Machine Learning in Finance Midterm Review

Winnie & Xiaoyang

Disclaimer

The best way to review is to understand the lecture slides and homework.

There will be questions on definitions, significance, coding of topics covered.

We chose to highlight some important topics here but

This is NOT an exhaustive list of topics.

Regularization

- the penalty on a model's complexity
  - LASSO regression (L1 penalty) penalizes | *weight* | -> can drive weight to 0

Objective: $minimize\ L = L_{plain} + \lambda \left|\left|w_i\right|\right|_1 = L_{plain} + \lambda \sum_i |w_i|$

Derivative: $\frac{\partial L}{\partial w_i} = \frac{\partial L_{plain}}{\partial w_i} + \lambda * sign(w_i)$, where $sign(w_i) = \begin{cases} 1, & if\ w_i > 0 \\ -1, & if\ w_i < 0 \\ undefined, & if\ w_i = 0 \end{cases}$

Update: $w_i^{t+1} \rightarrow w_i^t - \eta \left(\frac{\partial L_{plain}}{\partial w_i}\right) - \eta\lambda * sign(w_i^t)$

  - Ridge regression (L2 penalty) penalizes *weight^2 -> close to 0, always improves generalization in linear models*

Objective: $minimize\ L = L_{plain} + \lambda \left|\left|w\right|\right|_2^2 = L_{plain} + \lambda \sum_i w_i^2$

Derivative: $\frac{\partial L}{\partial w_i} = \frac{\partial L_{plan}}{\partial w_i} + 2\lambda * w_i$

Update: $w_i^{t+1} \rightarrow w_i^t - \eta \left(\frac{\partial L_{plain}}{\partial w_i}\right) - 2\eta\lambda w_i$

L norm

- L0 norm = total number of non-zero vector elements
- L1 norm = Manhattan taxicab distance = sum of the magnitudes of vectors in a space

  Eg. $\left|\left|x\right|\right|_1 = |3| + |4| = 7$

- L2 norm = Euclidean norm = sum of shortest distances

  Eg. $\left|\left|x\right|\right|_2 = \sqrt{3^2 + 4^2} = 5$

- Linfinity norm = the largest magnitude among each element of a vector

  Eg. $x = [-6\ 1\ 3]$

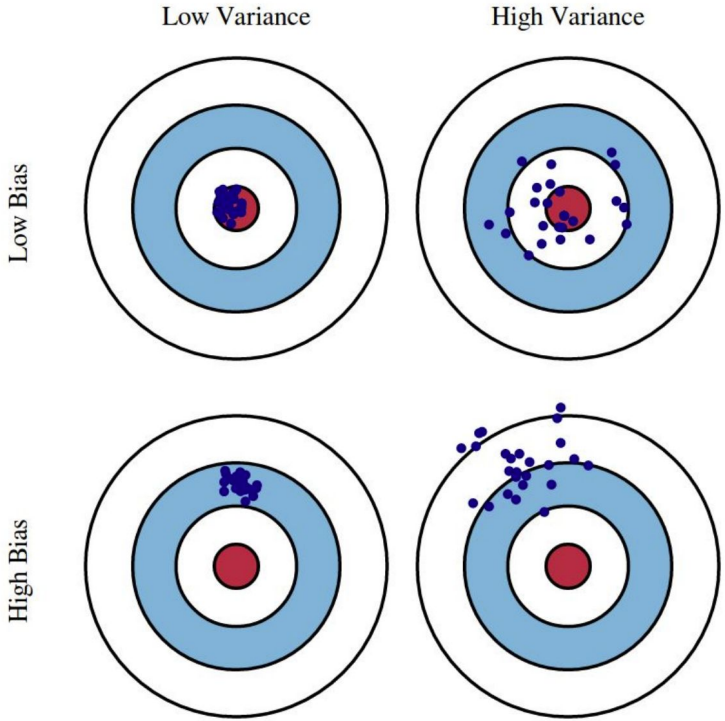
  $\left|\left|x\right|\right|_\infty = 6$

Bias-variance trade-off

- Bias – the difference between predicted value and correct value
  - High bias gives a large error in training as well as testing data
- Variance – the variability of model prediction for a given data point (for multiple builds of the model) which tells us spread of our data
  - High variance has a very complex fit to the training data and thus is not able to fit accurately on the data which it hasn’t seen before
- Irreducible (error) – cannot be reduced regardless of what algorithm is used.
  - It is the error introduced from the chosen framing of the problem and may be caused by factors like unknown variables that influence the mapping of the input variables to the output variable
- If the algorithm is too simple (few parameters) then it may be on high bias and low variance condition and thus is error-prone (underfitting)
- If algorithms fit too complex (too many parameters) then it may be on high variance and low bias (overfitting)

$$Err(x) = \left(E[\hat{f}\,(x)] - f(x)\right)^2 + E\left[\left(\hat{f}\,(x) - E[\hat{f}\,(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$



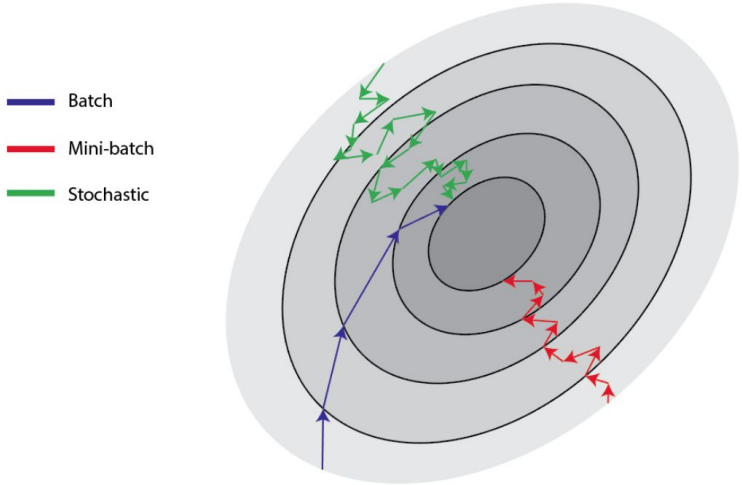
Encode

- What numerical data need to be encoded?
  - Integers
  - Numerical data that not intended to be multiplied by weights, e.g: zip code
- Label encoding: Encode target labels with value between 0 and n_classes-1.
- Hot encoding: machine-readable binary values for each feature
  - One-hot: A binary vector for all strings in the set, with one element set to 1, all others set to 0
  - Multi-hot: A binary vector where multiple elements can be set to 1

Other concepts

- Curse of dimensionality (CoD)
  - When the dimensionality increases, the volume of the space increases so fast that the available data become sparse
  - In order to obtain a reliable result, the amount of data needed often grows exponentially with the dimensionality
- Optimal learning rate
  - In one dimension, it's $\frac{1}{f''(x)}$
  - For two or more dimensions, it's the **inverse** of the Hessian matrix $(\mathbf{H}_f)_{i,j} = \frac{\partial^2 f}{\partial x_i \, \partial x_j}.$
- Gradient descent
  - Minimize loss by computing the gradients of loss with respect to the model's parameter (e.g. weight)
  - Steps of gradient descent
  - Batch sizes of the gradient

- Batch size of the gradient (cont'd)
  - Stochastic gradient descent: uses only one random sample from the training set in one step.
  - Mini-batch gradient descent: uses a predefined number of samples from the training set in one step (most common).
  - Full-batch gradient descent: uses all samples from the training set in one step.
  - The batch size affects overall training time, training time per step, quality of the model, etc. Usually, we chose the batch size as a power of two, in the range between 16 and 512. But generally, the size of 32 is a rule of thumb and a good initial choice.



Metrics (can calculate / identify)

- For binary classification:

Metric	Formula	Interpretation
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Overall performance of model
Precision	$\frac{TP}{TP + FP}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{TP}{TP + FN}$	Coverage of actual positive sample
Specificity	$\frac{TN}{TN + FP}$	Coverage of actual negative sample
F1 score	$\frac{2TP}{2TP + FP + FN}$	Hybrid metric useful for unbalanced classes

- ROC and AUC (in lecture slides)

Classification Methods

Logistic Regression - HW1,2

Decision Trees, RF - HW 2

KNN

SVM

LDA

K-Nearest Neighbor (KNN)

Can be used for both classification and regression

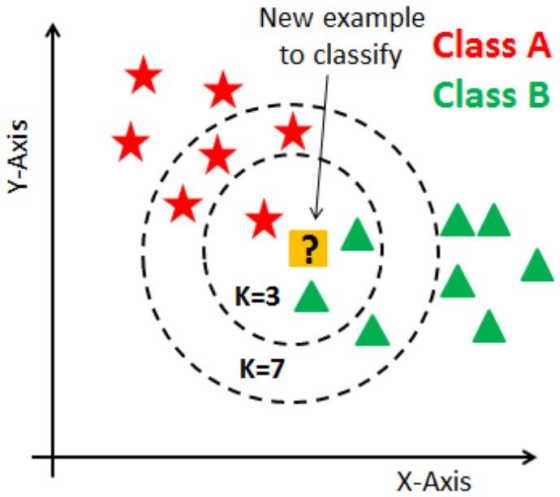
Any data point is characterized by its K nearest neighbors

In classification, take majority vote

usually have K odd for binary classification

In regression, take the mean

Example



Define Distance in Higher Dim

Manhattan - L1

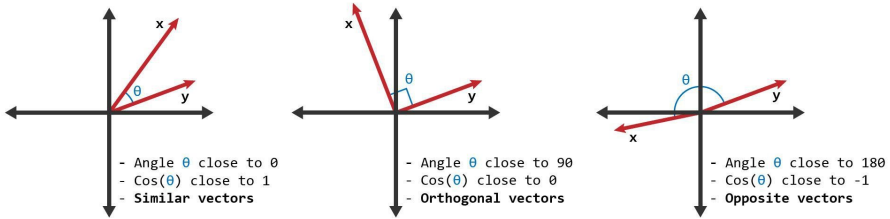
Euclidean - L2

Chebyshev - L inf

Minkowski - Lp, p positive int

Cosine

Cosine Distance=1-Cosine Similarity

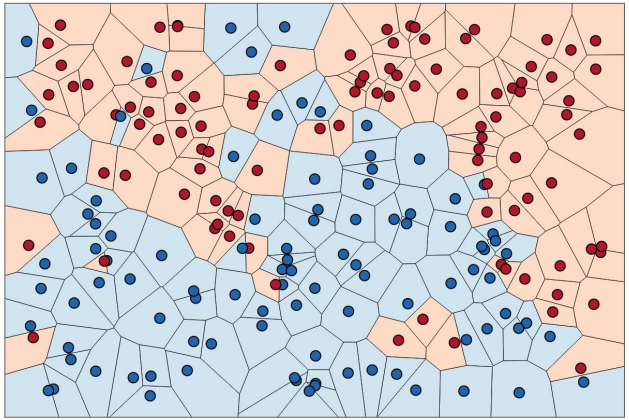


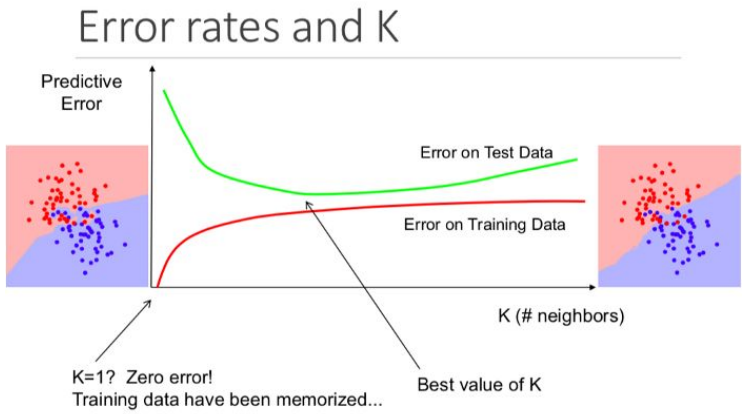
Choosing K in KNN

K=1, no training error

Large k will smooth the boundary, but not too large, K=N?

Hyperparameter tuning to find the best K for different dataset





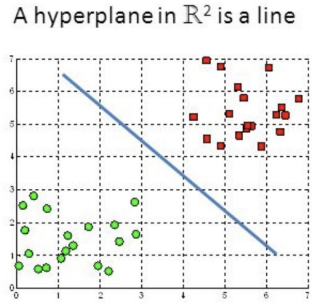
Support Vector Machines (SVM)

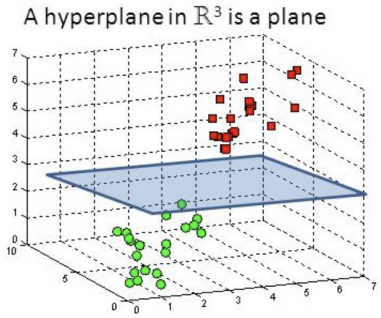
Binary linear classifier

Find a gap between the two categories

In 2 dimension, we need a line

In higher dimension, we need a hyperplane





The optimal hyperplane should maximize the distance between it and the closest data points.

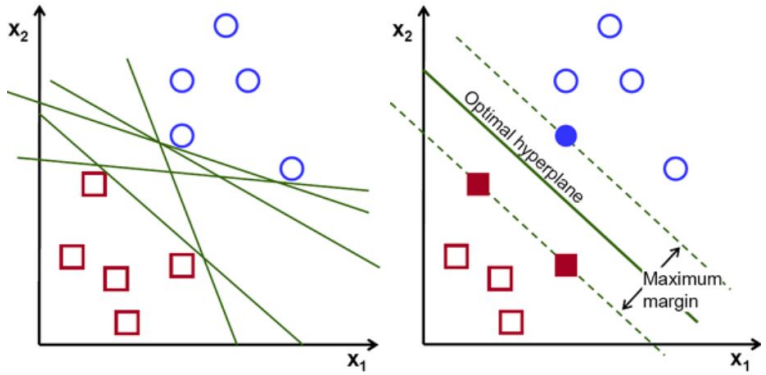
Such distance is called the margin

Such data points are called the support vectors, hence the name.

Hence SVM is a max margin classifier

Support Vectors

Support vectors ‘support’ the hyperplane



SVM loss

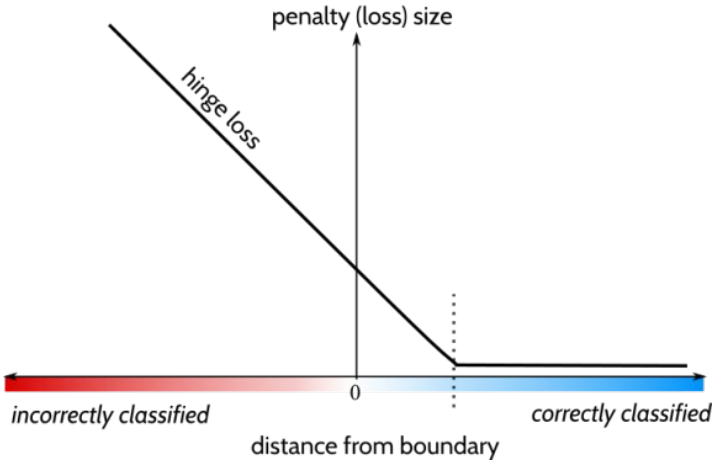
Hinge Loss

Loss(y)= max(0,1-t · y)

t=1 or -1 Intended output

y=wx+b classifier score

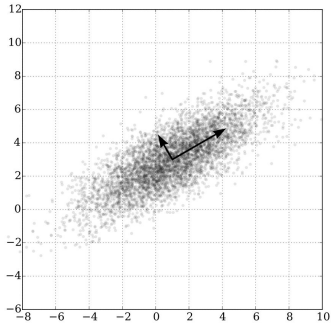
t y different signs give a bigger loss



Intuition:

If you're correctly classified beyond the margin (ty>1) then it's irrelevant just how far on the correct side you are. On the other hand, if you're within the margin or on the incorrect side, you get a penalty directly proportional to how far you are on the wrong side.

Linear Discriminant Analysis



Recall from linear algebra: matrices are linear transformations/maps

The idea of PCA:

Transform the data such that the greatest variance comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on

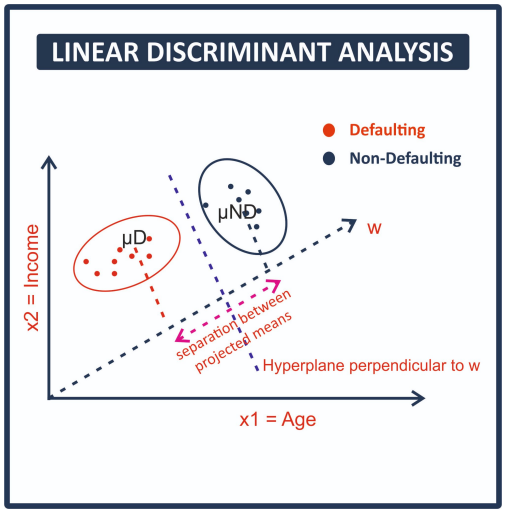
The idea of LDA:

Like a Supervised version of PCA:

Transform the data such that

Maximize separation between classes

Minimize variance within each class



Dimension Reduction

It is equivalent to maximizing the ratio of the between-class scattering to the within-class scattering

$$\max_{\mathbf{a}} \frac{\mathbf{a}^T \mathbf{B} \mathbf{a}}{\mathbf{a}^T \mathbf{W} \mathbf{a}} . \qquad (7)$$

To solve this,
We need to do eigendecomposition, then we can
Project data to first k dim by ranking the values of eigenvalues

Questions?