(1) what the last equation shows is that the squared reconstruction error is given by the (sum) of eigenvalues of the unused eigenvectors.

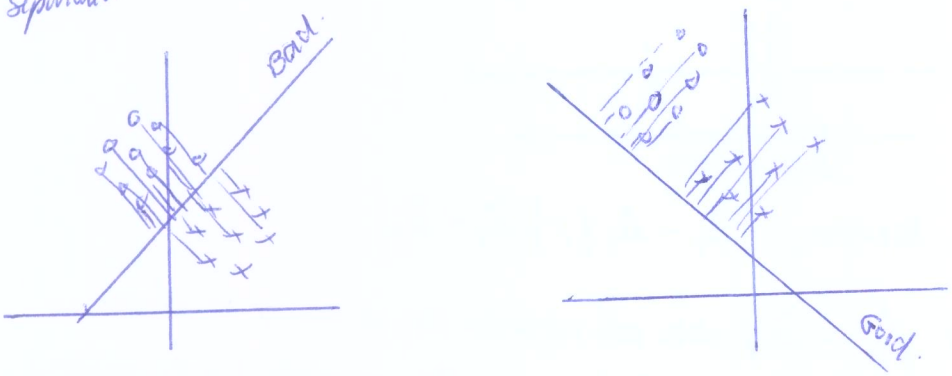
Data Representation (vs) Data classification

PCA ⇒ best data representation in a lower dimensional space.
Projects data in the directions of most/maximum variance.

~ Directions of max variance are not useful for classification.

~ LDA or Linear Discriminant Analysis or Fisher Discriminant Analysis project to a line which preserves direction useful for data classification.

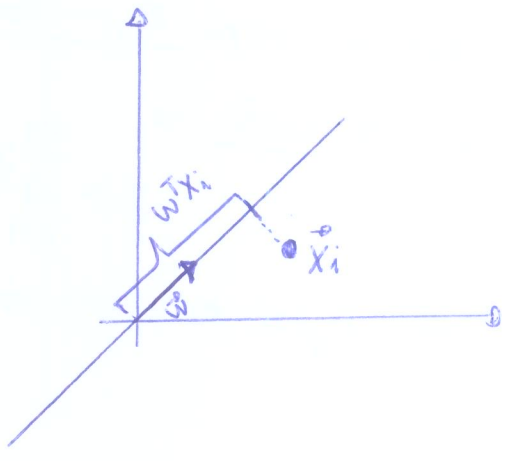
LDA find a projection to a line ~~subject~~ such that samples from different classes are well separated



LDA for (2) classes: Suppose we have (2) classes in (n)-dimensions $x_i \in \mathbb{R}^n$.

m_1 = # examples in class 1
m_2 = # examples in class 2.

~ Consider projection on a line.
~ Let the line direction be given by unit vector $\vec{w}$.
Scalar $\vec{w}^T\vec{x}_i$ is the distance of projection of $\vec{x}_i$ from the origin. Thus $\vec{w}^T\vec{x}_i$ is the projection of $\vec{x}_i$ onto a 1-d subspace



① Thus the projection of x_i onto a line in direction $\vec{w}$ is given by $\vec{w}^T \vec{x}_i$.

② How to measure the separation between projections of different classes?

③ Let $\tilde{\mu}_1$ and $\tilde{\mu}_2$ be the means of projections of classes ① and ②.

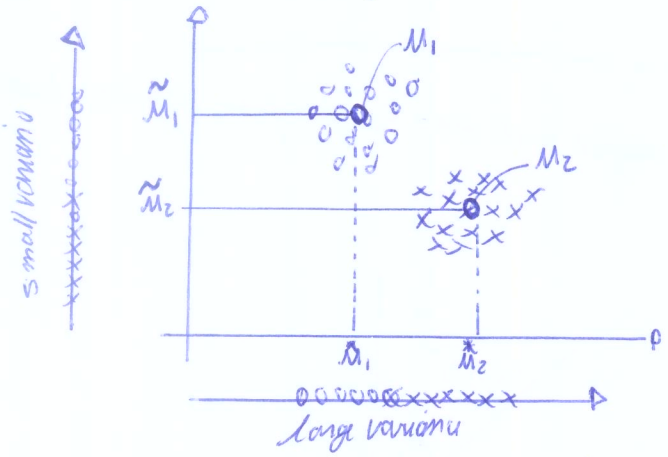
④ Let μ_1 and μ_2 be the means of classes ① and ②.

⑤ $J = |\tilde{\mu}_1 - \tilde{\mu}_2|$ seems like a good measure.

$$\tilde{\mu}_1 = \frac{1}{M_1} \sum_{x_i \in C_1}^{M_1} \vec{w}^T \vec{x}_i = \vec{w}^T \left(\frac{1}{M_1} \cdot \sum_{x_i \in C_1}^{M_1} \vec{x}_i \right) = \vec{w}^T \mu_1$$ where C_1 is class 1

Similarly $\tilde{\mu}_2 = \vec{w}^T \vec{\mu}_2$

⑥ How good is $J = |\tilde{\mu}_1 - \tilde{\mu}_2|$ for separation?



* Vertical axis is better. However, $|\tilde{\mu}_1^* - \tilde{\mu}_2^*| > |\tilde{\mu}_1 - \tilde{\mu}_2|$

* The problem is that $|\tilde{\mu}_1 - \tilde{\mu}_2|$ does not consider the variance of the classes.

* Therfore we need to normalize $|\tilde{\mu}_1 - \tilde{\mu}_2|$ by a factor proportional to the variance.

* For that, have samples $z_1, \dots z_n$ Sample mean is $\mu_2 = \frac{1}{n_2} \cdot \sum_{i=1}^{n_2} z_i$.

* Define the scatter as

$$S = \sum_{i=1}^{n_2} (z_i - \mu_2)^2$$

scatter is just the variance multiplied by n_2.

Fisher solution: normalize $|\tilde{\mu}_1 - \tilde{\mu}_2|$ by scatter.

Let $y_i = \vec{w}^T \vec{x}_i$ i.e., y_is are the projected samples

Scatter of projected samples of class ①

$$\tilde{S}_1^2 = \sum_{y_i \in \text{class } 1} (y_i - \tilde{\mu}_1)^2.$$

class 2

$$\tilde{S}_2^2 = \sum_{y_i \in \text{class } 2} (y_i - \tilde{\mu}_2)^2.$$

The fisher linear discriminant is then to project on line in the direction of $\vec{w}$ that maximizes

$$J(\vec{w}) = \frac{\boxed{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}}{\boxed{\tilde{S}_1^2} + \boxed{\tilde{S}_2^2}}$$

want projected means far from each other.

want scatter of class ① small

want scatter of class 2 small.

If we find $\vec{w}$ that makes $J(\vec{w})$ large we are in business!

All we need to do now is to express ⓙ as a function of $\vec{w}$ and maximize it

For that, ① define the separate class scatter matrices S_1 and S_2 for classes ① and ②. These measure the scatter of the original samples x_i (before projection)

$$S_1 = \sum_{x_i \in \text{class } 1} (\vec{x}_i - \mu_1) \cdot (\vec{x}_i - \mu_1)^T.$$

$$S_2 = \sum_{x_i \in \text{class } 2} (x_i - \mu_2) \cdot (x_i - \mu_2)^T.$$

② Now define the within-class scatter matrix

$$S_w = S_1 + S_2.$$

③ Now recall $\tilde{S}_1^2 = \sum_{y_i \in \text{class } 1} (y_i - \tilde{\mu}_1)^2.$

Using $y_i = w^T x_i$ and $\tilde{\mu}_1 = w^T \mu_1$.

$$\tilde{s}_1^2 = \sum_{y_i \in \text{class } 1} (w^T x_i - w^T \mu_1)^2.$$

$$= \sum_{y_i \in \text{class } 1} (w^T (x_i - \mu_1))^2$$

$$= \sum_{y_i \in \text{class } 1} (w^T (x_i - \mu_1))^T \cdot (w^T (x_i - \mu_1))$$

$$= \sum_{y_i \in \text{class } 1} ((x_i - \mu_1)^T \cdot w)^T \cdot ((x_i - \mu_1)^T \cdot w)$$

$$= \sum_{y_i \in \text{class } 1} w^T \cdot (x_i - \mu_1) \cdot (x_i - \mu_1)^T \cdot w = \boxed{w^T \cdot S_1 \cdot w}$$

Similarly, $\boxed{\tilde{s}_2^2 = w^T \cdot S_2 \, w}$.

Therefore $\tilde{s}_1^2 + \tilde{s}_2^2 = w^T S_1 w + w^T S_2 w = \boxed{w^T S_w w}$

④ Now we define the between class scatter matrix

$$S_B = (\mu_1 - \mu_2) \cdot (\mu_1 - \mu_2)^T.$$

S_B measures separation between the means of two classes (before projection)

⑤ Let's now re-write the separations of the projected means

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (w^T \mu_1 - w^T \cdot \mu_2)^2.$$

$$= w^T (\mu_1 - \mu_2) \cdot (\mu_1 - \mu_2)^T \cdot w.$$

$$= \boxed{w^T \cdot S_B \cdot w}$$

Our objective can then be written as

$$J(\vec{w}) = \frac{\left(\tilde{\mu}_1 - \tilde{\mu}_2\right)^2}{\tilde{s}_1^2 + \tilde{s}_2^2} = \boxed{\frac{\vec{w}^T S_B \vec{w}}{\vec{w}^T . S_W \vec{w}}}$$

$J(w)$ now needs to be maximized. For that, we derive and equate to zero.

$$\frac{\partial J(w)}{\partial w} = \frac{\partial}{\partial w}\left[\frac{w^T S_B w}{w^T S_W w}\right] = 0 \Rightarrow$$

$$\left[w^T . S_W w\right] \frac{\partial\left[w^T . S_B w\right]}{\partial w} - \left[w^T . S_B w\right] . \frac{\partial\left[w^T . S_W w\right]}{\partial w} = 0 .$$

$$\left[w^T . S_W w\right] . 2 S_B w - \left[w^T . S_B w\right] 2 S_W w = 0 .$$

① Dividing by $\boxed{w^T S_W w}$

$$\left[\frac{w^T S_W w}{w^T S_W w}\right] S_B w - \left[\frac{w^T S_B w}{w^T S_W w}\right] . S_W w = 0$$

$$S_B w - J S_W w = 0 .$$

$$S_W^{-1} . S_B w - J w = 0 .$$

$$\boxed{S_W^{-1} . S_B w = J w}$$

generalized eigenvalue problem.

Solving the generalized eigenvalue problem, yields

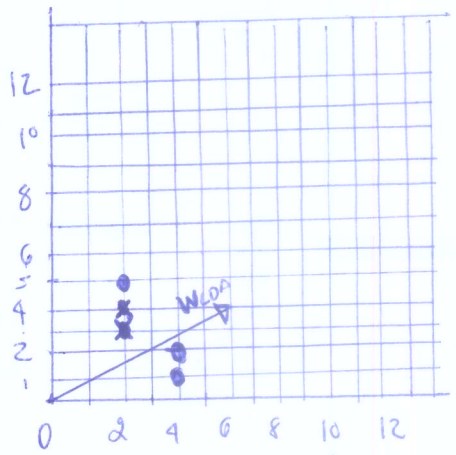
$$w^* = \arg\max\left[\frac{w^T S_B w}{w^T . S_W w}\right] = \boxed{S_W^{-1} . \left(\mu_1 - \mu_2\right)}$$

fisher's linear discriminant.

Example

$X_1 = \{(4,1), (2,4), (2,3), (3,6), (4,4)\}$.

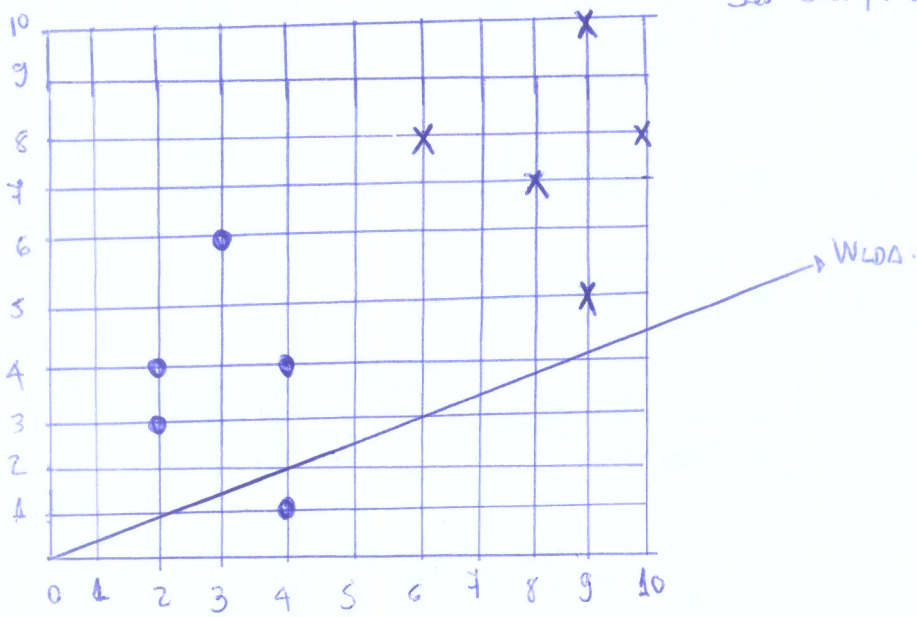
$X_2 = \{(9,10), (6,8), (9,5), (8,7), (10,8)\}$.



See example bellow.

Solution by hand

$$S_1 = \begin{bmatrix} .8 & -0.4 \\ \not{} & 2.64 \end{bmatrix} \qquad S_2 = \begin{bmatrix} 1.84 & -0.04 \\ \not{} & 2.64 \end{bmatrix}.$$



$$\mu_1 = \begin{bmatrix} 3.0 \\ 3.6 \end{bmatrix} \qquad \mu_2 = \begin{bmatrix} 8.4 \\ 7.6 \end{bmatrix}.$$

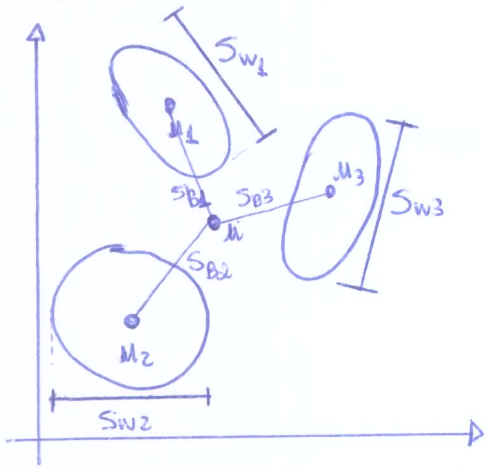
$$S_B = \begin{bmatrix} 29.16 & 21.6 \\ \not{} & 16.0 \end{bmatrix} \qquad S_W = \begin{bmatrix} 2.64 & -0.44 \\ / & 5.28 \end{bmatrix}$$

The (LDA) projection is then obtained as the solution of the generalized eigenvalue problem.

$$S_W^{-1} S_B w = \lambda w \Rightarrow w^* = S_W^{-1} \cdot (\mu_1 - \mu_2) = \begin{bmatrix} -0.91 & -0.39 \end{bmatrix}^T.$$

what about more classes?

LDA



~ Fisher's linear discriminant generalizes gracefully for C-class problems
Instead of one projection (Y), we will now seek (C-1) projections $[y_1, y_2, \cdots y_{C-1}]$ by means of (C-1) projection vectors (w_i) arranged by columns into a projection matrix W where

$$W = [w_1 | w_2 | \cdots | w_{C-1}]$$

$$y_i = w_i^T x \Rightarrow y = W^T x$$

Derivation: The within-class scatter generalizes as

$$S_W = \sum_{i=1}^{C} S_i$$

$$S_i = \sum_{x \in C_i} (x - \mu_i) \cdot (x - \mu_i)^T$$

$$\mu_i = \left(\sum_{x \in C_i} x \right) \cdot \frac{1}{M_i} \Rightarrow \frac{1}{M_i} \cdot \sum_{x \in C_i} x$$

$M_i = |C_i|$ nbr elements in class (C_i)

Between-class scatter

$$S_B = \sum_{i=1}^{C} \frac{1}{M_i} (\mu_i - \mu) \cdot (\mu_i - \mu)^T$$

$$\mu = \frac{1}{M} \cdot \sum_{\forall x} x = \frac{1}{M} \cdot \sum_{i=1}^{C} M_i \cdot \mu_i$$

matrix (S_T) will be known as the total scatter

$$S_T = S_B + S_W .$$

Similarly, we define the mean vector and scatter matrices for projected samples

$$\tilde{\mu}_i = \frac{1}{M_i} \sum_{y \in C_i} y$$

$$\tilde{\mu} = \frac{1}{M} \sum_{\forall y} y$$

$$\tilde{S}_B = \sum_{i=1}^{C} M_i (\tilde{\mu}_i - \tilde{\mu}) \cdot (\tilde{\mu}_i - \tilde{\mu})^T$$

$$\tilde{S}_W = \sum_{i=1}^{C} \sum_{y \in C_i} (y - \tilde{\mu}_i) \cdot (y - \tilde{\mu}_i)^T .$$

From our derivation for the two-class problem, we can write

$$\tilde{S}_W = W^T S_W W$$

$$\tilde{S}_B = W^T S_B W .$$

Recall that we are looking for a projection that maximizes the between-class to within-class ratio.

Since the projection is not a scalar but a $(c-1)$-d vector, we use the determinant of the scatter matrices to obtain a scalar objective function

$$J(W) = \frac{|\tilde{S}_B|}{|\tilde{S}_W|} = \frac{\det(W^T S_B W)}{\det(W^T S_W W)} .$$

and we will now look for a projection matrix W^* that maximizes this ratio.

(*) The optimal projection W^* is the one whose colums are the eigenvectors corresponding to the largest eigenvalues of the generalized eigenvalue problem

$$W^* = \arg\max \frac{|W^T S_B W|}{|W^T S_W W|} = \arg\max \left(\frac{\det(W^T S_B W)}{\det(W^T S_W W)} \right) \Rightarrow S_B - \lambda_i S_W) w_i^* = 0$$

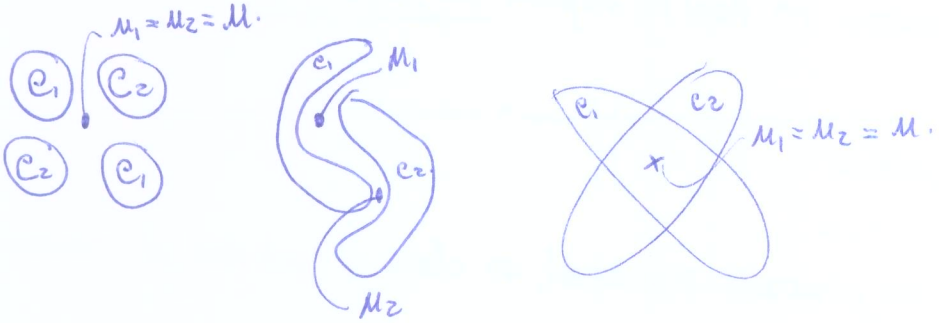
$$S_B = \lambda_i S_W w_i^* \Rightarrow \boxed{S_W^{-1} \cdot S_B = \lambda_i w^*}$$

The projections with maximum separability info are the eigenvectors corresponding to the largest eigenvalues of $|S_W^{-1} S_B|$

Limitations of LDA

(1) It produces at most $c-1$ feature projections. If the classification error estimates shows that more features are needed, other method should be used.

(2) LDA is a parametric method (it assumes unimodal Gaussian likelihoods) if the data is non-Gaussian, significantly, complex structures might not be preserved after the projection



With so many solutions seen so far, how do you choose what comes next?

Debugging a learning algorithm:

Suppose we implemented a regularized linear regression to predict grades of students

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(f_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{m}\theta_j^2\right]$$

However, when you test your hypothesis on a new set of students, you find that it makes huge errors in its predictions. What do you do?

many possibilities

(1) Get more training data. Does it always help/solve the problem? NO

(2) Try smaller sets of features $x_1, \dots, x_{100} \rightsquigarrow z_1, \dots z_k$ $k < 100$.
- feature selection
- dimensionality reduction.

objective is to avoid overfitting.

(3) Try getting additional features (new info about the problem).

(4) Try adding polynomial features (x_1^2, x_2^2, x_1x_2, etc).

(5) changing λ ↑ or ↓.

(*) Unfortunately what people do is to try at random out of all of these possibilities.

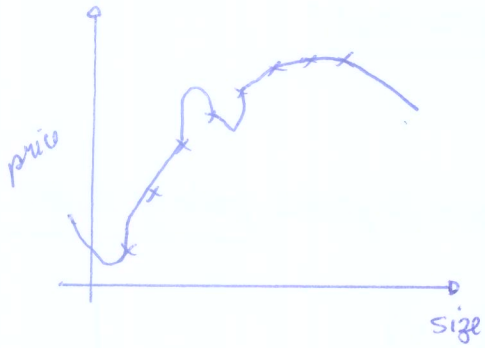
Fortunately we have some ways to rule out most options and focus on the ones that are really important to ~~edit~~ our problem. This is called machine Learning Diagnostics

machine Learning Diagnostics

A test for gaining insight about what is working/not working with a learning algorithm and gain guidance on how to improve its performance.

Evaluating a hypothesis

when we fit the parameters to a model, we choose it such that it minimizes the training error.



* fails to generalize to new examples not in training set

for this case, how do we check it?

$$f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

(1) we could plot $J(\theta)$ for 2-D problems. However most problems involve (m) dimensions.

(2) Split Training data
- 70/30
- 50/50
- ⋮
- many possibilities.

choose this @ random

now the procedure would for training/testing would be

(1) Learn parameters θ_i from training data (minimizing training error $J(\theta)$).

(2) Compute test set error:

$$J_{test}\left(\underset{\text{from training}}{\theta}\right) = \frac{1}{2M_{test}} \sum_{i=1}^{M_{test}} \underbrace{\left(f_\theta(x_{test}^{(i)}) - y_{test}^{(i)}\right)^2}_{\text{if we are using squared error and linear regression.}}$$

What if it is a classification problem ?

for instance, logistic regression:

$$J_{test}(\theta) = -\frac{1}{m_{test}} \cdot \sum_{i=1}^{m_{test}} \left(y_{test}^{(i)} \cdot \log f_\theta(x_{test}^{(i)}) + (1 - y_{test}^{(i)}) \cdot \log f_\theta(x_{test}^{(i)}) \right).$$

or, we can use a simpler form based on the misclassification error.

misclassification error (0/1 misclassification error)

$$\text{error}(f_\theta(x), y) = \begin{cases} 1 & \text{if } f_\theta(x) \geq 0.5,\ y = 0 \text{ (error)} \\ & \text{or if } f_\theta(x) < 0.5 \text{ if } y = 1 \text{ in your problem (error)} \\ 0 & \text{otherwise.} \end{cases}$$

$$\text{Test}_{\text{error}} = \frac{1}{M_{test}} \sum_{i=1}^{M_{test}} \text{error}\left(f_\theta(x_{test}^{(i)}), y_{test}^{(i)} \right).$$

fraction of mislabeled examples in the test set.

---

model selection and training/validation/testing sets

Problem:
- How to choose features to include?
- Polynomial features to add?
- Regularization parameter λ?
- This is called model selection problem

① Training set error is not a good predictor for how well an algorithm will go in practice later. Once parameters $\theta_1, \dots \theta_n$ were fit to some set of data (training set) the error of the parameters in training is likely to be much lower than the actual generalization error.

So consider the model selection problem for linear regression:

Model selection for
① $f_\theta(x) = \theta_0 + \theta_1 x$.
② $f_1(x) = \theta_0 + \theta_1 x + \theta_2 x^2$.
③ $f_2(x) = \theta_0 + \theta_1 x + \theta_3 x^3$.
⋮
⑩ $f_{10}(x) = \theta_0 + \theta_1 x + \dots + \theta_{10} x^{10}$.

The problem can be seen as finding the degree of the polynomial to use ⑦.

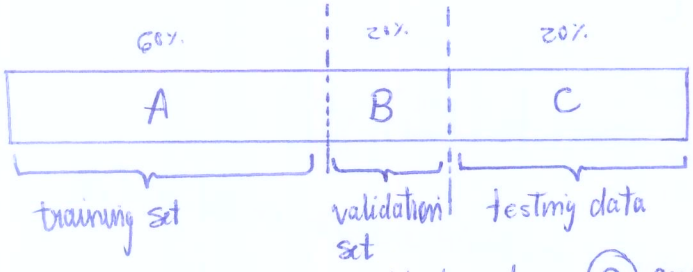
What to choose?

① You could calculate fitting and training error. The fitting would give you $\theta^{(1)}, \theta^{(2)}, \dots \theta^{(10)}$ for the different models and the test errors would give you $J_{test}^{(1)}(\theta^{(1)}), J_{test}^{(2)}(\theta^{(2)}) \dots, J_{test}^{(10)}(\theta^{(10)})$.

The selection would be the one with minimum J_{test}. So far, so good

② The problem with the aforementioned approach is that we are optimistic about the generalization error once we used the test set for taking a decision. Here the decision was the degree of the polynomial to use.

③ To solve it, we divide the data we have:



Then we fit θ on (A), select the best model based on (B) and test the best model on (C).

$$\begin{cases} J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_\theta(x^{(i)}) - y^{(i)}\right)^2 \longrightarrow \text{training error.} \\ J_{val}(\theta) = \frac{1}{2m_{val}} \sum_{i=1}^{m_{val}} f_\theta(x_{val}^{(i)}) - y_{val}^{(i)})^2 \longrightarrow \text{validation error.} \\ J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(f_\theta(x_{test}^{(i)}) - y_{test}^{(i)}\right)^2 \longrightarrow \text{test error} \end{cases}$$

The test op set (C) gives us an estimation of the generalization error.

Diagnosing Bias (vs) Variance

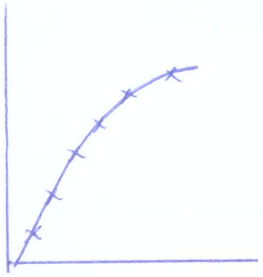
If you have problems running a learning algorithm (as it turns out, it does not do what you hoped for), you have certainly a problem of bias or high variance

bias	(vs)	variance
or		or
underfitting	(vs)	overfitting

Bias / variance

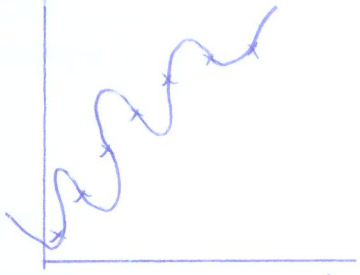
High bias (underfit)

$d = 1$ (degree).



"Just right"

$d = 2$

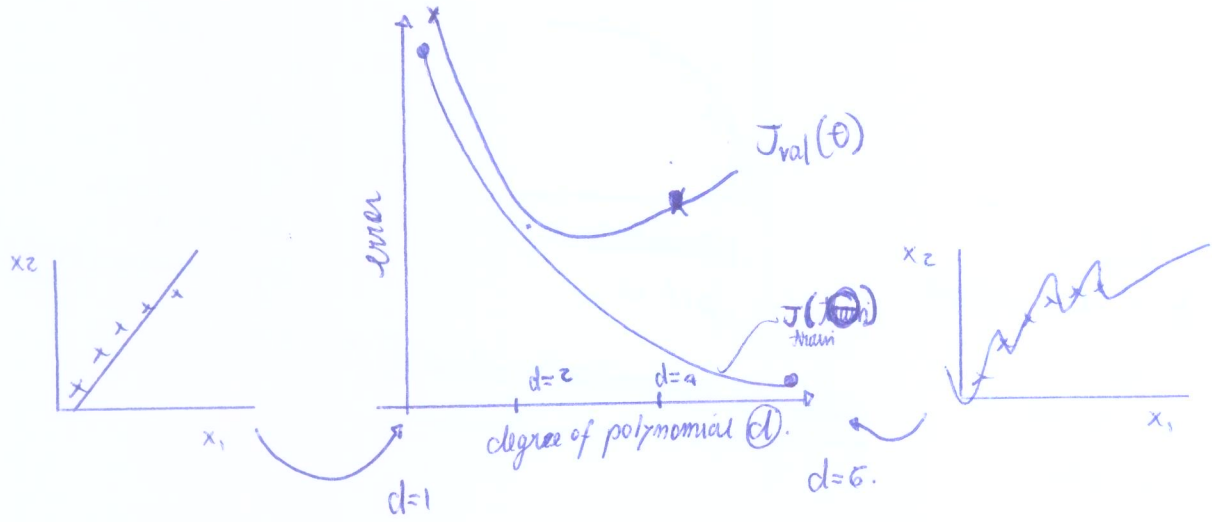


High variance (overfit)

$d = 6$.

Let's say our training error is $J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_\theta(x^{(i)}) - y^{(i)}\right)^2$

and our validation error $J_{val}(\theta) = \frac{1}{2 m_{val}} \cdot \sum_{i=1}^{m_{val}} \left(f_\theta(x_{val}^{(i)}) - y_{val}^{(i)}\right)^2$

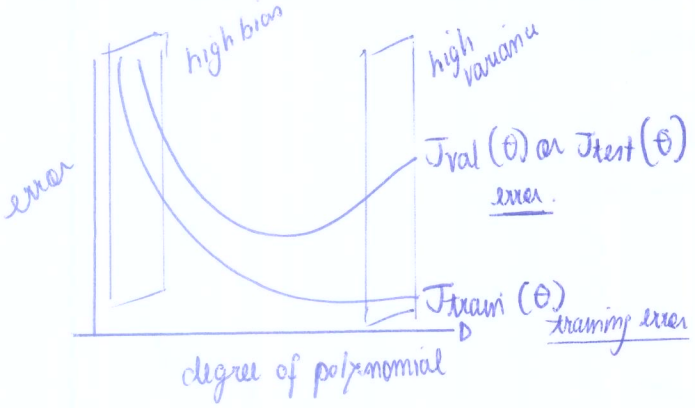


This type of plot helps us to understand the bias and variance problem.

Diagnosing Bias and Variance

Suppose we ~~had~~ have a learning algorithm not doing well. J_{val} or J_{test} is high.

Is it a bias or a variance problem?



Bias (underfit)
- $J_{train}(\theta)$ is high
- $J_{val} \approx J_{train}$

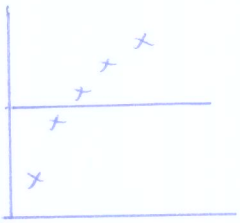
Variance (overfit)
- $J_{train}(\theta)$ is low.
- $J_{val} \gg J_{train}$.
  (much greater)

Regularization and Bias/Variance

We already know that ~~overfitting~~ regularization can help preventing overfitting but how does it affect bias and variance?

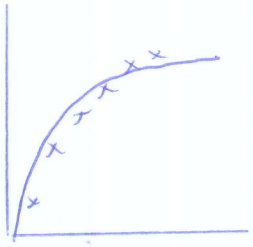
Suppose our model is $f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$.

To avoid overfitting, suppose we ~~not~~ regularize with $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_\theta(x^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{m} \theta_j^2$.

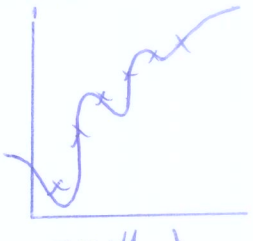


leads to large λ
high bias (underfit).

$\lambda = 1000$
$\theta_1 \approx 0 \quad \theta_2 \approx 0, \dots$
$f_\theta(x) \approx \theta_0$.



Intermediate λ
Just ok.



Small λ
High variance (overfit)
$\lambda \approx 0$.

How to automatically choose λ?

Choosing the right regularization parameter (λ)

To solve it, lets keep our cost function for training, validation and test without regularization

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_\theta(x^{(i)}) - y^{(i)}\right)^2 + \cancel{\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2}$$

eliminate just while calculating/plotting training error only (see below)

for fitting we use the normal regularized $J(\theta)$.

$$J_{val}(\theta) = \frac{1}{2m}\sum_{i=1}^{m_{val}}\left(f_\theta(x_{val}^{(i)}) - y_{val}^{(i)}\right)^2$$

$$J_{test}(\theta) = \frac{1}{2m}\sum_{i=1}^{m_{test}}\left(f_\theta(x_{test}^{(i)}) - y_{test}^{(i)}\right)^2$$

Now we define a range of (λ) values:

① $\lambda = 0 \rightsquigarrow \min_\theta J(\theta) \Rightarrow \theta^{(1)} \longrightarrow J_{val}(\theta^{(1)})$

② $\lambda = 0.01 \rightsquigarrow \min_\theta J(\theta) \Rightarrow \theta^{(2)} \longrightarrow J_{val}(\theta^{(2)})$.

③ $\lambda = 0.02$

④ $\lambda = 0.04$

⑤ $\lambda = 0.08$

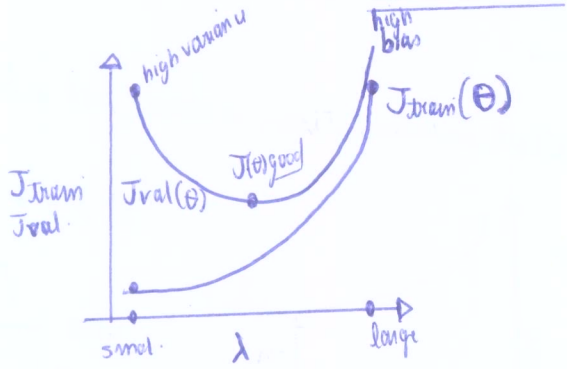
⋮

⑫ $\lambda = 10 \rightsquigarrow \min_\theta J(\theta) \Rightarrow \theta^{(12)} \longrightarrow J_{val}(\theta^{(12)})$.

multiplies (×2).

Then I choose the one with lowest J_{val}, say $\theta^{(k)}$ and calculate the test error

$J_{test}(\theta^{(k)})$.



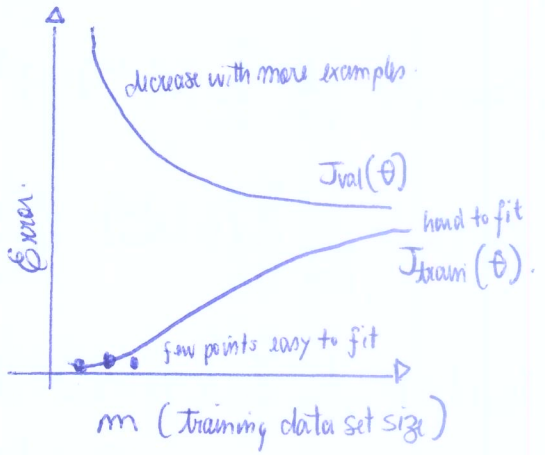
Learning Curves

Tool for ~~diagnosing~~ diagnosing the bias/variance problem.

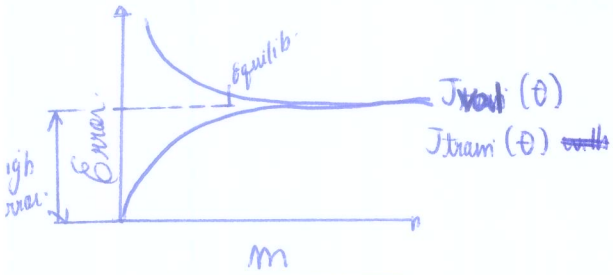
$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{val}(\theta) = \frac{1}{2m_{val}}\sum_{i=1}^{m_{val}}\left(f_\theta(x_{val}^{(i)}) - y_{val}^{(i)}\right)^2$$

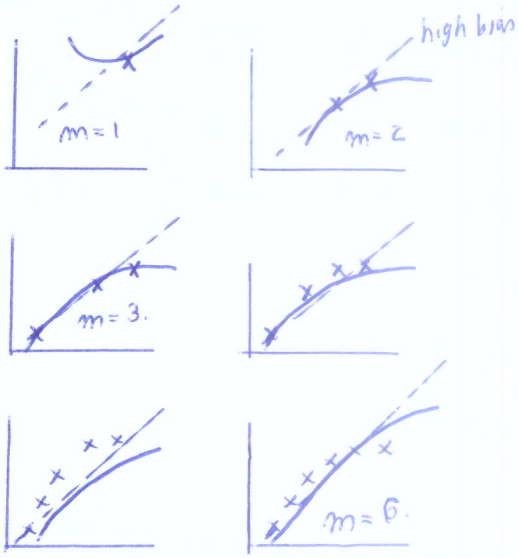
average squared error on the validation.



Let's see now a model with high bias (- - -)



$J_{val} \approx J_{train}$

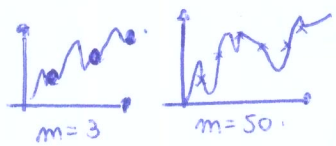


- If a learning algorithm is suffering from high bias, getting more data will not help
- If more data doesn't decrease much the validation error ⇒ bias problem.

- $f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
- measure Error only on the actual samples used.

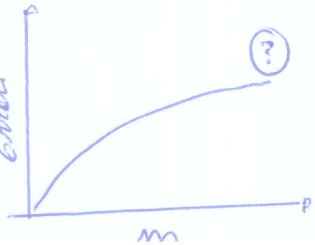
Let's see a high variance model

$$f_\theta(x) = \theta_0 + \theta_1 x \ldots + \theta_{100} x^{100}$$

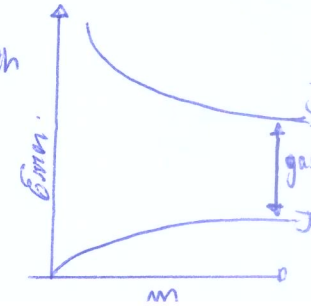


with small (λ).

If the training set size is small J_{train} is small. If data set increases, J_{train} increases just a bit.



(?) nope we actually end up with



high gap ⇒ high variance problem.

and more data will help

How does all of this help us in real life?

(*) If an algorithm didn't behave as expected we can

(1) Get more training data. —help→ fixes high variance.

(2) Select features. —help→ fixes high variance.

(3) Add features —help→ fixes high bias.

(4) Add complexity (higher order combination of features) —help→ fixes high bias.

(5) Decrease λ —help→ fixes high bias

(6) Increase λ. —help→ fixes high variance.

⋮

Neural Networks and overfitting

Small (NNs) { - fewer param
more prone to underfitting
less comp. expensive.

(VS)

Larger (NNs) { - more params
- more prone to overfitting.
- Comp. ⊕ expensive

Use regularization (λ) to address overfitting.

→ Use validation set to help choosing # number of units / hidden layers and ↗

Avaliação de modelos Preditivos - issm - Cap. 9

(1) There is no universal technique good at all kinds of problems.

(2) In some cases the problem properties can help us solve what to choose. For instance, for high dimensionality SVMs might be more appropriate than K-NN. In another case, if we need to interpret all step/choices of the algorithm, a tree classifier might be more appropriate than a neural network.

(3) In many situations we need to compare different approaches to solve a problem or the same approach with different parameters.

Forms of comparison

1. Classification accuracy.
2. Comprehensiveness level of the model. How ~~can~~ easily can we understand the created model?
3. Storage requirements
4. Model complexity.

Error metrics

$$err(\hat{f}) = \frac{1}{m}\sum_{i=1}^{m} I\left(y_i \neq \hat{f}(x^{(i)})\right)$$

(y_i: labeled class; $\hat{f}(x^{(i)})$: predicted class)

$I(a) = 1$ if a is true and 0 otherwise.

The error rate varies from [0-1]. The lower the better. Its complement is known as classifier accuracy.

$$acc(\hat{f}) = 1 - err(\hat{f}).$$

The higher the values the better.

Another way of evaluating a classifier is through a confusion matrix

confusion matrix

- Counts correct and incorrect classifications for each class.
- Rows ⇒ correct classes / columns ⇒ predicted classes

(m_{ij}) of a confusion matrix (m_c) shows the # elements of class (i) classified as members of class (j). For (k) classes, m_c is a $k \times k$ matrix.

The diagonal represents the correct classifications.

(*) Using this matrix, we can point out which classes the algorithm has more troubles / problems at classifying.

Confusion matrix Example:

True class	$\hat{A}$	$\hat{B}$	$\hat{C}$	predicted classes
A	20	2	4	$\|A\| = 26$
B	1	5	0	$\|B\| = 6$
C	2	1	7	$\|C\| = 10$

In this case, 20/26 examples of class (A) were correctly classified.

---

metrics for Regression

Recall that in the case of regression, our metric is the distance between the known (y_i) value and the one predicted by the model, $\hat{f}(x^{(i)})$. The two most ~~common~~ common forms of evaluation for regression are the mean squared error and the mean absolute distance.

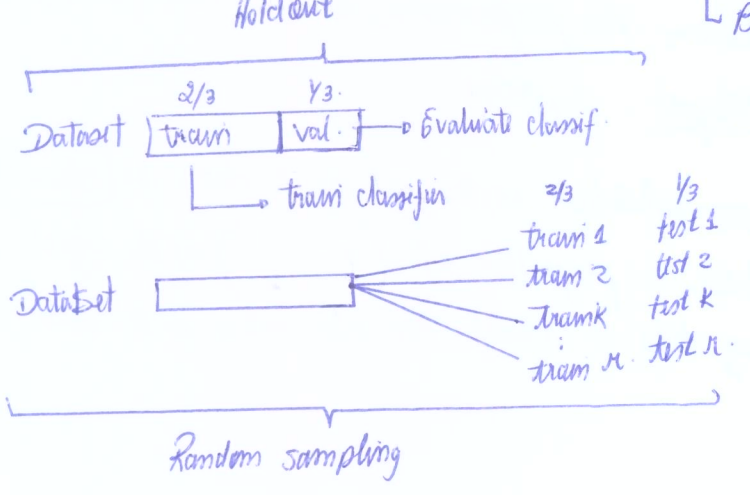
$$mse(\hat{f}) = \frac{1}{m} \cdot \sum_{i=1}^{m} \left(y_i - \hat{f}(x^{(i)})\right)^2$$

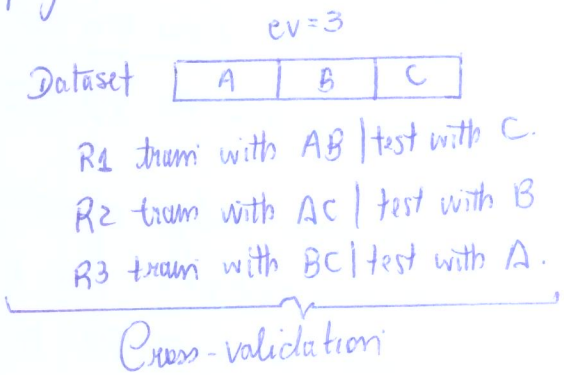
$$mAD(\hat{f}) = \frac{1}{m} \cdot \sum_{i=1}^{m} \left| y_i - \hat{f}(x^{(i)}) \right|$$

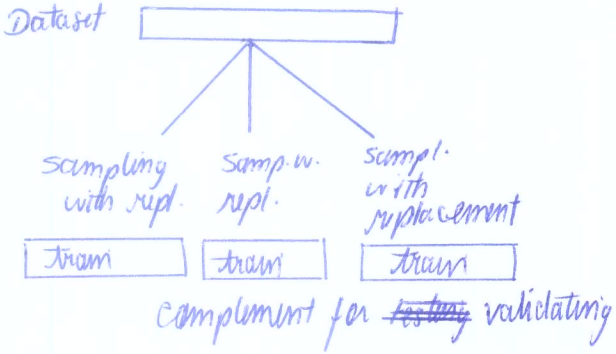
Sampling

Sometimes we do not have enough data for training and evaluating a classifier. In this case, as we already discussed in the last class, we need to divide the available data.

The most important sampling methods are:
- Holdout
- Random sampling
- Cross validation
- Bootstraping







In all cases, we calculate the average classification accuracy and variance / std

High std ⇒ model instability in learning params.

~~testing~~ Hold out ~~and~~ and random sampling

Hold out: we divide the dataset in (p) for training and (1-p) for testing. Normally, p = 2/3 is used. Critiques: ~~std~~ underestimation of the accuracy and no insight about classifier variation with different training data.

Random sampling: perform several hold out and report average and standard deviations of the results.

Cross-validation

~ In (k)-fold cross-validation, we split the ~~data~~ dataset into (k) parts ~~and~~, train with (k-1) and test with the remaining part. We repeat this process (k) times always changing the testing part. We report in the end the average accuracy and standard deviation.

~ In the case where k = m (m = # of elements in the dataset), we have the special case called leave one out validation. The performance in this case is the sum of accuracies over the (m) runs. Critiques: expensive.

~ The main critique regarding cross-validation is the part of the data is shared between training subsets. For k ≥ 2 folds, a proportion of $\left(1 - \frac{2}{k}\right)$ of the objects is shared. For k = 10, 80% of training data is shared. The alternative is to ~~use~~ ~~used~~ use a k x 2 cross-validation.

K×2 cross validation

Divide the data in two parts (A) and (B). Train with (A) and test with (B)

↓

Equally!

Then change. Train with (B) and test with (A). Repeat this (K) times and report the average and standard deviation. Normally, (K) ↓ division splitting

$K=5$

Bootstrapping

In this case, we create (K) training sets with sampling with replacement out of the ~~train~~ dataset. Non sampled data form the testing set each time. The final result is the average classification accuracy and standard deviation.

Normally, we use $K \geq 100$. The caveat: it is a method too expensive to perform.

There are many bootstrap methods and the commonest is the ℓ_0.

ℓ_0 bootstrap method

Each training set has (m) examples. Each example has probability of $1-\left(1-\frac{1}{m}\right)^m$ of being drawn at least once. For large (m), this tends to $\left(1-\frac{1}{e}\right) = 0,632$ which means the average fraction of non-repeated examples in the training is 63.2%. The final classification is given by the average across the sets and the standard deviation. The results are statistically equivalent to the leave one out but with a smaller variance.

Problems of classification with 2-classes

Consider a 2-class problem and the confusion matrix :

true class \ predicted class	+	−
+	TP	FN
−	FP	TN

$m = TP + TN + FP + FN$

(TP) : true positives or positive examples correctly classified.

(TN) : negative examples correctly classified.

(FP) : examples misclassified as ⊕ but that are indeed ⊖.

(FN) : examples misclassified as ⊖ but that are indeed ⊕.

Performance measures :

① False negative Rate (FNR) : proportion of examples in the class ⊕ wrongly classified.

$$f_{NR}(\hat{f}) = err_{\oplus}(\hat{f}) = \frac{FN}{TP + FN}.$$

② False positive Rate (FPR) : proportion of examples in the class ⊖ wrongly classified.

$$f_{PR}(\hat{f}) = err_{\ominus}(\hat{f}) = \frac{FP}{FP + TN}.$$

③ Total Error Rate :

$$err(\hat{f}) = \frac{FP + FN}{m}$$ where m is the sum of elements in the confusion matrix.

④ Total accuracy

$$acc(\hat{f}) = \frac{TP + TN}{m}$$

Precision: proportion of positive examples correctly classified among all examples predicted as positive.

$$precision(\hat{f}) = \frac{TP}{TP+FP}$$

Sensitivity or Recall: also known as true positive rate (TPR), it corresponds to the rate of correct classification for the positive class.

$$Sensitivity(\hat{f}) = recall(\hat{f}) = TPR(\hat{f}) = \frac{TP}{TP+FN}$$

Specificity: rate of correct classification for the negative class. Its complement is the FPR.

$$Especificity(\hat{f}) = \frac{TN}{TN+FP} = 1 - FPR(\hat{f})$$

Note ① we can easily extend these measures for multi-class by taking each class as positive each time and the remaining as negative and obtaining a performance measure for each class.

② For global accuracy in multi-class problems, we just consider the diagonal.

③ The precision can be seen as a measure of goodness of a model and recall as a measure of completeness. A ~~precision~~ precision = 1 for a given class means each element classified of being of that class actually is of that class. However it doesn't inform how many examples of c were wrongly classified. On the other hand, a Recall = 1 means all examples of class © were classified as being of class © but doesn't inform about how many examples of other classes were classified as of class ©.

④ Because precision and recall are always analyzed together, there is a combination for them into a single measure known as F-measure

F-measure : harmonic weighted mean of precision and recall.

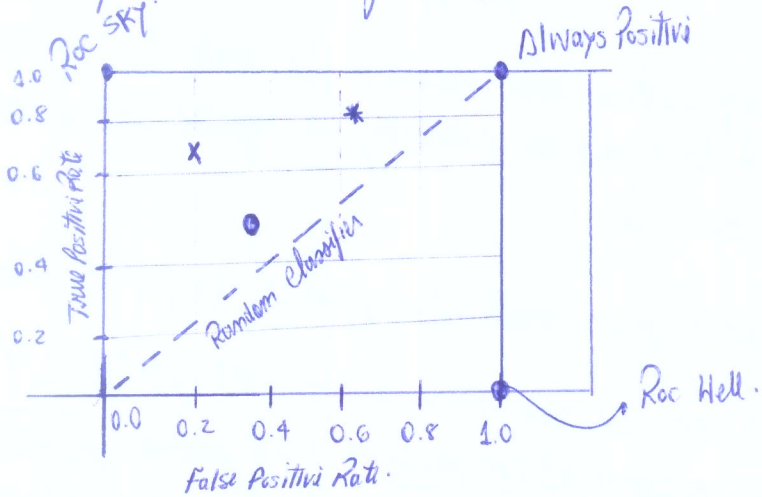
$$F_m(\hat{f}) = \frac{(w+1) \times recall(\hat{f}) \times precision(\hat{f})}{recall(\hat{f}) + w \times precision(\hat{f})}.$$

With $w=1$, means putting the same importance on the precision and recall and we end up with (F1)

$$F_1(\hat{f}) = \frac{2\, precision(\hat{f}) \times recall(\hat{f})}{precision(\hat{f}) + recall(\hat{f})}.$$

Receiver Operating Curve (ROC) Analysis

This is an alternative form of evaluating classifiers.



- x classifier (1)
- * classifier (2)
- • classifier (3).

A classifier is considered better if it is above and to the left of another in the Roc space.

The usual way to compare classifiers is to create a ROC curve instead of just plotting points.

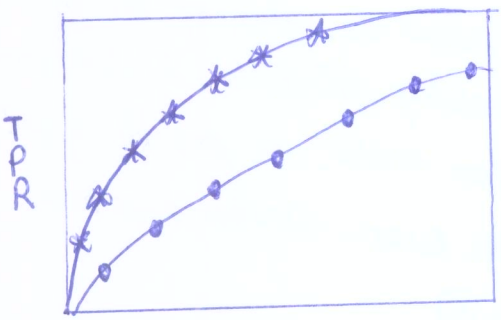
Roc curves

many classifiers produce a ~~continuum~~ real-valued output or can be adapted for that

Examples: MLPs, SVMs, Naive Bayes, Logistic Regression, etc.

Normally, these outcomes are discretized for issuing a final classification: $\hat{f}(x) \in \{0, 1\}$, for instance using a threshold ($> 0.5 \Rightarrow$ class +1). Plotting the TFP and TPR for different thresholds gives us a curve (Roc).

For comparing different algorithms with Rocs, we can calculate the area under the curve (AUC).



It is advisable to calculate different AUCs (from different data partitions) for a better statistical analysis and report average AUC and their standard deviation.

Roc Advantages:

1. Analyze different performance measures independ of the chosen thresholds
2. Allows studying different thresholds for tackling unbalancing on that data (classes of very different sizes).

   Example: Consider a case of a class (A) with 100 examples and (B) with just 1.

Roc Disadvantages

1. Appropriate for 2-class problems.
2. For more classes, (1 vs All) methods and several curves can be used.

Hypotheses Testing

In many situations, we need to decide which technique is better for a given problem. For that, we need to evaluate them on our datasets and perform statistical testing.

The common strategy is to partition the dataset in such a way we train the algorithms to be compared with the same data and test them also on the same data.

For example, consider the case of a **(K)-fold cross validation**.

After training/testing, each algorithm has (K) performance measures (e.g. accuracies). Now to compare them based on their averages is not enough. We need a statistical test.

Statistical Hypothesis: allegation about the value of one or more parameters or about a given probability distribution. If (μ_1) and (μ_2) are the average errors of two models, a possible hypothesis would be $\mu_1 - \mu_2 = 0$ or equivalent errors. Another hypothesis: $\mu_1 - \mu_2 > 0$ error (1) is higher than (2).

* In a statistical test, normally we have two contradictory options

$$\begin{cases} H_0: \mu_1 - \mu_2 = 0 \\ H_1: \mu_1 - \mu_2 \neq 0 \end{cases}$$

Our task is to decide which one holds true.

Initially, we assume H_0 (null hypothesis) as TRUE. (H_1) is then the alternative. The null hypothesis is rejected in favor of (H_1) if some evidence on the samples in the experiment suggest (H_0) is false. The rule for deciding if (H_0) is rejected is called ~~statistical~~ test.

In such a test, we have
1. a statistical test based on the data samples (e.g. classifier accuracy)
2. a rejection region for which (H_0) is rejected.

The test procedures can be wrong due to sample variation and other reasons. Two types of error:

- Type (I) error: (H_0) is rejected but it is true.
- Type II error: (H_0) is false but is not rejected.

Type (I) is more serious, therefore most statistical tests try to control this error using a parameter (α) also known as the test significance. This is the same as saying, for $\alpha = 0.05$ that a test has 95% confidence level of not having rejected (H_0) when it was true.

We will discuss two tests that are not parametric (do not have the restriction of their data following a distribution)

1. Wilcoxon-signed rank
2. Friedman

Both tests are paired (same tests data) and non-parametric (data doesn't need to follow a distribution).

Two application scenarios for tests

1. Compare different algorithms on the same data. (each point can be accuracy for a fold).
2. Compare different alg. considering different datasets (each point can be accuracy for a dataset).

Comparing two models/classifiers

It is recommended the Wilcoxon-signed-ranks (also known as Mann-Whitney U test) in which we calculate inially the performance differences of the two models being compared and then we rank such differences in absolute value (lower to higher). with the rank, we compare the differences in positions for the algorithms.

Given two algorithms (A) and (B) if we always calculate the differences as (B-A) using a performance measure whose higher values are better (e.g. AUC) positive differences show a better performance of (B).

Example

Accuracy

Datasets	Alg. A	Alg. B	B-A Difference	|B-A| Absolute Diff	Position
Lung	0.583	0.583	0.000	0.000	1.5
Funghi	0.583	0.583	0.000	0.000	1.5
Atmosphere	0.882	0.888	+0.006	0.006	3.0
Breast	0.599	0.591	-0.008	0.008	4.0

Rules for position: for ties, sum positions and put the averages. Position 1 + Position 2 / 2

Now consider R^+ the sum of the ranks for which (B) is better than (A). R^- is the contrary (A > B). Positions of ties are equally divided. (if we have ~~the odd~~ number of ties, ~~one is discarded~~) a null difference, it is discarded.

$$R^+ = \sum_{di>0} rank(di) + \frac{1}{2}\sum_{di=0} rank(di). \qquad R+ = 3 + 3/2 = 4.5.$$

$$R^- = \sum_{di<0} rank(di) + \frac{1}{2}\sum_{di=0} rank(di). \qquad R- = 4 + 3/2 = 5.5.$$

Let W be the smaller of these sums. Then the statistical calculation is

$$z = \frac{W_{wilcox} - 1/4\, N(N-1)}{\sqrt{\frac{1}{24} N(N+1)\cdot(2N+1)}}$$

with $\alpha = 0.05$, the null hypothesis that (A) and (B) behave the same way can be rejected if $z < -1.96$.

(N) number of elements (rows) in the table (e.g.; folds/datasets). Values for (N) up to 25 are given in statistical books. For more, use the formula above.

$W < z$, Ho rejected under α

Another example:

fold set	Alg. (A)	Alg. (B)	Diff (A-B)	Pos	Ranked Diff.	
1	25	32	-7	7	7.5	7.5
2	29	30	-1	1	2.5	2.5
3	10	8	2	2	5.5	5.5
4	31	32	-1	1	2.5	2.5
5	27	20	7	7	7.5	7.5
6	24	32	-8	8	9	9
7	26	27	-1	1	2.5	2.5
8	29	30	-1	1	2.5	2.5
9	30	32	-2	2	5.5	5.5
10	32	32	0			
11	20	30	-10	10	10	10
12	5	32	-27	27	11	11

4 1+2+3+4
10/4 = 2.5

5+6 = 11/2 = 5.5
7+8 = 15/2 = 7.5
9

Add the ranks

$R_+ = 13$

$R_- = 53$

$W = \min(R_+, R_-) = 13$

$N = 12 - 1 = 11$

(one ignored (zero) entry)

Using a table of critical values

2-tailed test

N	$\alpha = 0.05$	$\alpha = 0.02$
6	0	–
7	2	0
8	4	2
9	6	3
10	8	5
11	10	7
12	13	10

Just an example.

2-tailed test ⇒ no preference for either side $H_0 : u_1 = u_2$

As $z = 10$ and $W > z$ for $\alpha = 0.05$, there is no evidence for rejecting H_0 (A and B are the same).

Observation here we discard the null terms and perform.

$$R^+ = \sum_{d_i > 0} \text{rank}(d_i)$$ (no term for zero).

Rule $W < z \Rightarrow$ **there is** difference to reject H_0.

Comparing multiple algorithms/models

When we compare multiple techniques, we need to change the test given that we have more comparisons (this is more chance of detecting a difference that doesn't exist).

For J tests, the probability of ending up with **at least** one mistake/error is

$$1 - (1-\alpha)^J$$

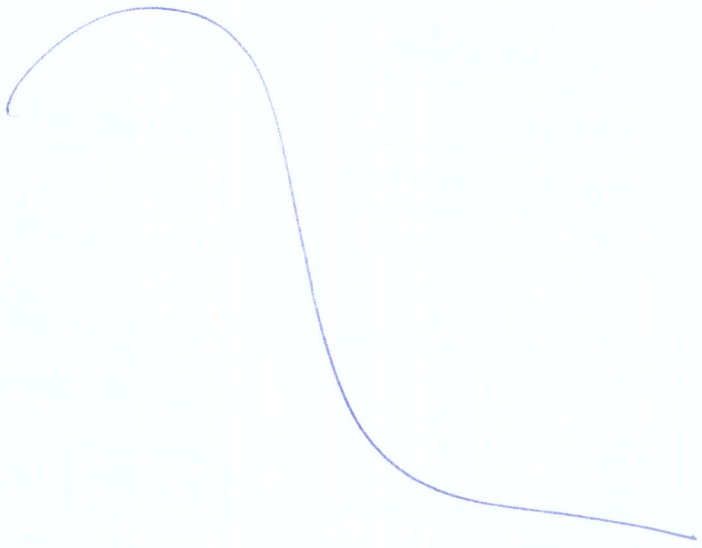
For $J = 20$ and $\alpha = 0.05$,

$$1 - (1-0.05)^{20} \approx 64\%$$

This is known as the multiplicity problem.

For multiple tests, we turn to the Friedman test that also relies on rank comparisons. However, now the absolute performance value for each algorithm is compared in each dataset. In other words, we rank the algorithms several times, one for each dataset.

See example in the next page.



Wilcoxon test Worked Example:

In order to investigate whether adults report verbally presented material more accurately from their right than from their left ear, a dichotic listening task was carried out. The data were found to be positively skewed.

Number of words reported:

Participant	Left ear	Right ear
1	25	32
2	29	30
3	10	8
4	31	32
5	27	20
6	24	32
7	26	27
8	29	30
9	30	32
10	32	32
11	20	30
12	5	32

What test should we use?

We have two conditions, with each participant taking part in both conditions. Since we are told that the data are skewed, and so do not fulfil the requirements for the use of a parametric test, the appropriate analysis is a Wilcoxon test

(the non-parametric counterpart of the dependent measures t-test).

How to carry out a Wilcoxon test:

a) Find the differences between each pair of scores
b) Rank these differences, ignoring any "0" differences and ignoring the sign of the difference

[So, different from a Mann Whitney U test, where the data themselves are ranked]

STEP ONE:

Participant	Left ear	Right ear	Difference (d)
1	25	32	- 7
2	29	30	- 1
3	10	8	2
4	31	32	- 1
5	27	20	7
6	24	32	- 8
7	26	27	- 1
8	29	30	- 1
9	30	32	- 2
10	32	32	0
11	20	30	- 10
12	5	32	- 27

STEP THREE:

Add together the ranks belonging to scores with a positive sign:

5.5 + 7.5 = 13

STEP FOUR:

Add together the ranks belonging to scores with a negative sign:

7.5 + 2.5 + 2.5 + 9 + 2.5 + 2.5 + 5.5 + 10 + 11 = 53

STEP FIVE:

Whichever of these sums is the smaller, is our value of W. So, W = 13.

STEP SIX:

N is the number of differences (omitting "0" differences).
We have 12 – 1 = 11 differences.

[Important to note that these are not the same as degrees of freedom. We only use N–1 here because we have 1 difference which equals zero]

STEP SEVEN:

Use the table of critical Wilcoxon values. With an N of 11, what is the critical value for a two-tailed test at the 0.05 significance level?

Critical values For Wilcoxon's signed-rank test

	Two-Tailed Probability			
N	*.05*	*.025*	*.01*	*.005*
5	*1*			
6	*2*	*1*		
7	*4*	*2*	*0*	
8	*6*	*4*	*2*	*0*
9	*8*	*6*	*3*	*2*
10	*11*	*8*	*5*	*3*
11	*14*	*11*	*7*	*5*
12	*17*	*14*	*10*	*7*
13	*21*	*17*	*13*	*10*
14	*26*	*21*	*16*	*13*
15	*30*	*25*	*20*	*16*
16	*36*	*30*	*24*	*19*
17	*41*	*35*	*28*	*23*
18	*47*	*40*	*33*	*28*
19	*54*	*46*	*38*	*32*
20	*60*	*52*	*43*	*37*

The critical value = 14.

With the Wilcoxon test, an obtained W is significant if it is LESS than the critical value.

So, in this case

Obtained W = 13
Critical value = 14

Our obtained value of 13 is *less* than 14, and so we can conclude that there is a difference between the number of words recalled from the right ear and the number of words recalled from the left ear.

The Friedman test

The Friedman test is a test for comparing three or more related samples and which makes no assumptions about the underlying distribution of the data. The data is set out in a table comprising n rows by k columns. The data is then ranked across the rows and the mean rank for each column is compared.

Example. A water company sought evidence the measures taken to clean up a river were effective. Biological oxygen demand (BOD) at 12 sites on the river were compared before cleanup and 1 month and 1 year after cleanup. The results are given below.

	A	B BOD	C
Site	**before**	**after 1 month**	**after 1 year**
1	17.4	13.6	13.2
2	15.7	10.1	9.8
3	12.9	10.3	10.3
4	9.8	9.2	9.0
5	13.4	11.1	10.7
6	18.7	20.4	19.6
7	13.9	10.4	10.2
8	11	11.4	11.5
9	5.4	4.9	5.2
10	10.4	8.9	9.2
11	16.4	11.2	11.0
12	5.6	4.8	4.6

can be seen as one algorithm

K = 3 algorithms / approaches

can be seen as datasets.

can be any performance measure e.g. error in classification.

The Friedman test involves ranking the data in the rows, then comparing the mean rank in each column. Thus the values of BOD would be ranked across each row as shown below. Where two samples have the same value a mean rank is assigned. (site 3)

	A	B BOD	C
Site	**before**	**after 1 month**	**after 1 year**
1	3	2	1
2	3	2	1
3	3	1.5	1.5
4	3	2	1
5	3	2	1
6	1	3	2
7	3	2	1
8	1	2	3
9	3	1	2
10	3	1	2
11	3	2	1
12	3	2	1

If the cleanup procedure had been ineffective, the ranking of values over time would be randomly distributed at the various sites and the sum of the ranks for each column would be similar. However, if the cleanup procedure were effective, there would be significant differences in the sum of the ranks of at least one column.

Null hypothesis

H_0: The cleanup procedure has had no effect on the BOD Same as algorithms are equally effective.

H_1: The cleanup procedure has affected the BOD Algorithms are "different".

Decision Rule

Reject H_0 if M ≥ critical value at $\alpha = 5\%$

Calculation method

The differences between the sum of the ranks is evaluated by calculating the Friedman test statistic, M from the formula

$$M = \frac{12}{nk(k+1)} \sum R_j^2 - 3n(k+1)$$

Where:

k = number of columns (often called "treatments")

n = number of rows (often called "blocks")

R_j = sum of the ranks in column j.

If there is no significant difference between the sum of the ranks of each of the columns, then M will be small, but if at least one column shows significant difference then M will be larger.

For the BOD example these calculations work out as follows

	BOD		
Site	**before**	**after 1 month**	**after 1 year**
Sum of ranks	32	22.5	17.5
(Sum of ranks)^2	1024	506.25	306.25

No of Columns, k	3	
No of Rows, n	12	
ΣR^2	1836.5	(= 1024 + 506.25 + 306.25)
12/nk(k+1)	0.083	(= 12/12 x 3 x 4)
3n(k + 1)	144	(= 3 x 12 x 4)
Test Statistic M	9.042	(= 0.083 x 1836.5 – 144)

The significance of M may then be looked up in tables.

The critical value of M for 3 columns and 12 rows at $\alpha = 5\%$ is 6.5

Thus M > critical value so we can reject H_0 and conclude that the treatment has had a significant effect on the BOD for that stretch of river.

If the values of *k* and/or *n* exceed those given in tables, the significance of M may be looked up in chi-squared (χ^2) distribution tables with *k-1* degrees of freedom.

The null hypothesis is rejected if $F_f > F_{(k-1),(k-1)\cdot(N-1)}$ (Friedman)

$M >$ Prob. dist. with $(c-1)$, $(c-1)\cdot(N-1)$ degrees of freedom.

Friedman in the sample.

In case we reject H_0, there is statistical difference but we still need to find which algorithms have it. For finding that, we need to perform a post-test

Post-tests

The performance of two algorithms is statistically different if the difference of their average ranking is greater than the critical value

C = K = nbr classifiers/techniques

$$CD = q_\alpha \cdot \sqrt{\frac{C(C+1)}{6N}}$$

If all algorithms are compared 2x2, we can find q_α using Nemenyi statistics. When we compare one algorithm to the rest, we can use the Bonferroni Dunn statistics.

	2	3	4	5	6
Nemenyi	1.96	2.343	2.569	2.728	2.850
Bonferroni-Dunn	1.96	2.241	2.399	2.498	2.576

algorithms

Example:

In the example we just saw, we want to check if B is better than C or the contrary.

$$CD = \sqrt{3(4)/(12\times 6)} \cdot 2.343 = \sqrt{12/(2\times 6)} \cdot 2.343 = \sqrt{1/6} \cdot 2.343 = \frac{\sqrt{6}}{6} \cdot 2.343 = 0.9565$$

The average ranking difference for C and B is:

$$B = \frac{22.5}{12} = 1.875$$

$$C = \frac{17.5}{12} = 1.4583$$

$> CD = 0.4167$

B is not better than C or C is not better than B.

What about C x A?

$$A = \frac{32}{12} = 2.667$$

$2.667 - 1.4583 = 1.208$

C is better than A!

Critical values for the Friedman Test

$$M = \frac{12}{nk(k+1)} \sum R_j^2 - 3n(k+1)$$

	k=3		k=4		k=5		k=6	
n	α=5%	α=1%	α=5%	α=1%	α=5%	α=1%	α=5%	α=1%
2	—	—	6.000	—	7.600	8.000	9.143	9.714
3	6.000	—	7.400	9.000	8.533	10.130	9.857	11.760
4	6.500	8.000	7.800	9.600	8.800	11.200	10.290	12.710
5	6.400	8.400	7.800	9.960	8.960	11.680	10.490	13.230
6	7.000	9.000	7.600	10.200	9.067	11.870	10.570	13.620
7	7.143	8.857	7.800	10.540	9.143	12.110	10.670	13.860
8	6.250	9.000	7.650	10.500	9.200	13.200	10.710	14.000
9	6.222	9.556	7.667	10.730	9.244	12.440	10.780	14.140
10	6.200	9.600	7.680	10.680	9.280	12.480	10.800	14.230
11	6.545	9.455	7.691	10.750	9.309	12.580	10.840	14.320
12	6.500	9.500	7.700	10.800	9.333	12.600	10.860	14.380
13	6.615	9.385	7.800	10.850	9.354	12.680	10.890	14.450
14	6.143	9.143	7.714	10.890	9.371	12.740	10.900	14.490
15	6.400	8.933	7.720	10.920	9.387	12.800	10.920	14.540
16	6.500	9.375	7.800	10.950	9.400	12.800	10.960	14.570
17	6.118	9.294	7.800	10.050	9.412	12.850	10.950	14.610
18	6.333	9.000	7.733	10.930	9.422	12.890	10.950	14.630
19	6.421	9.579	7.863	11.020	9.432	12.880	11.000	14.670
20	6.300	9.300	7.800	11.100	9.400	12.920	11.000	14.660
∞	5.991	9.210	7.815	11.340	9.488	13.280	11.070	15.090

For values of ***n*** greater than 20 and/or values of ***k*** greater than 6, use χ^2 tables with ***k*-1** degrees of freedom

Critical Values of the Wilcoxon Signed Ranks Test

n	Two-Tailed Test		One-Tailed Test	
	$\alpha = .05$	$\alpha = .01$	$\alpha = .05$	$\alpha = .01$
5	--	--	0	--
6	0	--	2	--
7	2	--	3	0
8	3	0	5	1
9	5	1	8	3
10	8	3	10	5
11	10	5	13	7
12	13	7	17	9
13	17	9	21	12
14	21	12	25	15
15	25	15	30	19
16	29	19	35	23
17	34	23	41	27
18	40	27	47	32
19	46	32	53	37
20	52	37	60	43
21	58	42	67	49
22	65	48	75	55
23	73	54	83	62
24	81	61	91	69
25	89	68	100	76
26	98	75	110	84
27	107	83	119	92
28	116	91	130	101
29	126	100	140	110
30	137	109	151	120