CPSC 340: Machine Learning and Data Mining

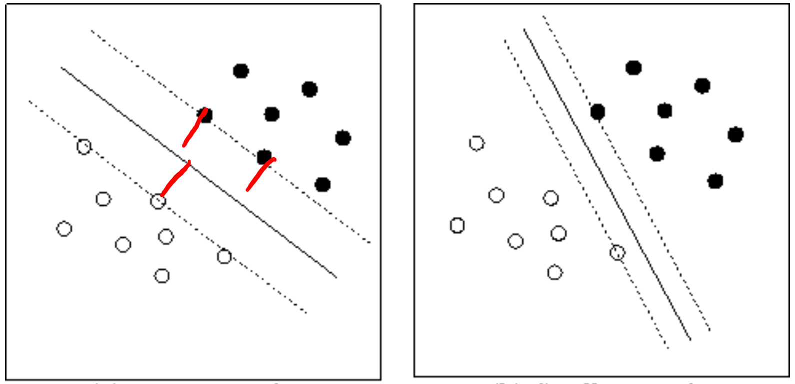
Kernel Methods

Admin

- Assignment 3:
  - Due Sunday evening
  - Solutions will be posted on Thursday
- Assignments 1 and 2:
  - You can now see each other’s work
- Midterm March 1
  - Past exams posted
  - Midterm covers Assignments 1-3 / lectures 1-16
  - Tutorials after break will cover practice exam questions
  - In class, 1pm-1:55pm, closed-book, 1 page double-sided “cheat sheet”.

Last Time: SVMs and Kernel Trick

- We discussed the maximum margin view of SVMs:
  - Yields an L2-regularized hinge loss.



(a) Larger margin (b) Smaller margin

- We introduced the kernel trick:
  - Write model to only depend on inner products between features vectors.

$$\hat{y} = \hat{K}(K + \lambda I)^{-1} y$$

$t \times n$ matrix $\hat{Z}Z^T$ containing inner products between test examples and training examples.

$n \times n$ matrix ZZ^T containing inner products between all training examples.

  - So everything we need to know about z_i is summarized by the $z_i^T z_j$.
  - If you have a kernel function $k(x_i,x_j)$ that computes $z_i^T z_j$, then you don't need to compute the basis z_i explicitly.

http://svr-www.eng.cam.ac.uk/~kkc21/thesis_main/node12.html

Polynomial Kernel with Higher Degrees

- Assume that I have 2 features and want to use the degree-2 basis:

$$z_i = \begin{bmatrix} 1 & \sqrt{2}x_{i1} & \sqrt{2}x_{i2} & x_{i1}^2 & \sqrt{2}x_{i1}x_{i2} & x_{i2}^2 \end{bmatrix}^T$$

- I can compute inner products using:

$$(1 + x_i^T x_j)^2 = 1 + 2x_i^T x_j + (x_i^T x_j)^2$$

$$= 1 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}^2 x_{j2}^2$$

$$= \underbrace{\begin{bmatrix} 1 & \sqrt{2}x_{i1} & \sqrt{2}x_{i2} & x_{i1}^2 & \sqrt{2}x_{i1}x_{i2} & x_{i2}^2 \end{bmatrix}}_{z_i^T} \underbrace{\begin{bmatrix} 1 \\ \sqrt{2}x_{j1} \\ \sqrt{2}x_{j2} \\ x_{j2}^2 \\ \sqrt{2}x_{j1}x_{j2} \\ x_{j2}^2 \end{bmatrix}}_{z_j}$$

$$= z_i^T z_j$$

Polynomial Kernel with Higher Degrees

- To get all degree-4 "monomials" I can use:

$$z_i^T z_j = (x_i^T x_j)^4$$

Equivalent to using a z_i with weighted versions of $x_{i1}^4, x_{i1}^3 x_{i2}, x_{i1}^2 x_{i2}^2, x_{i1} x_{i2}^3, x_{i2}^4, \ldots$

- To also get lower-order terms use $z_i^T z_j = (1 + x_i^T x_j)^4$
- The general degree-p polynomial kernel function:

$$k(x_i, x_j) = (1 + x_i^T x_j)^p$$

  – Works for any number of features 'd'.
  – But cost of computing $z_i^T z_j$ is $O(d)$ instead of $O(d^p)$.
- Take-home message: I can take the dot products without constructing the feature vectors themselves.

Kernel Trick

- Using polynomial basis of degree 'p' with the kernel trick:
  - Compute K and $\widehat{K}$:

$$K_{ij} = (1 + x_i^T x_j)^p \qquad \hat{K}_{ij} = (1 + \underbrace{\hat{x}_i^T}_{\text{test example}} \underbrace{x_j}_{\text{train example}})^p$$

  - Make predictions using:

$$\underbrace{\hat{y}}_{t\times 1} = \underbrace{\hat{K}}_{t\times n} \underbrace{(K + \lambda I)^{-1}}_{n\times n} \underbrace{y}_{n\times 1}$$

To form $K = ZZ^T$

- Training cost is only $O(n^2d + n^3)$, despite using $O(d^p)$ features.

To invert $n\times n$ matrix

  - Testing cost is only $O(ndt)$.

To form $\hat{K} = \hat{Z}Z^T$

Linear Regression vs. Kernel Regression

Linear Regression

Training

1. Form basis Z from X.
2. Compute $w = (Z^T Z + \lambda I)^{-1} \backslash (Z^T y)$

(Recall the "backslash" notation for solving a linear system)

Testing

1. Form basis $\hat{Z}$ from $\hat{X}$
2. Compute $\hat{y} = \hat{Z} w$

Kernel Regression

Training:

1. Form inner products K from X.
2. Compute $v = (K + \lambda I)^{-1} \backslash y$

Non-parametric

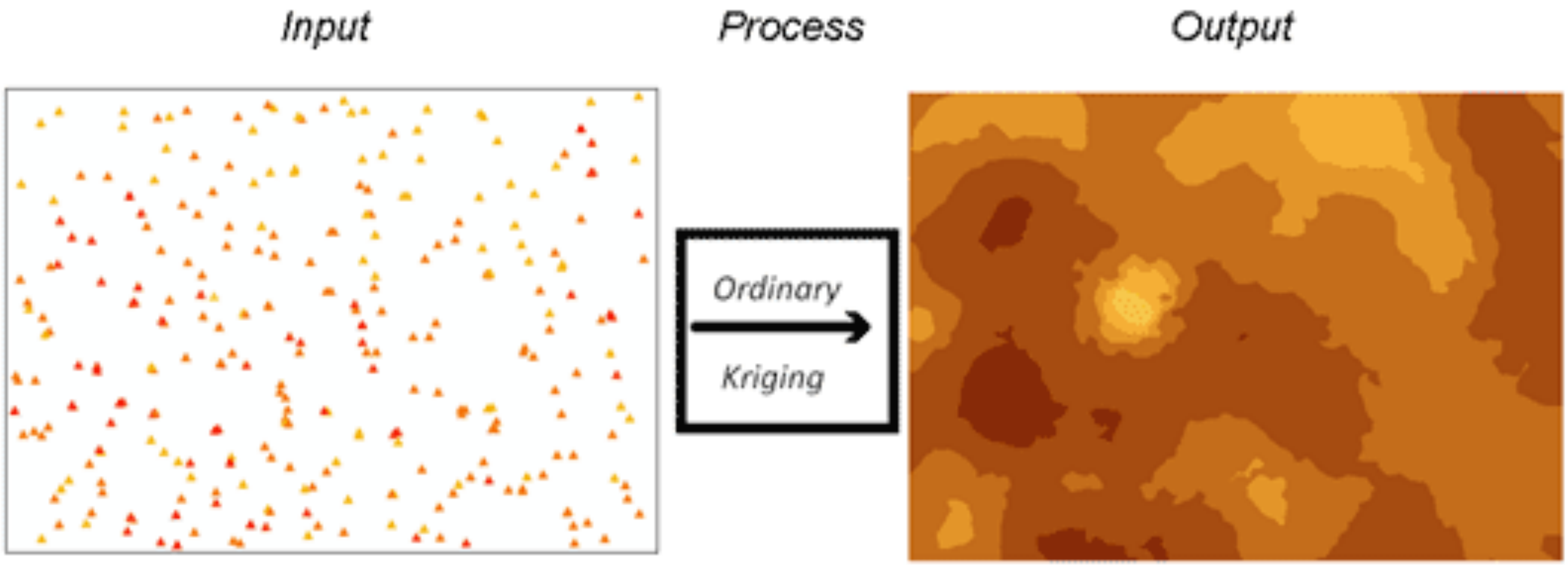
Testing:

1. Form inner products $\hat{K}$ from X and $\hat{X}$
2. Compute $\hat{y} = \hat{K} v$

Observation: this requires all training examples ☹

Motivation: Finding Gold

- Kernel methods first came from mining engineering ('Kriging'):
  - Mining company wants to find gold.
  - Drill holes, measure gold content.
  - Build a kernel regression model (typically use RBF kernels).



Gaussian-RBF Kernel

- Most common kernel is the Gaussian RBF kernel:

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

- Same formula and behaviour as RBF basis, but not equivalent:
  – Before we used RBFs as a basis, now we're using them as inner-product.
- Basis z_i giving the Gaussian RBF kernel is infinite-dimensional.
  – Not much hope of doing this without the kernel trick…
- Kernel trick lets us fit regression models without explicit features:
  – We can interpret $k(x_i,x_j)$ as a "similarity" between objects x_i and x_j.
  – We don't need z_i and z_j if we can compute 'similarity' between objects.

Kernel trick for structured data

- Consider data that doesn't look like this:

$$X = \begin{bmatrix} 0.5377 & 0.3188 & 3.5784 \\ 1.8339 & -1.3077 & 2.7694 \\ -2.2588 & -0.4336 & -1.3499 \\ 0.8622 & 0.3426 & 3.0349 \end{bmatrix}, \quad y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix},$$

- But instead looks like this:

$$X = \begin{bmatrix} \text{Do you want to go for a drink sometime?} \\ \text{J'achète du pain tous les jours.} \\ \text{Fais ce que tu veux.} \\ \text{There are inner products between sentences?} \end{bmatrix}, y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix}.$$

- Instead of using features, can define kernel between sentences.
  - E,g, "string kernels": weighted frequency of common subsequences.
- There are also "image kernels", "graph kernels", and so on…

Valid Kernels

- What kernel functions $k(x_i,x_j)$ can we use?

- Kernel ‘k’ must be an inner product in some space:
  - There must exist a mapping from x_i to some z_i such that $k(x_i,x_j) = z_i^T z_j$.

- It can be hard to show that a function satisfies this.
- But there are some simple rules for constructing valid kernels from other valid kernels (bonus slide).

Kernel Trick for Other Methods

- Besides L2-regularized least squares, when can we use kernels?
  - Methods based on Euclidean distances between examples:
    - Kernel k-nearest neighbours.
    - Kernel clustering (k-means, DBSCAN, hierarchical).
    - Kernel outlierness.
    - Kernel "Amazon Product Recommendation".
    - Kernel non-parametric regression.

$$\|z_i - z_j\|^2 = z_i^T z_i - 2 z_i^T z_j + z_j^T z_j = k(x_i, x_i) - 2k(x_i, x_j) + k(x_j, x_j)$$

  - L2-regularized linear models ("representer theorem" -- see bonus slides):
    - L2-regularized robust regression.
    - L2-regularized logistic regression.
    - L2-regularized support vector machines.

less obvious but true

With a particular implementation testing cost is reduced from $O(ndt)$ to $O(\underbrace{m}_{\text{Number of support vectors.}}dt)$

Kernel trick continued

- Because of the support vectors, kernels are used with SVMs quite often, but much less so with logistic regression.

sklearn.svm.SVC

class `sklearn.svm.` **SVC** (*C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape=None, random_state=None*) [source]

sklearn.linear_model.LogisticRegression

class `sklearn.linear_model.` **LogisticRegression** (*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1*) [source]

RBF kernel vs RBF features

- Like the RBF features, the RBF kernel…
  - can learn any decision boundary given enough data
  - as a result it is prone to overfitting, so we need to use regularization
  - σ parameter controls smoothness: larger σ means smoother boundaries
    - This is called ”gamma” in sklearn and it’s 1/σ
  - λ parameter controls regularization: larger λ means more regularization
    - This is called “C” in sklearn and it’s 1/λ

Summary

- Kernels let us use similarity between objects, rather than features.
- The RBF kernel allows us to use infinitely many features in finite computational time.

- We'll spend the rest of today's class reviewing recent topics.

Bonus Slide: Features Corresponding to RBF Kernel

Guasian-RBF Kernels

- The most common kernel is the Gaussian-RBF (or 'squared exponential') kernel,

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right).$$

- What function $\phi(x)$ would lead to this as the inner-product?
  - To simplify, assume $d = 1$ and $\sigma = 1$,

$$\begin{aligned} k(x_i, x_j) &= \exp(-x_i^2 + 2x_i x_j - x_j^2) \\ &= \exp(-x_i^2)\exp(2x_i x_j)\exp(-x_j^2), \end{aligned}$$

  so we need $\phi(x_i) = \exp(-x_i^2)z_i$ where $z_i z_j = \exp(2x_i x_j)$.
    - For this to work for *all* x_i and x_j, z_i must be infinite-dimensional.
  - If we use that

$$\exp(2x_i x_j) = \sum_{k=0}^{\infty} \frac{2^k x_i^k x_j^k}{k!},$$

  then we obtain

$$\phi(x_i) = \exp(-x_i^2)\begin{bmatrix} 1 & \sqrt{\frac{2}{1!}}x_i & \sqrt{\frac{2^2}{2!}}x_i^2 & \sqrt{\frac{2^3}{3!}}x_i^3 & \cdots \end{bmatrix}.$$

Bonus Slide: Designing Valid Kernel Functions

Constructing Valid Kernels

- If $k_1(x_i, x_j)$ and $k_2(x_i, x_j)$ are valid kernels, then the following are valid kernels:
  - $k_1(\phi(x_i), \phi(x_j))$.
  - $\alpha k_1(x_i, x_j) + \beta k_2(x_i, x_j)$ for $\alpha \geq 0$ and $\beta \geq 0$.
  - $k_1(x_i, x_j)k_2(x_i, x_j)$.
  - $\phi(x_i)k_1(x_i, x_j)\phi(x_j)$.
  - $\exp(k_1(x_i, x_j))$.
- Example: Gaussian-RBF kernel:

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right)$$

$$= \underbrace{\exp\left(-\frac{\|x_i\|^2}{\sigma^2}\right)}_{\phi(x_i)} \underbrace{\exp\left(\underbrace{\frac{2}{\sigma^2}}_{\alpha \geq 0} \underbrace{x_i^T x_j}_{\text{valid}}\right)}_{\exp(\text{valid})} \underbrace{\exp\left(-\frac{\|x_j\|^2}{\sigma^2}\right)}_{\phi(x_j)}.$$

Bonus Slide: Kernels for Linear Model plus L2-Reg

Representer Theorem

- Consider linear model differentiable with losses f_i and L2-regularization,

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2.$$

- Setting the gradient equal to zero we get

$$0 = \sum_{i=1}^{n} f_i'(w^T x_i) x_i + \lambda w.$$

- So any solution w^* can written as a linear combination of features x_i,

$$\begin{aligned} w^* &= -\frac{1}{\lambda}\sum_{i=1}^{n} f_i'((w^*)^T x_i) x_i = \sum_{i=1}^{n} z_i x_i \\ &= X^T z. \end{aligned}$$

- This is called a representer theorem (true under much more general conditions).

Bonus Slide: Kernels for Linear Model plus L2-Reg

Representer Theorem

- Using representer theorem we can use $w = X^T z$ in original problem,

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2$$

$$= \underset{z \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{n} f_i(\underbrace{z^T X x_i}_{x_i^T X^T z}) + \frac{\lambda}{2}\|X^T z\|^2$$

- Now defining $f(z) = \sum_{i=1}^{n} f_i(z_i)$ for a vector z we have

$$= \underset{z \in \mathbb{R}^n}{\operatorname{argmin}} f(XX^T z) + \frac{\lambda}{2} z^T X X^T z$$

$$= \underset{z \in \mathbb{R}^n}{\operatorname{argmin}} f(Kz) + \frac{\lambda}{2} z^T K z.$$

- Similarly, at test time we can use the n variables z,

$$\hat{X} w = \hat{X} X^T z = \hat{K} z.$$