Kernel methods

Narayana Santhanam

EE 645
Jan 22, 2023

University of Hawaiʻi Mānoa

Linear $\rightarrow$ Non-linear

Two approaches:

Kernel methods

Linear $\rightarrow$ Non-linear

Two approaches:

Kernel methods

guarantees, well understood

Linear $\rightarrow$ Non-linear

Two approaches:

Kernel methods

guarantees, well understood

computationally intensive (small data)

Neural networks

less well understood

Linear $\rightarrow$ Non-linear

Two approaches:

Kernel methods

- guarantees, well understood
- computationally intensive (small data)

Neural networks

- less well understood
- computationally cheap (large data)

Next couple of weeks

Kernel methods

Next couple of weeks

Kernel methods

High level picture

Next couple of weeks

Kernel methods

High level picture
Support Vector classification and regression

Next couple of weeks

Kernel methods

High level picture
Support Vector classification and regression
Kernel PCA

Next couple of weeks

Kernel methods

High level picture
Support Vector classification and regression
Kernel PCA
Random Fourier Futures

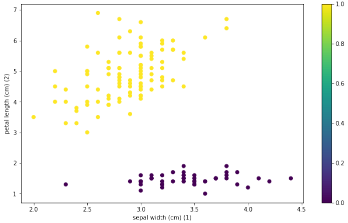
Next couple of weeks

Kernel methods

High level picture
Support Vector classification and regression
Kernel PCA
Random Fourier Futures
Credit risk, Power data

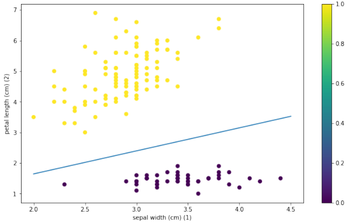
Classification

Linearly separable



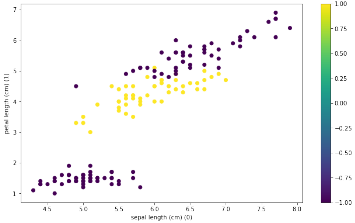
Classification

Linearly separable



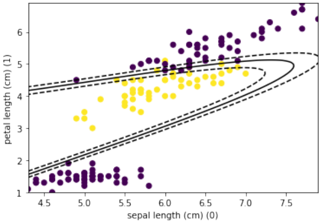
Classification

Not linearly separable



Classification

Not linearly separable



Boundaries not linear..

but are linear in a higher dimensional space!

Closer look

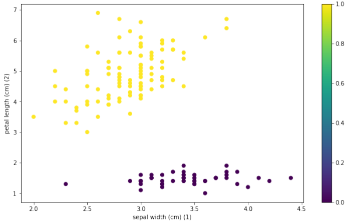
Principled approach for linear $\rightarrow$ nonlinear
Powerful, yet generalizes well
Often explainable
    at least more than other state of art
New advances increase reach (more data)
    but not as much as NN

University of Hawai'i Mānoa

Stepping back

Linearly separable

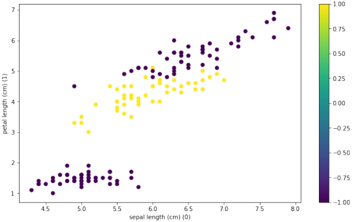
Where would you draw a *linear* classifier?



Stepping back

Not linearly separable

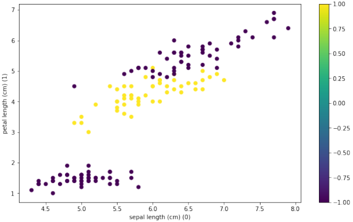
Where would you draw a *linear* classifier?



Stepping back

Not linearly separable

Where would you draw a *linear* classifier?

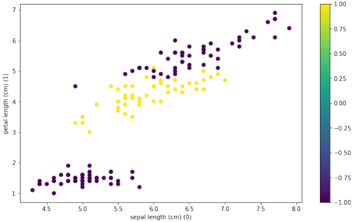


Minimum number of mistakes?

Stepping back

Not linearly separable

Where would you draw a *linear* classifier?



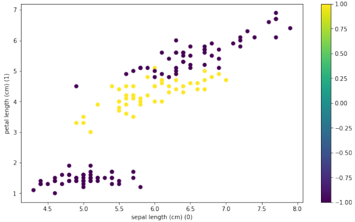
Minimum number of mistakes?

infeasible, NP-hard

Stepping back

Not linearly separable

Where would you draw a *linear* classifier?



Minimum number of mistakes?

infeasible, NP-hard

Instead: variation of ℓ_1 loss, sum of margins

Setting up linear classification

Distance of $\mathbf{x}$ from a plane $\mathbf{w}^T\mathbf{x} - b = 0$?

Setting up linear classification

Distance of $\mathbf{x}$ from a plane $\mathbf{w}^T\mathbf{x} - b = 0$?

$$\frac{\mathbf{w}^T\mathbf{x} - b}{||\mathbf{w}||}$$

Formulation of Support Vector Classification

Analyze this in some detail

Formulation of Support Vector Classification

- Analyze this in some detail
- One of the key advantages: interpretability
    - Comes through formulation

Formulation of Support Vector Classification

Analyze this in some detail

One of the key advantages: interpretability

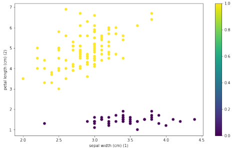
- Comes through formulation

Primal/Dual formulation is a key optimization idea

- accelerations of training neural networks
- resource allocation/optimization in economics,
  - urban planning

University of Hawai'i Mānoa

Formulation

Linearly separable points: 

Training data:

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$$

Margin (closest distance from separating boundary)

$$\gamma(\mathbf{w}, b) = \min_{\mathbf{x}_i} \frac{|\mathbf{w}^T \mathbf{x}_i - b|}{||\mathbf{w}||}$$

Redundant parameterization scaling $\mathbf{w}, b$ by any number does not change the margin

Formulation

Support vector machine formulation:

$$\mathbf{w}^*, b^* = \arg\max_{\mathbf{w}, b} \gamma(\mathbf{w}, b)$$

subject to $y_i(\mathbf{w}^T \mathbf{x}_i - b) \geq 0$ (for all training $(\mathbf{x}_i, y_i)$)

Redundant parameterization: scaling $\mathbf{w}$, b changes nothing

- only the directions/intercepts matter
- represent by scaling of $\mathbf{w}$, b that ensures

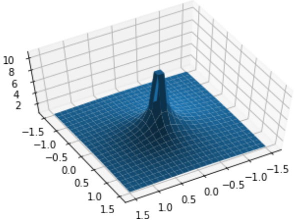
$$\min_{\mathbf{x}_i} |\mathbf{w}^T \mathbf{x}_i - b| = 1$$

Formulation

Support vector machine formulation
(removing the redundant formulation)

$$\mathbf{w}^*, b^* = \arg\max_{\mathbf{w},b} \frac{1}{||\mathbf{w}||}$$

subject to $y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1$ (for all training $(\mathbf{x}_i, y_i)$)



Formulation

$\frac{1}{||\mathbf{w}||}$ not convex/concave

But it is easy to come with a convex formulation

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \frac{1}{2}||w||^2$$

subject to $y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1$ (for all pairs $(\mathbf{x}_i, y_i)$)

A bit on convex optimization

General concepts beyond support vector machine formulation

A bit on convex optimization

General concepts beyond support vector machine formulation

Lagrangian:

$$L(\mathbf{w}, b, \Lambda) = \frac{1}{2}||w||^2 + \sum_i \lambda_i \left(1 - y_i(\mathbf{w}^T \mathbf{x}_i - b)\right)$$

$L(\mathbf{w}, b, \Lambda)$ is a key idea in any constrained optimization

Primal/dual formulation

Neat property of Lagrangians:

Optimal point

$\min_{\mathbf{w},b} \max_{\Lambda \geq 0} L(\mathbf{w}, b, \Lambda)$ (primal formulation)

Dual formulation (Nash, von Neumann)

$\max_{\Lambda \geq 0} \min_{\mathbf{w},b} L(\mathbf{w}, b, \Lambda)$ (dual formulation)

Primal/dual formulation

Neat property of Lagrangians:

Optimal point

$\min_{\mathbf{w},b} \max_{\Lambda \geq 0} L(\mathbf{w}, b, \Lambda)$ (primal formulation)

Dual formulation (Nash, von Neumann)

$\max_{\Lambda \geq 0} \min_{\mathbf{w},b} L(\mathbf{w}, b, \Lambda)$ (dual formulation)

Incidentally Nash won both the Nobel Prize in Econ and the Abel Prize

Primal/dual formulation

For free:

$$\min_{\mathbf{w},b} \max_{\Lambda \geq 0} L(\mathbf{w}, b, \Lambda) \geq \max_{\Lambda \geq 0} \min_{\mathbf{w},b} L(\mathbf{w}, b, \Lambda)$$

But equality in many cases

- in many convex formulations, including our current case
- so one could solve either version
- dual in kernel methods very insightful for explainability

Dual formulation: 2 key insights

- Setting gradient of $L(\mathbf{w}, b, \Lambda)$ to 0, optimal $\mathbf{w}^*$ satisfies

$$\mathbf{w}^* = \sum_i \lambda_i y_i \mathbf{x}_i$$

Representer theorem: solution $\mathbf{w}^*$ is linear combination of inputs

Dual formulation: 2 key insights

- Setting gradient of $L(\mathbf{w}, b, \Lambda)$ to 0, optimal $\mathbf{w}^*$ satisfies

$$\mathbf{w}^* = \sum_i \lambda_i y_i \mathbf{x}_i$$

Representer theorem: solution $\mathbf{w}^*$ is linear combination of inputs

- Finding Λs

$$\max_{\Lambda \geq 0} \sum \lambda_i - \frac{1}{2} \begin{bmatrix} \lambda_1 y_1 & \ldots & \lambda_n y_n \end{bmatrix} XX^T \begin{bmatrix} \lambda_1 y_1 \\ \vdots \\ \lambda_n y_n \end{bmatrix}$$

Data only shows up through dot products (XX^T)

Crux of Kernel approach to nonlinearity

What if not linearly separable?

$$\mathbf{w}^*, b^* = \arg\min \frac{1}{2}||\mathbf{w}||^2$$

subject to $y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1$ (for all training $(\mathbf{x}_i, y_i)$)

In this case, no $\mathbf{w}, b$ pair will satisfy all constraints
In some cases, $y_i(\mathbf{w}^T\mathbf{x}_i - b) = 1 - \xi_i$ for some $\xi_i \geq 0$
If $\xi_i > 1$, the point is misclassified

Slack ξ_i: hinge loss

For each example, $y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \xi_i$ (for $\xi_i \geq 0$)
if $\xi = 0$ then inequality, if $\xi > 0$ equality

Equivalently, $\xi_i = \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b))$ (Hinge loss)

Non-linear separable formulation

$$\mathbf{w}^*, b^* = \arg\min \frac{1}{2}||\mathbf{w}||^2 + C\sum \xi_i$$

subject to $y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \xi_i$ (for all training $(\mathbf{x}_i, y_i)$)
$\xi \geq 0$ (or $-\xi_i \leq 0$)

Dual formulation

$$\max_{C \geq \Lambda \geq 0} \sum \lambda_i - \frac{1}{2}\begin{bmatrix}\lambda_1 y_1 & \dots & \lambda_n y_n\end{bmatrix} XX^T \begin{bmatrix}\lambda_1 y_1 \\ \vdots \\ \lambda_n y_n\end{bmatrix}$$

Dual formulation: 2 key insights

- Setting gradient of $L(\mathbf{w}, b, \{\xi_i\}, \Lambda, \{\mu_i\})$ to 0, optimal $\mathbf{w}^*$ satisfies

$$\mathbf{w}^* = \sum_i \lambda_i y_i \mathbf{x}_i$$

  Representer theorem: solution $\mathbf{w}^*$ is linear combination of inputs

Dual formulation: 2 key insights

- Setting gradient of $L(\mathbf{w}, b, \{\xi_i\}, \Lambda, \{\mu_i\})$ to 0, optimal $\mathbf{w}^*$ satisfies

  $$\mathbf{w}^* = \sum_i \lambda_i y_i \mathbf{x}_i$$

  Representer theorem: solution $\mathbf{w}^*$ is linear combination of inputs

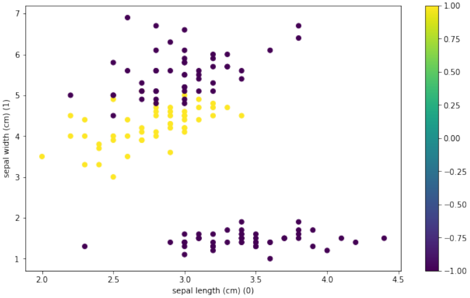
- Finding Λs

  $$\max_{C \geq \Lambda \geq 0} \sum \lambda_i - \frac{1}{2} \begin{bmatrix} \lambda_1 y_1 & \dots & \lambda_n y_n \end{bmatrix} X X^T \begin{bmatrix} \lambda_1 y_1 \\ \vdots \\ \lambda_n y_n \end{bmatrix}$$

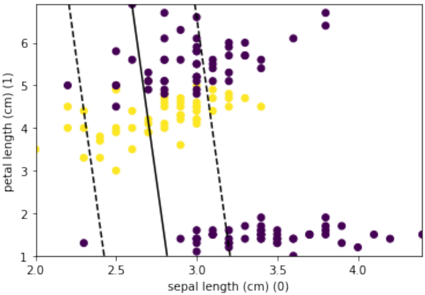
  Data only shows up through dot products (XX^T)

  Crux of Kernel approach to nonlinearity

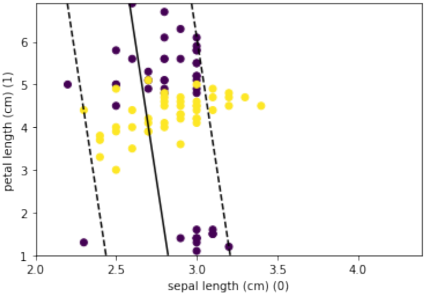
Visualization



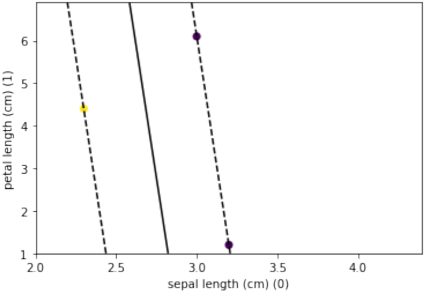
Classification



Support vectors



Support vectors



But how?

Only things that matter:
dot products of test/examples
dot products between examples

But how?

Only things that matter:
dot products of test/examples
dot products between examples

Replace $\mathbf{x}_i^T \mathbf{x}_j$ by a nonlinear function $k(\mathbf{x}_i, \mathbf{x}_j)$

not any nonlinear function, must be chosen properly

if chosen properly k reflects dot product in lifted space

Generalization: informal

Informally, quantify quality of a SVC by proportion of support vectors

Cross validation

Generalization: informal

Informally, quantify quality of a SVC by proportion of support vectors
Cross validation

Training sample n, k support vectors
Leave one out cross validation

Generalization: informal

Informally, quantify quality of a SVC by proportion of support vectors
Cross validation

Training sample n, k support vectors
Leave one out cross validation
Train on $n-1$, validate on remaining

Generalization: informal

Informally, quantify quality of a SVC by proportion of support vectors
Cross validation

Training sample n, k support vectors
Leave one out cross validation

    Train on $n - 1$, validate on remaining
    If training set has all k support vectors, no error on test

Generalization: informal

Informally, quantify quality of a SVC by proportion of support vectors

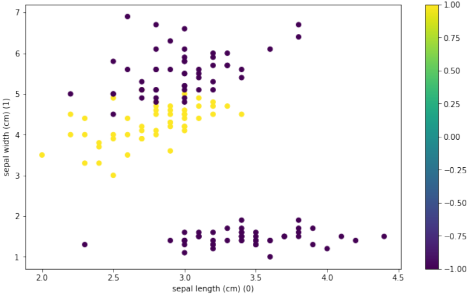
Cross validation

Training sample n, k support vectors

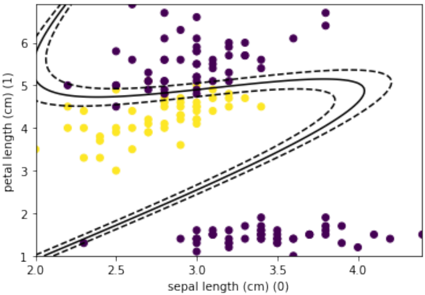
Leave one out cross validation

- Train on $n-1$, validate on remaining
- If training set has all k support vectors, no error on test
- Average over all train/validate sets
- Estimated generalization error: k/n

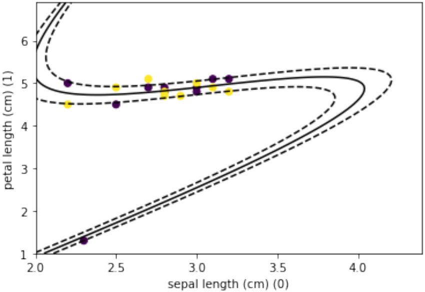
Linear to nonlinear



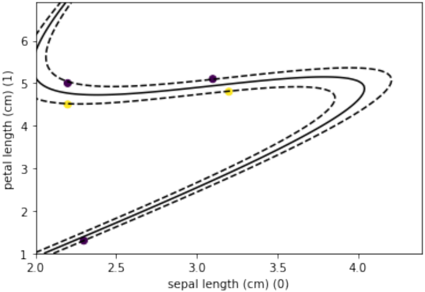
Classification: nonlinear



Support vectors: nonlinear



Support vectors: nonlinear



Introducing nonlinearities

Recall: for prediction on test $\mathbf{z}$

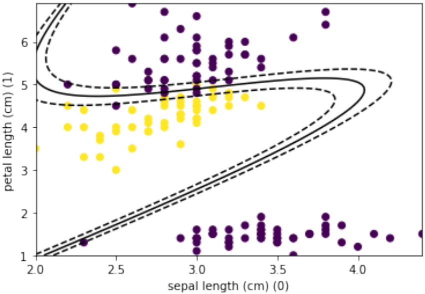
only need $\mathbf{x}_i^T \mathbf{x}_j$, for every pair of training examples

quadratic complexity in training set!

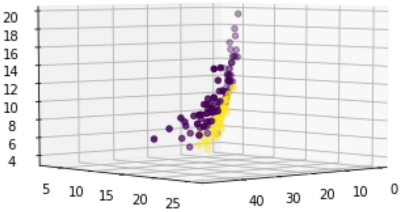
$\mathbf{z}^T \mathbf{x}_i$ for every training example

Key idea: Replace $\mathbf{x}_i^T \mathbf{x}_j$ with a *kernel function* $k(\mathbf{x}_i, \mathbf{x}_j)$

Understanding kernel functions



Understanding kernel functions



Lifting training points

Lift $\mathbf{x} \to \phi(\mathbf{x})$
In the example above:
$(x_1, x_2) \to (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$

Lifting training points

Lift $\mathbf{x} \to \phi(\mathbf{x})$
In the example above:
$(x_1, x_2) \to (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$

Linear classifier in the higher dimension
Since transformation nonlinear

linear boundaries in higher dimension look non-linear

Higher dimension space

Doesn't computation scale with the dimension of higher space?

Kernels to the rescue:

No, $\phi(\mathbf{x})^T \phi(\mathbf{y}) = k(\mathbf{x}, \mathbf{y})$

Dimension of $\phi(\mathbf{x})$ doesn't matter!

What does the higher dimensional space look like?

Reproducing Kernel Hilbert spaces

What does the higher dimensional space look like?

Reproducing Kernel Hilbert spaces
Abstract spaces that generalize Euclidean spaces

What does the higher dimensional space look like?

Reproducing Kernel Hilbert spaces
Abstract spaces that generalize Euclidean spaces
each point a function (not necessarily finite dimensional)

What does the higher dimensional space look like?

Reproducing Kernel Hilbert spaces
Abstract spaces that generalize Euclidean spaces
each point a function (not necessarily finite dimensional)
Dot product, but may look very different from vectors

What does the higher dimensional space look like?

Reproducing Kernel Hilbert spaces
Abstract spaces that generalize Euclidean spaces
each point a function (not necessarily finite dimensional)
Dot product, but may look very different from vectors
Each “function” has a length derived from the dot product

What does the higher dimensional space look like?

Reproducing Kernel Hilbert spaces
Abstract spaces that generalize Euclidean spaces
each point a function (not necessarily finite dimensional)
Dot product, but may look very different from vectors
Each "function" has a length derived from the dot product
"Smooth spaces"

What does the higher dimensional space look like?

Reproducing Kernel Hilbert spaces
Abstract spaces that generalize Euclidean spaces
each point a function (not necessarily finite dimensional)
Dot product, but may look very different from vectors
Each “function” has a length derived from the dot product
“Smooth spaces”
Shorter “length” $\leftrightarrow$ smoother function

Non linear SVC in RKHS terms

Map $\mathbf{x} \rightarrow f_{\mathbf{x}}$, where $f_{\mathbf{x}}$ is a point in a RKHS

Prediction with $f_{\mathbf{w}}$ on test point $f_{\mathbf{z}}$: $\langle f_{\mathbf{w}}, f_{\mathbf{z}} \rangle - b$

Find the function that minimizes the hinge loss
subject to a smoothness constraint ($||f_{\mathbf{w}}|| < T$)

UNIVERSITY of HAWAI'I MĀNOA

Ridge regression

Host of other problems that lend themselves to kernel methods

LLS: Given training X and target $\mathbf{y}$, find

$$\arg\min_{\mathbf{w}} ||\mathbf{y} - X\mathbf{w}||^2$$

Optimal solution is

$$\hat{\mathbf{w}} = (X^T X)^{-1} X^T \mathbf{y}$$

Ridge regression

Host of other problems that lend themselves to kernel methods

Regularized LS: Given training X and target $\mathbf{y}$, find

$$\arg\min_{\mathbf{w}} \left(||\mathbf{y} - X\mathbf{w}||^2 + \lambda||\mathbf{w}||^2\right)$$

Optimal solution is

$$\hat{\mathbf{w}} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$$

Ridge regression

Regularized LS: Given training X and target $\mathbf{y}$, find

$$\arg\min_{\mathbf{w}} \left(||\mathbf{y} - X\mathbf{w}||^2 + \lambda ||\mathbf{w}||^2 \right)$$

Optimal solution is

$$\hat{\mathbf{w}} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$$

Prediction is

$$\mathbf{z}^T \hat{\mathbf{w}} = \mathbf{z}^T (X^T X + \lambda I)^{-1} X^T \mathbf{y} = \mathbf{z}^T X^T (X X^T + \lambda I)^{-1} \mathbf{y}$$

Once again

$\mathbf{z}^T X^T$: dot product of $\mathbf{z}$ with training examples

$X X^T$ pairwise dot products between examples

Replace dot products $\mathbf{x}_i^T \mathbf{x}_j$ with a kernel $k(\mathbf{x}_i, \mathbf{x}_j)$

Kernel ridge regression

Replace dot products $\mathbf{x}_i^T\mathbf{x}_j$ with $k(\mathbf{x}_i, \mathbf{x}_j)$

Representer Theorem still holds in the abstract RKHS:
$\phi(\mathbf{w}) = \sum_i \lambda_i y_i \phi(\mathbf{x}_i)$

Prediction is

$$k(\mathbf{z}, \hat{\mathbf{w}} = k(\mathbf{z}, X), (k(X, X) + \lambda I)^{-1}\mathbf{y}$$

where $k(\mathbf{z}, X) = \begin{bmatrix} k(\mathbf{z}, \mathbf{x}_1) & \dots & k(\mathbf{z}, \mathbf{x}_n) \end{bmatrix}$
and

$$k(X, X) = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

Kernels?

Popular kernels:

Polynomial: $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^d$

Hyperparameters c, d

RKHS$(d, c) \subset$ RKHS$(d+1, c')$ (for appropriate c, c')

Radial Basis function:

Hyperparameter s (scale factor)

$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2s}\right)$

RKHS$(s) \subset$ RKHS(s') if $s' < s$

Rule of thumb: kernel with rich enough RKHS

specific kernels can incorporate structure (eg. periodicity)

Kernels

More on kernels, deeper insights into non-linear features in a separate optional video

When is a bivariate function $k(\mathbf{x}, \mathbf{x}')$ a valid kernel?
It must be positive semi-definite: namely for any n and any $\mathbf{x}_1, \ldots, \mathbf{x}_n$,

$$k(X, X) = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & \ldots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

must be positive semi-definite

Positive semi-definite

An $n \times n$ A is positive semi-definite (definite) if for all $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{w}^T A \mathbf{w} \geq 0$. ($\mathbf{w}^T A \mathbf{w} > 0$)

Equivalent: All eigenvalues of A must be ≥ 0 (> 0)

Entries need to be all-positive, all-positive matrices are not positive definite

$\begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$ is positive definite

$\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$ is not.

Credit risk monitoring: case study

Fair-Isaac Corporation (FICO) models credit risk

- FICO score you are familiar with

Credit risk monitoring: case study

Fair-Isaac Corporation (FICO) models credit risk

- FICO score you are familiar with

Start linear, modify to make non-linear

Original/base methods

- Fisher Discriminant
- Logistic Regression

Credit risk monitoring: case study

Fair-Isaac Corporation (FICO) models credit risk

- FICO score you are familiar with

Start linear, modify to make non-linear

Original/base methods

- Fisher Discriminant
- Logistic Regression

Note: both are lienar methods

Not an accident: need explainability

Fisher discriminant for credit risk

Generally FD works when each class can be normally distributed

Perhaps not a bad assumption in this case

FD is a linear method

Manually make up non-linear functions of features

Done with lot of background on econometrics models

Huge part of effort into feature engineering

Logistic Regression for credit risk

Another linear approach, but with different results
Usually works well when we have the correct features

Maximum entropy: given the features observed, find generative model for each class that makes no assumptions other than that the model matches the observed moments of features

Not even implicit assumptions are made

Again, feature engineering is the key to success, and built on insights and analysis into credit risk

Support vector approach

Instead of feature engineering, look at a rich abstract space of features (RKHS obtained via a kernel)

Domain knowledge is not assumed

  but you will see how to augment results with it

We will work with credit risk data from Kaggle

  link in discord

Miniproject

Build a model to predict risk using

Base SVM approach (choose kernel)

Use different margins for positive/negatives

Bayesian interpretations (paper provided)

Find out the econometric features used

Logistic Regression using the above features

Compare the two

We will work together (including me!)