsymbol{x}^T\boldsymbol{z}\right)^2$

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

$$k(x,z) = (x^T z)^2$$
$$= (x_1 z_1 + x_2 z_2)^2$$
$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2$$
$$= (x_1^2, \sqrt{2}\, x_2 x_1, x_2^2) \begin{bmatrix} z_1^2 \\ \sqrt{2}\, z_1 z_2 \\ z_2^2 \end{bmatrix} = \varphi(x)^T \cdot \varphi(z)$$

φ_1 φ_2 φ_3

طيب في فكره عشان نقدر نبني ال kernels هو ان في basic rules نقدر نمشي وراها .. اللي هو لو عندي kernels اصلا من الأول .. فاقدر أ compose أكتر من كيرنال عشان يبقا عندي kernels تانيين ....

Rules to construct Kernels

- Let $k_1(\boldsymbol{x}, \boldsymbol{x}')$ and $k_2(\boldsymbol{x}, \boldsymbol{x}')$ be valid kernels
- The following kernels are also valid:
  1. $k(\boldsymbol{x}, \boldsymbol{x}') = ck_1(\boldsymbol{x}, \boldsymbol{x}') \quad \forall c > 0$
  2. $k(\boldsymbol{x}, \boldsymbol{x}') = f(\boldsymbol{x})k_1(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x}') \quad \forall f$
  3. $k(\boldsymbol{x}, \boldsymbol{x}') = q(k_1(\boldsymbol{x}, \boldsymbol{x}'))$ q is polynomial with coeffs ≥ 0
  4. $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(k_1(\boldsymbol{x}, \boldsymbol{x}'))$
  5. $k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}')$
  6. $k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}')k_2(\boldsymbol{x}, \boldsymbol{x}')$
  7. $k(\boldsymbol{x}, \boldsymbol{x}') = k_3(\phi(\boldsymbol{x}), \phi(\boldsymbol{x}'))$
  8. $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}'$ $\boldsymbol{A}$ is symmetric positive semi-definite
  9. $k(\boldsymbol{x}, \boldsymbol{x}') = k_a(\boldsymbol{x_a}, \boldsymbol{x}'_a) + k_b(\boldsymbol{x_b}, \boldsymbol{x}'_b)$
  10. $k(\boldsymbol{x}, \boldsymbol{x}') = k_a(\boldsymbol{x}_a, \boldsymbol{x}'_a)k_b(\boldsymbol{x}_b, \boldsymbol{x}'_b)$

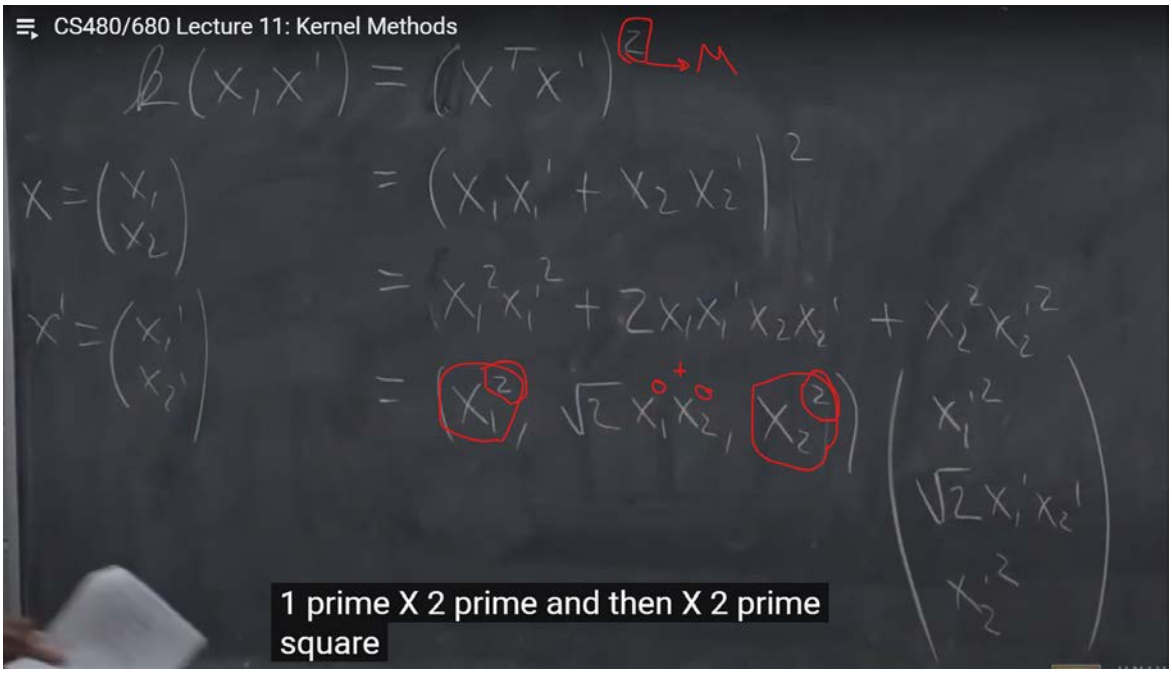
where $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{pmatrix}$

University of Waterloo CS480/680 Spring 2019 Pascal Poupart 12

في شوية كيرنالز موجودين في ارض الواقع من ضمنهم ال polynomial kernel ... هنا انت بتاخد الدوت بروضكت في الأوريجنال سبيس وبعدين ترفعها لباور .. لما ترفع لباور M .. هتجيب polynomial expression of degree M .. الفيتشر سبيس هو ال all degree M products of entries in x .... لو عندك ال x and x prime are 2 images .. في الحاله ديه ال feature space هيبقا كل الضرب بتاع ال M pixel intensities ..... تعال نرجع للمثال اللي فات ... كان في باور 2 .. فهنا ال M = 2 .. و ال implicit feature space هيبقو monomials of degree M ..

الكيرنالز هي a special construction that allows dot product between pairs of data points ... وده مفيد لو هتشتغل في ال generalized linear models و هنا انت بتبدأ ب space وبعدين تعمل mapping phi توديك ل new space وبعدين توديك تعمل regression and classification هناك

المشكله اللي شفناها ان لما بتروح لل new space .. بيبقا ليه high dimensionality وبتبقا عاوزلو lots of basis function فبتدفع التمن computation cost ... انما بالكيرنالز .. كل الحسابات عباره عن dot product في ال new space .. فال kernel بتبقا في ال new space ايه هي ال dot product مابين pairs of points و دلوقت لما بتقدر تحدد ده علي طول من غير ما تضطر تروح لل new space و تعمل هناك ال dot product فانت تقدر ت save the computation و تدفع التمن بتاع ال dimensionality في ال original space ...



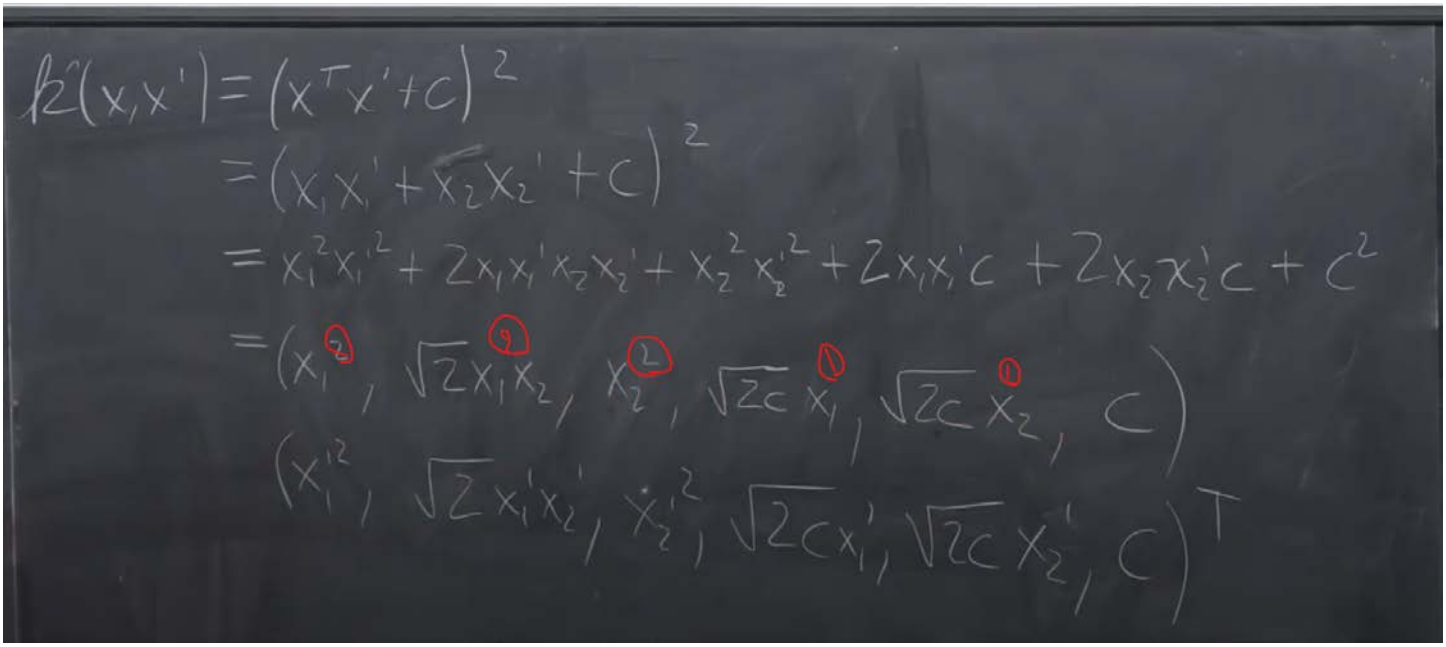
تعال نتكلم علي مثال لو خدنا ال x and x prime بيقو صورتين .. الإكس هيبقا فيكتور جواه pixel intensites .. لما هتروح لل new feature space عن طريق انك تاخد ال x transpose x prime وترفعهم لباور .. انت هتجيب فيتشرز تانيه .. هتبقا الضرب بتاع every combination of M pixels together where some pixels might repeat .. فالجمال هنا ان دلوقت انت بقا عندك richer space وبعدين تتمني ان في ال ritcher space اللي عندك ده تقدر تلاقي فانكشن اللي هي linear و implicitly nonlinear in the original space اللي بي captures the classes في الأوريجنال سبيس .... طيب ده بيشتغل كويس لو انت عاوز فيتشرز ليها degree M .. طب بالنسبه للفيتشرز بتاعت كل الدجريز up to M .. فهنا انت بتضيف ثابت اسمو C .. فده كدا انت بتاخد في الاعتبار عندك كل ال degrees up to M ...

Common Kernels

- Polynomial kernel: $k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}')^M$
  - M is the degree
  - Feature space: all degree M products of entries in $\boldsymbol{x}$
  - Example: Let $\boldsymbol{x}$ and $\boldsymbol{x}'$ be two images, then feature space could be all products of M pixel intensities

- More general polynomial kernel:
  $$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}' + c)^M \text{ with } c > 0$$
  - Feature space: all products of up to M entries in $\boldsymbol{x}$

مثال علي كدا:



هنشوف مثال علي كيرنال تاني .. جاوسين كيرنال

Common Kernels

- Gaussian Kernel: $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\left|\left|\boldsymbol{x}-\boldsymbol{x}'\right|\right|^2}{2\sigma^2}\right)$
- Valid Kernel because:

- Implicit feature space is infinite!