

# MLR 570

Recording from Sept. 16: (part 1)

↳ PCA: maximizing variance vs. minimizing reconstruction error.

eigenvalue decom-  
on covariance  
matrix.

singular value  
decomp. on  
dataset itself.

Correlation: Derivation.

feature selection: filter methods [independent of models]

var threshold

$$\text{var}(X_j) = \frac{1}{n} \sum (x_{ij} - \mu_i)^2$$

feature  $\in \mathbb{R}^n$

corr. coeff.

$$\rho(X_j, y) = \frac{\text{cov}(X_j, y)}{\sigma_{X_j} \sigma_y}$$

mutual information

Put in equation  
form.

Below threshold, remove.  
because don't contribute.

$$\sigma_{X_j} \sigma_y$$

If full data,  
leverage of data?

Wrapper methods

[based on model]

importance of features depends on  
the specific models we are  
using.

RFE: rank based  
on importance & then  
recursively eliminate.  
→ rank w/ all, rank  
(model coefficients), remove  
least important.

FFS (forward feature selection)

No features → add features  
based on improvement  
1st most important feature, +++,

Embedded methods [introduce something to obj. function]  
to force features to be zero

During training

L1 regularization

force some feature coefficients  
to become 0, selecting subset  
of features.

$$\min_w \frac{1}{2n} \|Xw - y\|_2^2 + \lambda \|w\|_1$$

$\underbrace{\quad}_{L2 \text{ norm}} \qquad \underbrace{\quad}_{L1 \text{ norm}}$

Initialization:

Bad → algo in bad  
point. Not converge to  
good place.

Non-convex space!

Norm: map vector to scalar  
 $a = [1 \ 2 \ 3]$

$$\|a\|_2 = \sqrt{1^2 + 2^2 + 3^2}$$

$$\|a\|_1 = 1 + 2 + 3$$

↳ L1, encourage vector  
to be sparse during training.

Similarity measures:

→ correlation coeff.:  $\rho = \frac{\text{cov}(A, B)}{\sigma_A \sigma_B}$

linear relationship between variables  
fails for non-linear relationships.

<p>def. variance: <math>E(X^2) - [E(X)]^2 = \text{var}(X)</math></p> <p>Definition based on expectation.</p>	$\sigma^2 = \text{variance}$ $\sigma = \sqrt{\text{var}(X)}$ standard deviation	$\text{var}(\alpha A) = \alpha^2 \text{var}(A)$ $\text{cov}(A, B) = \frac{1}{n} (\bar{A}_i - \bar{A})(\bar{B}_i - \bar{B})$ covariance value.
--	--	---

ρ cannot capture non-linear relationship. Does not mean that there's no relationship.

→ Rolling coefficient? As opposed to the whole data.

Euclidean distance: straight-line distance between two points in Euclidean space

↑ In high dimensions, it will not carry same information

High dimensional statistics

Limitations: does not have a meaning.

$$d(A, B) = \sqrt{\sum_{i=1}^n (A_i - B_i)^2}$$

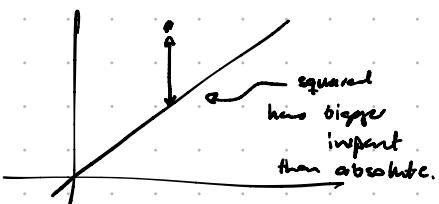
Manhattan distance: Along right angles. Grid structure. Restricted, streets etc..

$$d(A, B) = \sum_{i=1}^n |A_i - B_i|$$

### Recording from Sept. 16: (part 2)

Adv. Manhattan over Euclidean → grid

→ High dimensional data: curse of dimensionality. dist. converges in higher dims.



Manhattan more stable. Axis independent.

→ Better if features are independent. (Manhattan better)

→ Manhattan more robust to outliers.

Closeness of clusters.

Cosine Similarity: cos of angle between two vectors. Direction instead of magnitude.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

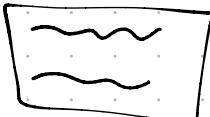
Same direction, different magnitudes  
 $\Rightarrow \cos(\theta) = 1 \rightarrow \text{angle is } 0$

Robust to different sizes when comparing. How change word to vector? encoding frequency. ( $\cos(C1) = 0$ )  
 Word2vec embedding. Not part of course.

Jaccard similarity: Binary or categorical data. Document similarity is an application.

$$J = \frac{|A \cap B|}{|A \cup B|}$$

Also robust to size different sizes.



## Recording from Sept. 23: (part 1):

### Lecture 5: Model evaluation

Exam: Think deeply about concepts.  
Understand concepts well.

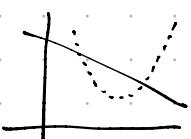
Training  $\rightarrow$  training error

Validation  $\rightarrow$  fine tune the hyperparameters.  
 $\hookrightarrow$  validation error.

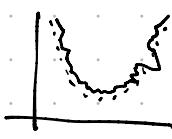
$\leftarrow$  intermediate testing set.

Testing  $\rightarrow$  Generalization on unseen data.

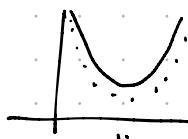
### Underfitting / Overfitting



underfitting



overfitting

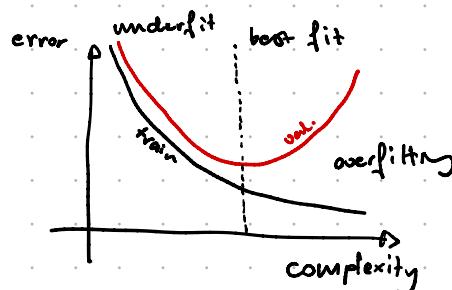


optimal.

high training  
& testing error

low training error  
high testing error

$\leftarrow$  model too  
complex



/ this can be  
the num. epochs as  
well.

Expressive power: how powerful model is regardless of data. Depends on parameters of the algorithm.  
outside of course

We select model based on the different errors. Similar to model hyperparameter tuning.

K-fold cross validation

$K-1$  folds for testing.



Can also look at the maximum possible error instead of average.  
Reduces overfitting risk  
More reliable.

Central limit theorem: error converges to certain value / distribution.

(n) business? Shuffling? Other things.

	predicted		
actual	TN	FP	
	FN	TP	

always misleading  
for imbalanced  
data.

We don't know acc. based on  
the classes. Just overall.

Precision: proportion of true positives  
among all predicted positive } Limitation: not useful  
if we care about FN

Type I error:  $1 - \text{precision}$

Recall: proportion of true positives  
among all actual positives } If recall comes at  
the cost of precision.

Type II error:  $1 - \text{recall}$

$$F_1 \text{ score: } 2 \times \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

$\leftarrow$  does not count for TN.

Harmonic mean of precision &  
recall.

Why the harmonic mean? Balances between prec. & recall. Penalizes the extreme values  
more than the geometric or arithmetic mean.

$\hookrightarrow$  Esp. if one is significantly smaller than the other.

Arithmetic: equal weights to  
both metrics

Harmonic mean is monotonic to both precision & recall. If one increases, it also increases

Geometric mean: multiplicative process. Growth, returns, compounded.

Harmonic  $\leq$  geometric  $\leq$  arithmetic  
 ↓                   ↑  
 lower bounds each      } Equality if precision = recall.

## Recording from Sept. 23: (part 2)

Begin with numerical examples. Harmonic mean penalizes extreme values.

If we swap FN & FP, then the  $F_1$  score does not change. If prec. & recall swapped while everything else stayed the same, it wouldn't change  $F_1$ . [huge limitation].

↳ weighted  $F_1$  score allows us to differentiate

$$F_p = (1 + \beta^2) \frac{\text{Precision} \cdot \text{recall}}{(\beta^2 \text{precision}) + \text{recall}}$$

What if  $\beta$  value too large? then we have bad representation.

Generalized mean:

maximum  $p=10$

cubic mean  $p=3$

rms  $p=2$

arithmetic  $p=1$

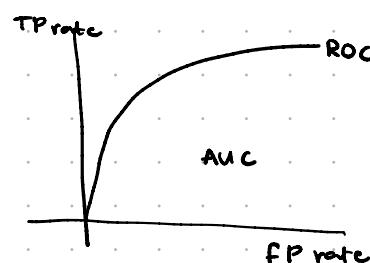
geometric  $p=0$

harmonic  $p=-1$

minimum  $p=-\infty$

$$M_p(x_1, \dots, x_n) = \left( \frac{1}{n} \sum_{i=1}^n x_i^p \right)^{1/p}$$

Receiver-operator characteristic (type I, II error)



Mean squared error:  $\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$  → heavy penalty for large values (squaring), amplified sensitive to outliers.

Mean absolute error:  $\frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$  Better for handling outliers.

↑ optimize this loss function if we have outliers

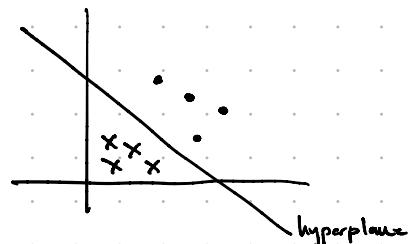
Cannot take derivative of absolute value func. Need to perturb, approximate, etc...

Side point: regularization does not only prevent overfitting, it helps us make sure the model does not learn things we don't want it to learn.

## Recording from Sept. 30<sup>th</sup>: (part 1)

Supervised learning:  $f: X \rightarrow Y$ ,  $X \in \mathbb{R}^n$  to output label  $Y$

/ \ regression classification



↳ Linear classification:

↳ Linear separability: A dataset is linearly separable if  $\exists$  hyperplane s.t. all datapoints belonging to one class are on one side, of vice versa.

For  $\mathbb{R}^n$ , hyperplane:  $wz + b = 0$ ,  $z \in \mathbb{R}^n$ . For linear separability condition,  $\exists w, b$  s.t.

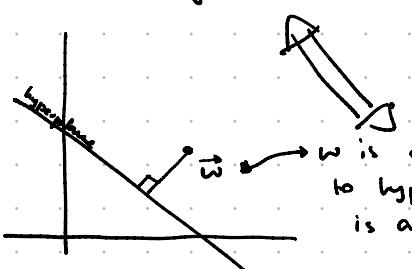
$w = \text{weights}$ ,  $b = \text{biases}$

$$\sum_i y_i (wz_i + b) > 0 \forall i$$

$y_i(wx_i + b) \rightarrow$  for some  $w, b$ , if  $y_i(wx_i + b) > 0, y_i(wx_i + b) > 0$  or  $y_i < 0, wx_i + b < 0$   
 $y_i(wx_i + b) < 0, \Rightarrow y_i > 0, wx_i + b < 0$   
or  $y_i < 0, wx_i + b > 0$

Orthogonal to the hyperplane.

Moving above is one class, moving below will give another class.

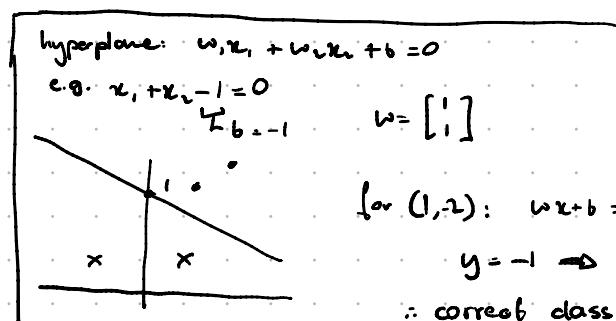


If we move in direction of  $\vec{w}$ , positive class. If opposite direction, negative class.

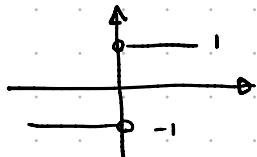
Write the equation in standard form,

$$wx + b = 0$$

hyperplane



Perceptron Algorithm:  $f(x) = \text{sign}(wx + b)$  where  $b \in \mathbb{R}$



Hinge loss:  $L(w, b) = \max(0, -y_i(wx_i + b))$

↳ no update if loss is 0.

if correctly classified,  $y_i(wx_i + b) > 0$

$$\therefore -(-y_i(wx_i + b)) < 0, \Rightarrow \max(0, (-))$$

= 0. No loss.

↳ no loss → no update

Otherwise,  $\max(0, (+)) > 0$

for incorrectly classified.

loss  $\neq 0$ . update.

Partial derivatives:  $\frac{\partial L}{\partial w} = \begin{cases} -y_i x_i & \text{if } y_i(wx_i + b) \leq 0 \\ 0 & \text{otherwise} \end{cases}$

$\frac{\partial L}{\partial b} = \begin{cases} -y_i & \text{if } y_i(wx_i + b) \leq 0 \\ 0 & \text{otherwise} \end{cases}$

Gradient descent approach

$$\text{updates: } w \leftarrow w - \gamma \frac{\partial L}{\partial w} = w + \gamma y_i x_i$$

$$b \leftarrow b - \gamma \frac{\partial L}{\partial b} = b + \gamma y_i$$

The direction of  $w$  for optimizing (where  $w$  is orthogonal to hyperplane) is super important in general.

Perceptron has infinite number of solutions. Local updates, not global updates

order that we approach with also matters.

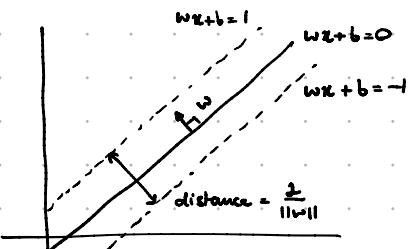
↑ computationally expensive.  
many algorithms do this.

Limitations:

- Linearity requirement. Diverges for non-linear data.
- No "margin" optimization.

Recording from Sept. 30<sup>th</sup>: (part 2)

SUMs: builds on the limitations of perceptron (margin optimization)



Now, the correctly classified points satisfy  $y_i(wx_i + b) \geq 1 \forall i$

To maximize the margin, we minimize  $\|w\|$  or  $\frac{\|w\|^2}{2}$  (makes taking derivative easier).

Optimization:  $\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2$  subject to  $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i$

Using Lagrange multipliers:  $L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1)$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \Rightarrow \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L}{\partial b} = - \sum_{i=1}^n \alpha_i y_i \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0$$

replace  $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$ :  $\max_{\alpha} \left[ -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + \sum_{i=1}^n \alpha_i \right]$

$$\text{s.t. } \sum_{i=1}^n \alpha_i y_i = 0, \alpha_i \geq 0 \quad \forall i$$

Why  $\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2$ ?

The margin is the perpendicular distance between hyperplane & closest data points.

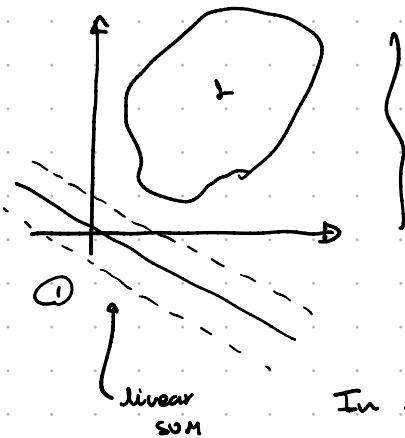
The distance from  $\mathbf{x}_0$  to hp:  $d = \frac{|\mathbf{w} \mathbf{x}_0 + b|}{\|\mathbf{w}\|}$

The support vectors pass through  $\mathbf{w} \mathbf{x}_i + b = \pm 1$ . These are two other hyperplanes, not the decision boundary.

Total margin size:  $\frac{2}{\|\mathbf{w}\|} \rightarrow$  why?  $d = \frac{|\mathbf{w} \mathbf{x}_0 + b|}{\|\mathbf{w}\|}$ , at top margin,  $|\mathbf{w} \mathbf{x}_0 + b| = 1$ , at bottom  $|\mathbf{w} \mathbf{x}_0 + b| = -1 = 1$

$\Rightarrow \frac{1}{\|\mathbf{w}\|} + \frac{1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$ . Margin is inversely proportional to  $\|\mathbf{w}\|$ . Larger  $\|\mathbf{w}\|$  means smaller margin.

When SVM / maximum margin bad?



In this case, class 2 has huge variance.  
Thus maximum margin may be troublesome.

→ solution can be soft margin or SVMs.

Huge limitation of SVMs: computational complexity.

In general, bigger margin means more robustness.

Linear regression:

$b: X \in \mathbb{R}^{n \times p} \rightarrow Y \in \mathbb{R}^n$

where  $y = X\beta + \epsilon$

If  $\epsilon$  is distributed normally with constant covariance, then we have optimality.

$$\min_{\beta} \|y - X\beta\|_2^2 = \min_{\beta} \sum_{i=1}^n (y_i - X_i \beta)^2$$

Assume  $p \leq n$ .

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \|X\|_2^2 = (\sqrt{x_1^2 + x_2^2 + \dots + x_n^2})^2 = x_1^2 + x_2^2 + \dots + x_n^2 = \mathbf{x}^T \mathbf{x} = [x_1 \ x_2 \ \dots \ x_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\|y - X\beta\|_2^2 = (y - X\beta)^T (y - X\beta)$$

$$= y^T y - y^T X \beta - \beta^T X^T y + \beta^T X^T X \beta \leftarrow \text{minimize}$$

$$\frac{\partial}{\partial \beta} = 0 = 0 - (y^T X)^T - (X^T y) + X^T X \beta + (\beta^T X^T X)^T = 0$$

$$- X^T y - X^T y + X^T X \beta - X^T X \beta = 0$$

$$= -2X^T y + 2X^T X \beta = 0$$

$$\Rightarrow 2X^T y = 2X^T X \beta$$

$$\boxed{\beta = (X^T X)^{-1} X^T y}$$

This is what we use to optimize.

$$\beta = (X^T X)^{-1} X^T y$$

When can we move  $\ominus$  inside?

$$\text{So } \beta = X^{-1} (X^T)^{-1} X^T y ; X \in \mathbb{R}^{n \times p},$$

$$= X^{-1} y \quad \rightarrow \text{if } n=p \text{ assuming full rank.}$$

Why do we do  $(X^T X)^{-1}$  instead of  $(X X^T)^{-1}$ ?

$p \leq n$ , so  $(X X^T)$  is singular, cannot take the inverse.  $\text{rank}(X) \leq \min(n, p)$ .

Limitations:

- Assumes linear relationship

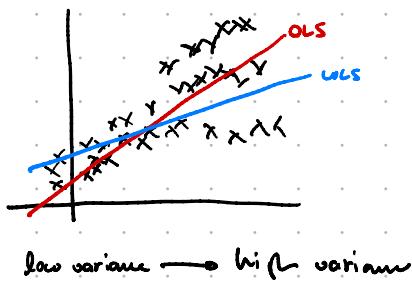
- Constant variance

- Singularity in  $(X^T X)^{-1}$   $\rightarrow$  perturbs  $(X^T X + \epsilon I)^{-1}$  exists

Weighted least squares (WLS): assumes observations have non-constant variance.

errors associated w/ different observations are not the same.

observations w/ higher variance have less weight, and vice versa.



$$\min_{\beta} \sum_{i=1}^n w_i (y_i - X_i \beta)^2$$

$$w_i = \frac{1}{\sigma_i^2} \quad \begin{matrix} \text{variance} \\ \text{of observation} \end{matrix}$$

w is a diagonal matrix.

$$\begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_n^2 \end{bmatrix}$$

Derivation:

$$\min_{\beta} (y - X\beta)^T w (y - X\beta)$$

$$\Rightarrow \frac{\partial}{\partial \beta} = X^T w y - X^T w X \beta$$

$$\Rightarrow \beta = (X^T w X)^{-1} X^T w y$$

$X^T X$  can be singular, often because of multi-collinearity. Unstable  $\rightarrow$  OLS unreliable.

Ridge regression:  $\min_{\beta} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2$

regularization term.

$\lambda$  shrinks coefficients to stabilize the solution.

$$\Rightarrow (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta$$

$$-2X^T y + 2X^T X \beta + 2\lambda \beta = 0$$

$$X^T y = X^T X \beta + 2\lambda \beta$$

$$X^T y = (X^T X + \lambda I) \beta$$

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

new factor that does perturbation to stabilize.

$\lambda \geq 0$ . if  $\lambda = 0$ , we get OLS

if  $\lambda > 0$ , penalize large coefficients.

## Recording from Oct. 7<sup>th</sup>: (part 1)

Lasso and ElasticNet added.

$$\text{Lasso: } \min_{\beta} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1,$$

$$\text{Elastic: } \min_{\beta} \|y - X\beta\|_2^2 + \lambda_1 \|\beta\|_2^2 + \lambda_2 \|\beta\|_1,$$

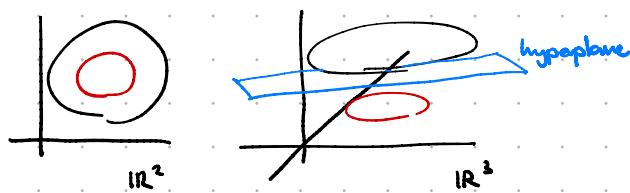
- flexible reg.
- stability for correlated features
- flexible regularization.

- good for feature selection, encourages sparsity
- high-dim data
- Interpretability
- instability with correlated features
- bias
- no closed form solution.

*Not part of course!*

Huge limitation of linear classification

Projection of lower dimm to higher dimm. for some  $x \in \mathbb{R}^{d_1}$ ,  $\phi(x): \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$  with  $d_2 > d_1$ . In this space, we can use the same linear models (because of linear separability)



Let  $x = [x_1 \ x_2]^T \in \mathbb{R}^2$ . Then define  $\phi(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{bmatrix} \in \mathbb{R}^3$

The transformation makes the data linearly separable.

$$h(\phi(x); \theta, \theta_0) = \text{sign}(\theta^T \phi(x) + \theta_0) \\ = \text{sign}(\theta_1 x_1 + \theta_2 x_2 + \theta_3 (x_1^2 + x_2^2) + \theta_0)$$

The linear hyperplane in  $\mathbb{R}^{d_2}$  would become non-linear when projected back down to  $\mathbb{R}^{d_1}$ .

Learning non-linear shape by using linear model <sup>in</sup> higher dimensional linearly separable data.

## Recording from Oct. 7<sup>th</sup>: (part 2)

Mapping from lower dimension to higher can be computationally expensive. Choice of kernel function is important.

Polynomial function:  $x \in \mathbb{R}^n$ , polynomial degree d.

Let  $x \in \mathbb{R}^2$ ;  $d=2$   $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_1 x_2 \end{bmatrix} \in \mathbb{R}^n$$

Why? Relationship can be non-linear.

This allows us to capture those.

$\text{size: } \binom{n+d}{n} \leftarrow \text{size of vector of our polynomial}$

$$\text{If I have } x^2 \Rightarrow j_1=2, j_2=0, j_3=0 \\ x^3 \Rightarrow j_1=0, j_2=2, j_3=0$$

$\rightarrow$  for  $x \in \mathbb{R}^3$ ,  $d=2$ :

Not care about the scalar factor.

$$\phi(x) = \frac{1}{\sqrt{j_1! j_2! \cdots j_n!}} x_1^{j_1} \cdots x_n^{j_n}, \quad j_1 + \cdots + j_n = d$$

$$\binom{n+d}{d} = \binom{4}{2} = 6$$

$$\begin{cases} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \end{cases}$$

Highest power is 2 because  $d=2$

Size of vector grows quickly. Lots of memory.

How does polynomial kernel allow good modeling? / linearly separable.

If  $n$  is fixed,  $d$  varies, what is the dim?  $\frac{(n+d)!}{d! n!} \Rightarrow$  As  $d$  increases  $\rightarrow \infty$ ;

Bottom line is, if  $n$  is fixed & only  $d$  grows,  $\frac{(n+d)!}{d! n!}$  also grows.

Radial basis function:  $\phi(x) = \exp(-\gamma x^2)$

$$\begin{bmatrix} 1 \\ \sqrt{\gamma} x_1 \\ \sqrt{(\gamma)^2 x_1^2} \\ \vdots \end{bmatrix}$$

Let's go back to perceptron algo.

```

 $\Theta \leftarrow 0$ 
for  $k = 1, \dots, T$ 
  for  $i = 1, \dots, n$ 
    if  $y^{(i)} (\Theta^T x^{(i)}) \leq 0$ :
       $\Theta \leftarrow \Theta + y^{(i)} x^{(i)}$ 

```

updates:

1	$y^{(1)} x^{(1)}$	✓	✓
2	✓	✓	✓
3	✓	$y^{(2)} x^{(2)}$	✓
4	$y^{(4)} x^{(4)}$	$y^{(4)} x^{(4)}$	✓

$$\text{Rewrite: } \Theta = \sum_{j=1}^n \alpha_j y^{(j)} x^{(j)}$$

$$\begin{aligned} \Theta &= y^{(1)} x^{(1)} + y^{(2)} x^{(2)} + y^{(3)} x^{(3)} + y^{(4)} x^{(4)} \\ &= y^{(1)} x^{(1)} + y^{(2)} x^{(2)} + 2y^{(3)} x^{(3)} \end{aligned}$$

$$\alpha_1 = 1, \alpha_2 = 0, \alpha_3 = 1, \alpha_4 = 4$$

Just a different way of showing  
the same thing.

⇒ rewrite algo.:

$$\Theta \leftarrow 0, \alpha_i \leftarrow 0 \quad \forall i$$

for  $k = 1, \dots, T$ :

for  $i = 1, \dots, n$

$$\text{if } y^{(i)} \left[ \sum_{j=1}^n \alpha_j y^{(j)} x^{(j)T} x^{(i)} \right] \leq 0$$

$$\alpha_i \leftarrow \alpha_i + 1$$

The two algorithms are equivalent.

If in higher dim,  $\phi(x)$  instead.

Why do we do  
transpose?

Because  $\Theta^T x$

Indices are  
different as well.

$\phi(x)$  can be huge. Use dot product instead.  
Does this improve things?

for  $k = 1, \dots, T$

for  $i = 1, \dots, n$

$$\text{if } y^{(i)} \left[ \sum_{j=1}^n \alpha_j y^{(j)} \phi(x^{(j)T}) \phi(x^{(i)}) \right] \leq 0$$

$$\alpha_i \leftarrow \alpha_i + 1$$

Hopefully this  
makes things  
simpler.

for  $k = 1, \dots, T$

for  $i = 1, \dots, n$

$$\text{if } y^{(i)} (\Theta^T \phi(x^{(i)})) \leq 0$$

$$\Theta \leftarrow \Theta + y^{(i)} \phi(x^{(i)})$$

Only for poly & rbf kernels.

$$\phi(x^{(i)T}) \phi(x^{(i)}) = (1 + x^{(i)T} x^{(i)})^d, \text{ if } \phi \text{ poly}$$

$$\phi(x^{(i)T}) \phi(x^{(i)}) = \exp(-\gamma \|x^{(i)} - x^{(i)}\|_2^2), \text{ if } \phi \text{ rbf.}$$

This is done in the original space!!

Numerical example:  $d=2, n=2$ .  $\phi(x) = [1 \ \sqrt{2}x_1 \ \sqrt{2}x_2 \ x_1 x_2 \ x_1^2 \ x_2^2]^T$

$$\phi(x) = [1 \ \sqrt{2}x_1 \ \sqrt{2}x_2 \ x_1^2 x_2^2 \ x_1^2 \ x_2^2]^T$$

$$\phi(x)^T \phi(x') = 1 + 2x_1 x'_1 + 2x_2 x'_2 + 2x_1 x'_1 x_2 x'_2 + x_1^2 (x'_1)^2 + x_2^2 (x'_2)^2$$

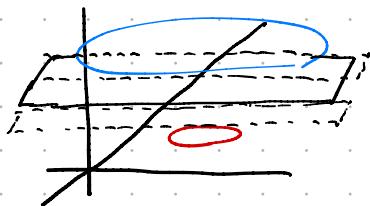
Calculate  $(1 + \mathbf{x}^T \mathbf{x}')^2$

$$= 1 + 1 \cdot x_1 \cdot x'_1 + 2 \cdot x_2 \cdot x'_2 + 2 \cdot x_1 \cdot x'_1 \cdot x_2 \cdot x'_2 + (\mathbf{x} \cdot \mathbf{x}')^2 + (\mathbf{x} \cdot \mathbf{x}')^2$$

They are equivalent. This is the kernel trick. Inner product is the same. Simplest & most convincing approach to kernel trick.

NO SLIDES on kernel SVM.

↳ Exactly the same idea. Kernel SVM considers maximum margin.



Kernel details:

- Additivity: kernel + kernel = kernel
- Scalar multiplication is valid kernel
- Product of kernels is kernel
- Exponentiation of kernel is a kernel

(Kernel Least squares (KLS)): Just add the  $\phi$  parameter.

$$\begin{aligned} \min_{\beta} \|y - \phi(\mathbf{x})\beta\|_2^2 &\longrightarrow (y - \phi(\mathbf{x})\beta)^T(y - \phi(\mathbf{x})\beta) \\ &\quad \frac{\partial}{\partial \beta} (y^T y - y^T \phi(\mathbf{x})\beta - \beta^T \phi^T(\mathbf{x})y - \beta^T \phi^T(\mathbf{x})\phi(\mathbf{x})\beta) \\ &= 0 - \phi^T(\mathbf{x})y - \phi^T(\mathbf{x})y + \phi^T(\mathbf{x})\phi(\mathbf{x})\beta + \phi^T(\mathbf{x})\phi(\mathbf{x})\beta \\ &= -2\phi^T(\mathbf{x})y + 2\phi^T(\mathbf{x})\phi(\mathbf{x})\beta \\ &\longrightarrow \phi^T(\mathbf{x})y = \phi^T(\mathbf{x})\phi(\mathbf{x})\beta \longrightarrow \beta = (\phi^T(\mathbf{x})\phi(\mathbf{x}))^{-1}\phi^T(\mathbf{x})y \end{aligned}$$

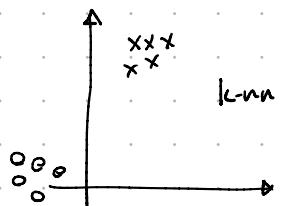
What if we add the regularization term?

$$\min_{\beta} \|y - \phi(\mathbf{x})\beta\|_2^2 + \lambda \|\beta\|_2^2 \longrightarrow \text{same derivation as before.}$$

$$\beta = (\phi^T(\mathbf{x})\phi(\mathbf{x}) + \lambda I)^{-1}\phi^T(\mathbf{x})y$$

- Limitations:
- complex patterns, like time varying
  - overfitting: Easily overfit. 100 samples, fit poly on 99, 100 in overshoot
  - Curse of dimensionality (arbitrarily high dim.)
  - Parameter tuning is non-trivial.

Recording from Oct. 20<sup>th</sup>:

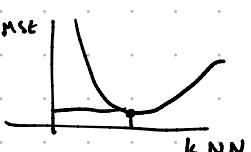


Euclidean distance.

	distance to class	1-NN for $d_q$ : +1 ( $d_1$ )
$d_1$	0.05	+ 1
$d_2$	0.25	- 1
$d_3$	0.13	- 1
$d_4$	0.08	+ 1
		mode operation

What if there is a tie in mode?

Choose  $k=1$ :



- weights for closer points
- Randomly choose
- Prefer closer.

Many different distance metrics.  
Higher dimensions → Euclidean does worse.

k-NN for regression:

weight	height
60	160
65	165
90	
95	175
80	180

$$1\text{-NN: } 175 \text{ or } 165$$

$$2\text{-NN: } \frac{175+165}{2} = 170.$$

arithmetic mean.

for query point  $x_q$ :

$$\hat{y}_q = \frac{1}{k} \sum_{i=1}^k y_i$$

If there are cases where better to use geometric or harmonic mean?

↳ Target values & features have multiplicative nature. Like stocks.

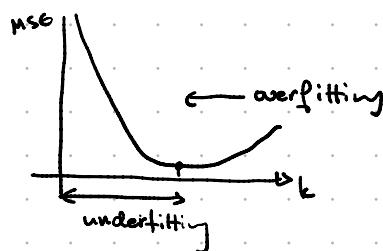
If tie, change k or something else?

Multi-objective optimization.

To improve performance, can use weighted.

$$\hat{y}_q = \frac{\sum_{i=1}^k w_i y_i}{\sum_{i=1}^k w_i}; w_i = \frac{1}{d(x_q, x_i)}$$

Weight of decay of information in high dim. is lower than Euclidean for Manhattan.



Approximate algorithms  
Random algorithms

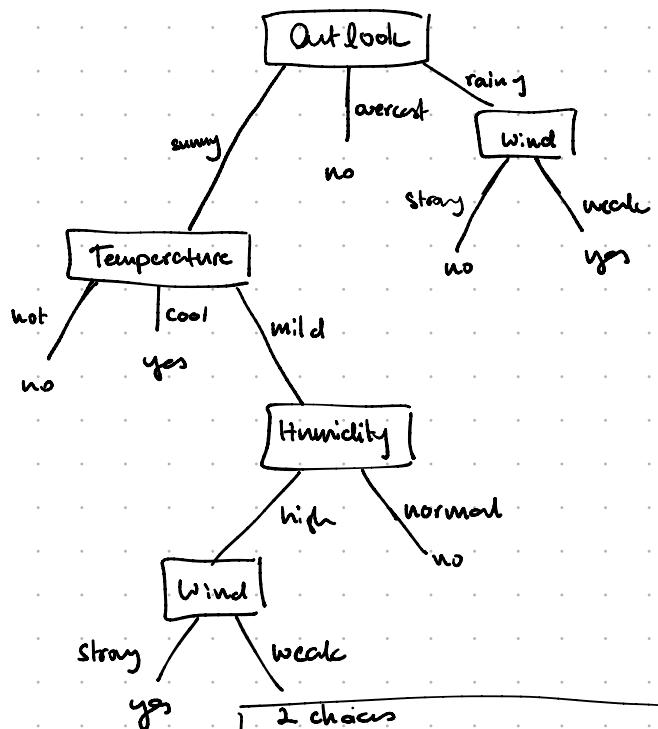
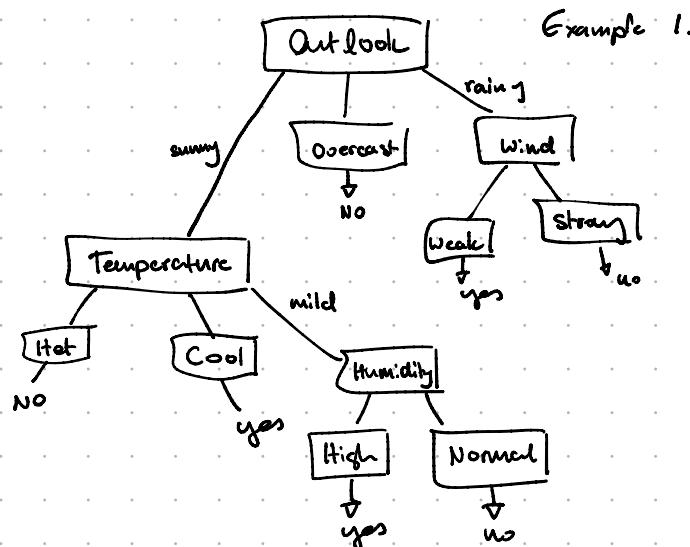
Limitations of k-NN:

- Curse of dimensionality (Use LDA, PCA, etc..)
- Computational complexity (Use KD-trees, ball trees)
- Imbalanced datasets bad
  - ↳ Use weighted voting.

Decision Trees:

↳ Internal nodes split data, leaf nodes classify.  
How do we construct tree? When do we stop?

} He goes through a DT example here.

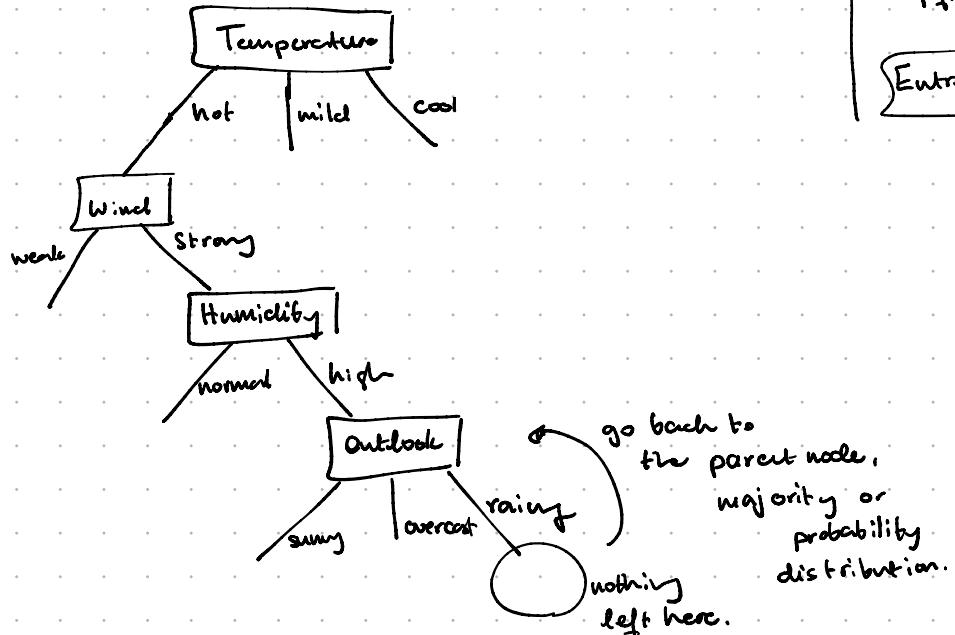


Local calculations in DT: at start, outlook was best. Does not mean optimal by the end.

2 choices  
(no features left)

↳ use majority  
or probability distribution

Example 3: Sunny  $\rightarrow$  hot  $\rightarrow$  high  $\rightarrow$  weak



Kernel decision trees? Not really helpful (or possible)?

Entropy minimization function!

Two interesting cases:

- no features left to split on
- no data left for a feature
  - Go back to parent,
- Majority or probability distribution.

Which feature? Optimal order is computationally expensive. Use greedy approach instead. Very local.

Data → temp  
→ humid  
→ wind  
→ outlook } then keep going. But we don't know what happens next.

Local exploration!

We want to reduce the uncertainty at each step. At root, maximum uncertainty

↳ If all classes equally distributed, the uncertainty is maximized. We want to maximize the entropy. What pd does this? Uniform!!

$$-\sum_{i=1}^k P(c_i) \log_2(P(c_i)) = H. \rightarrow \log_2 \text{ because we are referring to bits.}$$

If two classes (both equally likely), how many bits to represent them? 1 bit. If we start to skew the data in one direction (more likelihood), then we need less bits to represent it.

ID3 type of decision tree that uses information gain.

$$\text{Go back to example: } I\left(\frac{9}{14}, \frac{5}{14}\right) = -\left(\frac{9}{14} \log_2\left(\frac{9}{14}\right) + \frac{5}{14} \log_2\left(\frac{5}{14}\right)\right)$$

$$= 0.94.$$

sunny	Outlook	rainy
y: 2 n: 3	overcast y: 4 n: 0	y: 3 n: 2
$I\left(\frac{2}{5}, \frac{3}{5}\right)$ = 0.971	$I\left(\frac{4}{4}, 0\right) = 0$	$I\left(\frac{3}{5}, \frac{2}{5}\right)$ = 0.941
$S_{14}(0.99)$	0	$S_{14}(0.99)$

$$\Rightarrow 0.94 - 0.694 = 0.249$$

	Info-gain	
Humidity	0.789	0.151
Temperature	0.911	0.029
Wind	0.891	0.0485
Outlook	0.694	0.249

Why choose highest information gain?  
⇒ lowest possible entropy.

So the root node is gonna be the outlook.

We can keep going. Get the new entropy when we go in.

What if there is a tie in information gain? No idea. Diaconis was saying some stuff about ratios.

Another metric we can use is Gini index (CART):  $Gini(b) = 1 - \sum_{i=1}^C p_i^2$   
Where  $C$  is number of classes,  $p_i$  proportion of instances belonging to class  $i$  at node  $b$ .

Unify the framework: Tsallis Entropy. We have  $S_q(X) = \frac{1}{1-q} \left( \sum_{i=1}^n p_i^q - 1 \right)$ ;  $q \in \mathbb{R}$

Choosing an optimal  $q$  is still an open problem.

$$\lim_{q \rightarrow 1} S_q(X) = H(X)$$

Limit growth of tree:

→ pre-post pruning

- max depth,
- IG minimum
- examples in leaf nodes, etc...

fully grow the tree first.

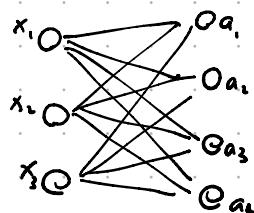
$$S_1(X) = Gini(X)$$

Recording from Oct. 18<sup>th</sup>:

Feed forward Neural Networks (FNNs):

$f: X \rightarrow Y$ ,  $X \in \mathbb{R}^n$ ,  $Y$  labels. Components:

- Layers (depth)
- Width (Neurons in each layer)
- Activation functions
- Loss function

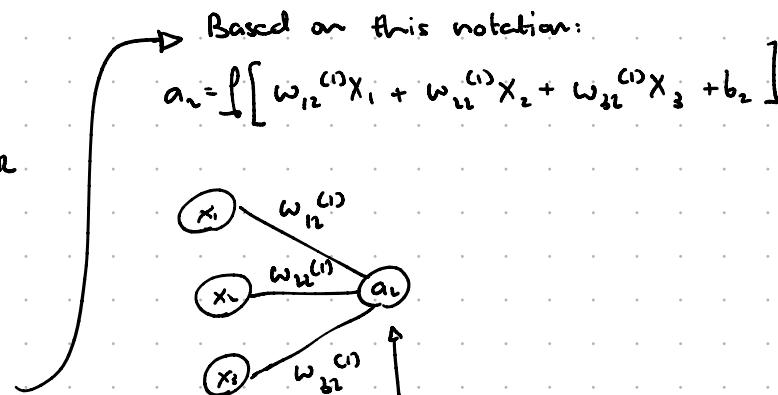


$$a_1 = f(\omega_1 x + b_1); x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathbb{R}^{3 \times 1}, b_1 \in \mathbb{R}$$

activation function.

$$a_1 = \begin{bmatrix} w_{11}^{(1)} x_1 \\ w_{21}^{(1)} x_2 \\ w_{31}^{(1)} x_3 \end{bmatrix} + b_1$$

$f$  can be linear or non-linear.

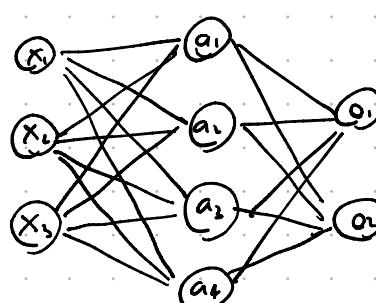


Let's look at a classification example:

$$a_1 = f \left[ w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{31}^{(1)} x_3 \right]$$

Recall one-hot encoding:

	1	2	3	4
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0



$$o_1 = f \left[ w_{11}^{(1)} a_1 + w_{21}^{(1)} a_2 + w_{31}^{(1)} a_3 + w_{41}^{(1)} a_4 + b^{(1)} \right]$$

What would be the activation function here? Softmax. Let  $o_i = f(x_i)$

At this stage, we don't know how to initialize weights, update weights, but assuming weights are there, we can do forward propagation to get outputs  $o_1$  &  $o_2$ .

$$\text{Loss function: } l = -[y \log_2(o_1) + (1-y) \log_2(o_2)] \\ = -[y \log_2(o_1) + (1-y) \log_2(1-o_1)]$$

$$\text{Since } o_1 + o_2 = 1 \Rightarrow o_1 = 1 - o_2$$

What if we have 3 classes? Assuming we have linear activations.  $l = -\sum_{i=1}^3 p_i \log_2(o_i)$

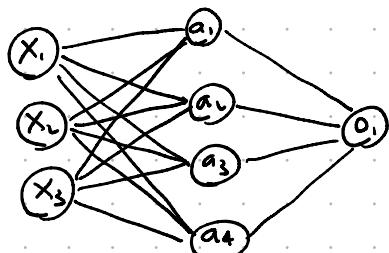
	$p_0$	$p_1$	$p_2$
class 0	1	0	0
class 1	0	1	0
class 2	0	0	1

noiseless labeled data.

$$l = -\sum_{i=1}^3 p_i \log_2(o_i) \quad \text{This is cross-entropy}$$

After training, take argmax to get the prediction.

What if we decide to do regression? Exactly the same. Output are regressors now.



For demonstration:

$$a_1 = f[w_{11}^{(1)}x_1 + w_{21}^{(1)}x_2 + w_{31}^{(1)}x_3 + b_1] \\ o_1 = f[w_{11}^{(2)}a_1 + w_{21}^{(2)}a_2 + w_{31}^{(2)}a_3 + w_{41}^{(2)}a_4 + b_2]$$

$$\text{Loss function: } l = (y - o_1)^2$$

linear function.

Squared error loss.

What if we have two outputs? i.e.  $y \in \mathbb{R}^2 = [y_1 \ y_2]^T \Rightarrow$

$$l = \frac{1}{2} [(y_1 - o_1)^2 + (y_2 - o_2)^2]$$

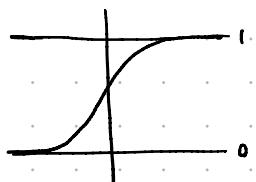
mean squared error.

$$\begin{cases} o_1 = f(w_1^{(2)}a_1 + b_1^{(2)}) \\ o_2 = f(w_2^{(2)}a_1 + b_2^{(2)}) \end{cases}$$

linear function

This only works for linear data. If we want to work w/ non-linear data, use non-linear activation.

↳ Sigmoid:  $f(x) = \frac{1}{1 + e^{-x}}$



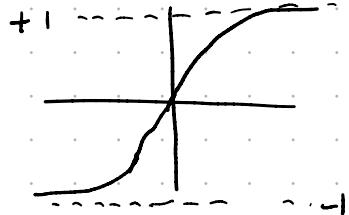
Limitation:

Vanishing gradient at small or large values.

↳ tanh function:  $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

0-centered output (this apparently helps with the convergence)

Larger gradients compared to the sigmoid, but can still vanish for Deep NNs.



↳ ReLU:  $f(x) = \max(0, x)$



Solves the vanishing gradient problem for large values. But now we have exploding.

- Dead neurons issue.

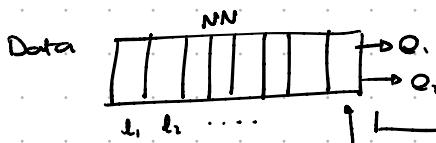
Vanishing gradient: so small that we have no updates.

ReLU  $\iff$  low variance

What did kernels do? Turn dim to high to make data linearly separable. Goal of activation function is to do the same across the depth / layers to separate data in higher dim.

Before: non-linear decision  $\rightarrow$  kernel  $\rightarrow$  linearly separable space (high-dim)  $\rightarrow$  kernel perception

NNs:



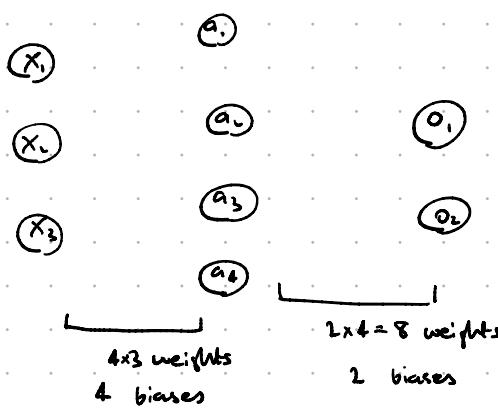
This, we want to be linearly separable. Just the last layer.

Lots of mappings & transformations, but only at the last layer do we care about linear separability.

Activation functions are element-wise activations.

$$f\left(\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}\right) = \begin{bmatrix} f(a_1) \\ f(a_2) \\ f(a_3) \\ f(a_4) \end{bmatrix}$$

softmax layer requires linearly sep. data.



Update weights & biases using the gradients. We turn to the same idea of gradient descent.

Stochastic = random. Probabilistic, not deterministic.

1000 samples  $\rightarrow$  gradient descent ] large number of calculations.  
In 10 batches of 100 samples  $\rightarrow$  batch gd or sgd ] much simpler.

What if we use momentum?

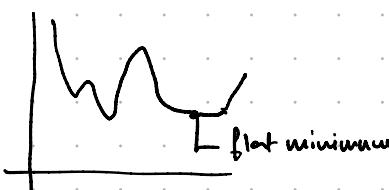
Exponential decay of previous gradients, gives us global information about the shape of the function we are exploring. Why not explore a little more?

Velocity vector.

Adaptive learning rate: slow down as we get to optimal point.

The gradients calculated have higher variance  $\Rightarrow$  allows us to explore the non-convex optimization landscape.

The optimization landscape does not have to be convex. Some minima are hard to get out of. That's why we use momentum / adaptive LR.



fractional derivative?  
To get the update.  
Not local anymore.  
w.r.t. other parameters.

$$w_{ii}^{(t+1)} = w_{ii}^{(t)} + \gamma a_i (y_i - o_i)$$

We have a batch size. What is  $a_i$ ?

$$a_i = f[w_{11}^{(t)} X_1 + w_{21}^{(t)} X_2 + w_{31}^{(t)} X_3]$$

$w_{11}^{(t)}, w_{21}^{(t)}, w_{31}^{(t)}$   $\leftarrow$  random.  $X$  inputs.

We update in back-prop. after we do forward prop.

This is just for one. But we can extend to a lot more.

For final: need to have mathematical maturity to play around with these things.

Describe scenario  $\rightarrow$  write algo., get results.

$$\frac{\partial L}{\partial w_{ii}^{(t)}} = \frac{\partial L}{\partial o_i} \frac{\partial o_i}{\partial w_{ii}^{(t)}}$$

$$w_{ii}^{(t+1)} = w_{ii}^{(t)} - \gamma \frac{\partial L}{\partial w_{ii}^{(t)}}$$

$$\frac{\partial L}{\partial o_i} = -(y_i - o_i)$$

$$\frac{\partial o_i}{\partial w_{ii}^{(t)}} = a_i$$

$$\Rightarrow \frac{\partial L}{\partial w_{ii}^{(t)}} = -a_i (y_i - o_i)$$

Might get backprop., not framed as neural network.

Possible questions:



Weighted least squares (non constant squares. variance)

Derive, get computational complexity, etc...

Data points + labels, but don't say that they are non-linearly separable.

He will not tell us what exactly to do.  
Will just be mechanical.

Another question:  $A \in \mathbb{R}^{m \times n}$

Derive least squares:  $(A^T A)^{-1} \left\{ \begin{array}{l} \text{differences} \\ \text{and } (A^T A)^{-1} \end{array} \right\}$

↓  
if m < n, invertible or not, regularization, etc...

Practice not same as the exam.

Novel, looking for some interesting function, creative solution.

Activation functions separate in different spaces / shapes. Look at the shapes they are representing

↳ Training process: make the lines to come up with something more interesting.

Weight initialization: → Random from distribution. Can cause large or small gradients.

→ He:  $w \sim N(0, \frac{\sqrt{2}}{n_{in}})$

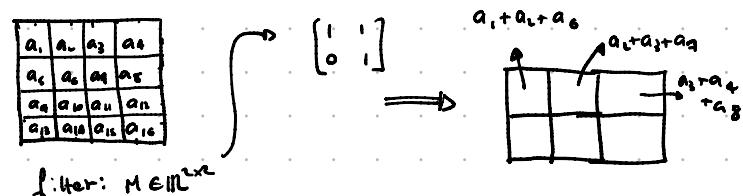
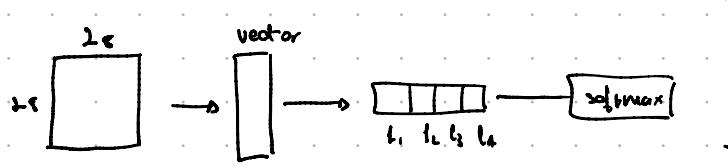
→ Xavier:  $w \sim U\left(\frac{-\sqrt{6}}{\sqrt{n_{in}+n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}\right)$

Depth more important than width for expressive power. Proven.

Shuffling: Exam review example. Shuffle data. Don't order / sequential. The more you shuffle, the better.

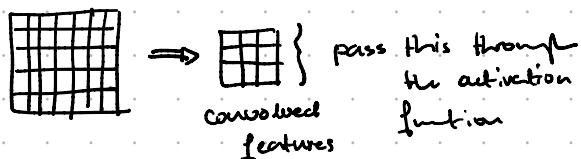
Recording from 3<sup>rd</sup> November:

Convolutional Neural Networks (CNNs)



Element-wise multiplication. Padding. The 1<sup>st</sup> filter would have the same parameters. This extracts some local features.

We can use multiple features to extract different spatial structure. Feature extractor larger before we get into the neural network.



These are some different filters.

Regardless, our goal is the prediction made at the end. NN is end-to-end, from feature extraction to the classification.

Feature engineering → reduces burden of feature extraction from NN.