06-20416 and 06-12412
(Intro to) Neural Computation

13 – Summary

Per Kristian Lehre

Last Lecture

- Convolutional Neural Networks
  - Popular for image recognition, video analysis, natural language processing, etc.
  - Properties
    - sparse interactions, parameter sharing, equivariance
- Convolutional Layer
  - Convolution stage
  - Detector stage / non-linearity (eg Relu)
  - Pooling stage, e.g. max-pooling with downsampling

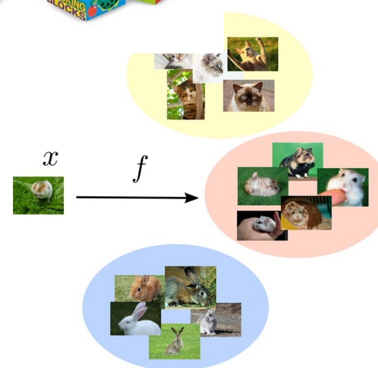
Outline

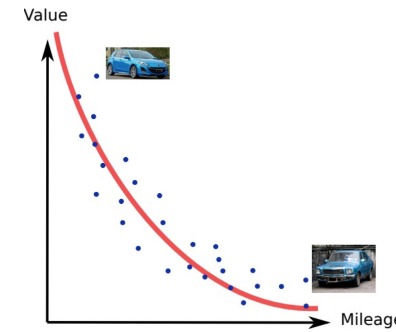
- Summary of each lecture topic
- Example questions
  - Indication of question type
  - Not comprehensive
- Advice on reading material

L1 Introduction

According to Mitchell (1997), machine learning occurs when

performance *P* of algorithm at task *T* improves with experience *E*





Learning task *T*

regression, classification, transcription, translation, synthesis and sampling, ...

$$(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)}) \sim \mathcal{D}$$

Performance measure *P*

depends on learning tasks, e.g., accuracy for classification

$$p(y \mid x) = \frac{\Pr_{(X,Y)\sim\mathcal{D}}(X = x \wedge Y = y)}{\Pr_{(X,Y)\sim\mathcal{D}}(X = x)}$$

Experience *E*

supervised learning, unsupervised learning, reinforcement learning

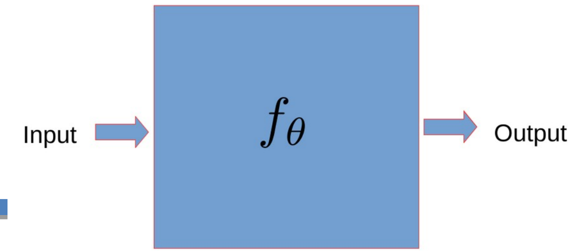
$$x^{(1)}, \dots, x^{(n)} \sim \mathcal{D}$$

$$p(x) = \Pr_{X\sim\mathcal{D}}(X = x)$$

L1 – Example Questions

- Categorise a given list of machine learning problem as supervised, unsupervised, or reinforcement learning
- State Mitchell’s definition of learning

L2 Linear Regression



- Linear regression models
  - model *linear* relationship between input and output
  - Mean square error as cost function
- Optimisation
- Derivatives
  - The chain rule
- Ordinary Least Square (OLS)
- Gradient Descent

Find a "model", i.e., a function

$$f : \mathbb{R}^d \to \mathbb{R}$$

such that on future observations, i.e.,

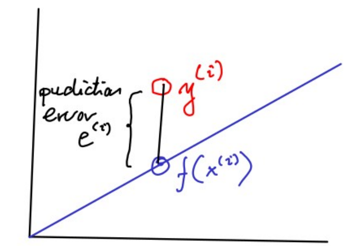
The unknown distribution which the data comes from (see lecture 1).

input

output

$$(X, Y) \sim \mathcal{D}$$

the predicted output $f(X)$ is "close" to the true output Y.

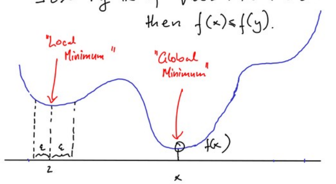


weight parameter

$$J(\omega) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \left(y^{(i)} - \omega x^{(i)} \right)^2$$

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

$\exists \varepsilon > 0 \ \forall y \in \mathbb{R}^d$ if $\forall i \in \{1, \ldots, d\}$ $|x_i - y_i| < \varepsilon$ then $f(x) \leq f(y)$.



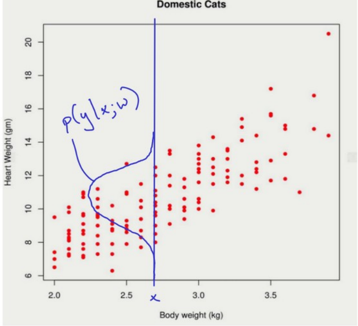
$$\boxed{\omega = \frac{\sum_{i=1}^{n} x^{(i)} y^{(i)}}{\sum_{i=1}^{n} \left(x^{(i)} \right)^2}}$$

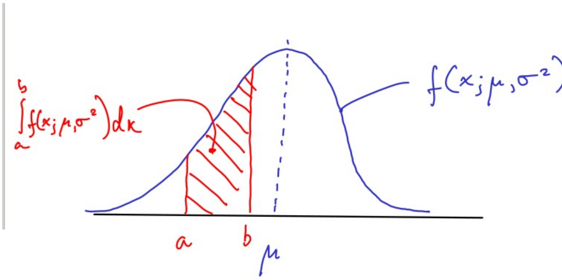
L2 – Example Questions

- Provide examples of machine learning problems for which linear regression is useful
- Derive OLS solution
- Explain difference between local and global optima of cost functions
- Find a local optimal solution of simple cost functions via derivatives

L3 Maximum Likelihood

- Probabilistic models
- Some probabilistic concepts
  - Random variable, density function, normal distribution, joint density function, empirical distribution
- Maximum likelihood
  - Likelihood function and Maximum likelihood estimate
  - Learning via log-likelihood. Example: linear regression





$$\mathcal{L}(\theta;\underbrace{(x^{(1)},y^{(1)}),\dots,(x^{(n)},y^{(n)})}_{\text{training data (fixed)}}) := \prod_{i=1}^{n} \underbrace{P_{model}(y^{(i)} \mid x^{(i)};\theta)}_{\text{conditional density}}$$

model parameter (variable)

$$\theta_{MLE} := \arg\max_{\theta} \mathcal{L}(\theta;(x^{(1)},y^{(1)}),\dots,(x^{(n)},y^{(n)}))$$

$$= \arg\max_{\theta} \prod_{i=1}^{n} P_{model}(y^{(i)} \mid x^{(i)};\theta)$$

$$J(\theta) = \mathbb{E}_{(x,y)\sim\hat{D}} - \log P_{model}(y \mid x;\theta)$$

cost function

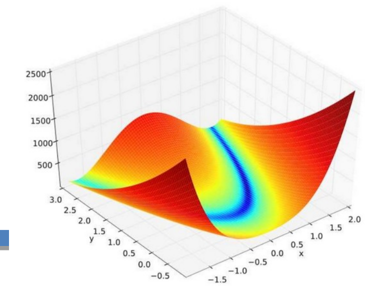
model parameter(s)

empirical distribution of data

L3 – Example Questions

- Compute maximum likelihood estimator (MLE) for simple probabilistic models
- Explain learning via negative log-likelihood

L4 Gradient Descent



- Functions of multiple variables

$$\nabla f := \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_m} \right)$$

- Partial derivatives and the chain rule

$$\underset{v,\, \|v\|=1}{\operatorname{argmax}} \; \nabla_v f(x)$$

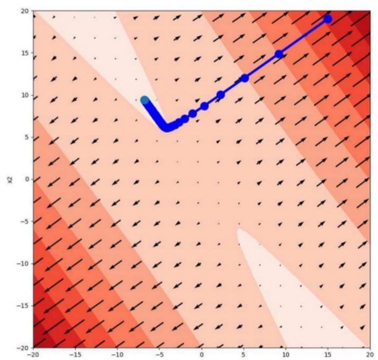
$$= \underset{v,\, \|v\|=1}{\operatorname{argmax}} \; \nabla f(x) \cdot v$$

angle between v and $\nabla f(x)$

$$= \underset{v,\, \|v\|=1}{\operatorname{argmax}} \; \|\nabla f(x)\| \|v\| \cdot \cos\theta$$

$$= \underset{v,\, \|v\|=1}{\operatorname{argmax}} \; \|\nabla f(x)\| \cdot \cos\theta$$

- Gradients
  - Direction of steepest ascent
- Gradient descent



Input: cost function $J : \mathbb{R}^m \to \mathbb{R}$

learning rate $\varepsilon \in \mathbb{R}$, $\varepsilon > 0$

$x \leftarrow$ some initial point in $\mathbb{R}^m$

while termination condition not met {

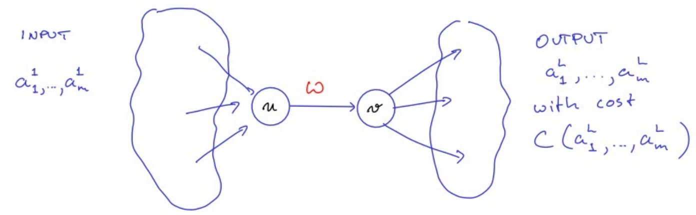
$\quad x \leftarrow x - \varepsilon \cdot \nabla J(x)$

}

L4 – Example Questions

- Compute gradient for simple 2D functions
- State pseudo-code for gradient descent
- Compute a few steps of gradient descent on a simple function given starting point and learning rate

L5 Backpropagation



- Computation graphs
- Feedforward Neural Networks
  - Input nodes and biases
  - Activation functions
  - Input, hidden, and output layers
- Backpropagation algorithm
  - The gradient can be computed using local derivatives.
  - Local derivatives are computed backwards, starting from the output layer

$$\frac{\partial C}{\partial w^l_{jk}} = \frac{\partial C}{\partial z^l_j} \cdot \frac{\partial z^l_j}{\partial w^l_{jk}} = \frac{\partial C}{\partial z^l_j} \cdot a^{l-1}_k$$

$$\delta^l_j = \begin{cases} \phi'(z^L_j) \cdot \frac{\partial C}{\partial a^L_j} & \text{if } l=L \text{ (output layer)} \\ \phi'(z^l_j) \cdot \sum_k \delta^{l+1}_k \cdot w^{l+1}_{kj} & \text{otherwise (hidden layer)} \end{cases}$$

$$\frac{\partial C}{\partial b^l_j} = \frac{\partial C}{\partial z^l_j} \cdot \frac{\partial z^l_j}{\partial b^l_j} = \frac{\partial C}{\partial z^l_j} \cdot 1$$

$$\delta^l_j := \frac{\partial C}{\partial z^l_j}$$

Backpropagation Algorithm

Input: A training example $(x, y) \in \mathbb{R}^m \times \mathbb{R}^{m'}$

1. Set the activation in the input layer
   $a^1 = x$
2. for each $l = 2$ to L, feed forward
   $z^l = w^l a^{l-1} + b^l$
   $a^l = \phi(z^l)$
3. compute local gradient for output layer
   $\delta^L := \nabla_{a^L} C \odot \phi'(z^L)$
4. backpropagate local gradients for hidden layers, i.e.
   for each $l = L-1$ to 2
   $\delta^l := ((w^{l+1})^T \delta^{l+1}) \odot \phi'(z^l)$
5. return the partial derivatives
   $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$
   $\frac{\partial C}{\partial b^l_j} = \delta^l_j$

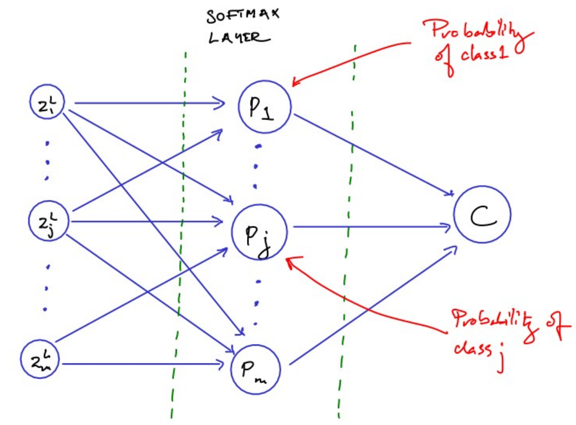
L5 – Example Questions

- What does backpropagation compute?
- What is the local gradient?
- State partial derivative of cost wrt weight as a function of the local gradient
- Derive backpropagation for simple variants of the networks we have discussed

L6 Softmax

- A softmax output layer allows output nodes to be interpreted as probabilities
- The probabilities indicate the likelihood of a class, given the input and the network

$$p_j := \frac{e^{z_j^L}}{Q} \quad \text{where} \quad Q := \sum_{k=1}^{m} e^{z_k^L}$$

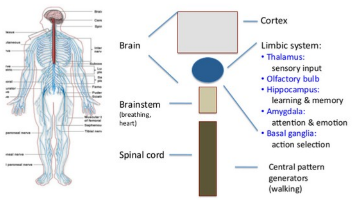


$$\delta_j^L = \frac{\partial C^{(i)}}{\partial z_j^L} = p_j - \delta_{y^{(i)}j}$$

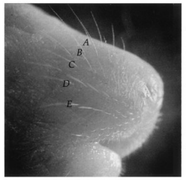
L6 – Example Questions

- When are softmax layers useful?
- Derive local gradient for softmax layer.

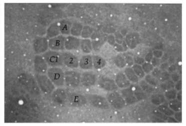
L7 The Biological Brain



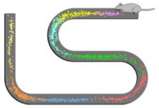
- Organisation of nervous system
- Brain makes model of the world to predict
- Model emerges from brain structure
  - Mouse whiskers and barrels in brain cortex
  - Grid cells create cognitive maps
- Synaesthesia
- Learning and structural change

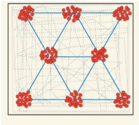


Whiskers in snout

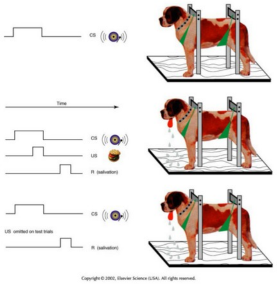


Microscopy image of barrels in brain cortex



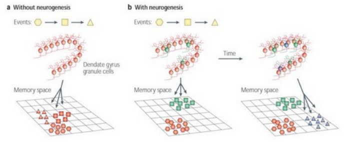


Classical conditioning (Pavlov)



- **Co-active** synapses
- Changes in **synaptic strength**
- **Changes in cell growth** to form new synapses:
  - Synaptic boutons
  - Dendritic spines
- **Generation of new neurons** to encode new memories correlated in time

Encoding of memory: **pattern integration**



No flexibility to encode New memories

Old memories are protected

New neurons born at different Times cluster together and Represent different inputs, **Encoding time into memories**

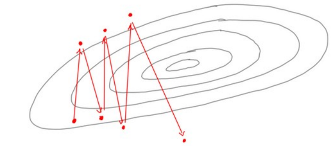
L7 – Example Questions

- Briefly describe the main components of the nervous system
- Give examples of how neural structure reflects a brain’s model of the world

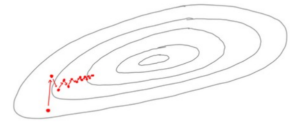
L8 Optimisation

- Learning rate has strong impact on SGD
- Alternatives to SGD
  - SGD with momentum
  - SGD with Nesterov momentum
  - AdaGrad
  - Adam
- Open research problem how to choose appropriate algorithms

Case 1: Too high learning rate ε

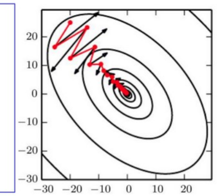


Case 2: Too low learning rate ε



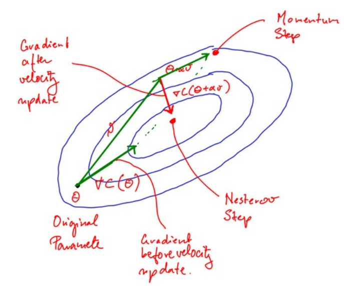
Gradient Descent with Momentum

choose an initial parameter θ
$v = 0$
while termination condition not satisfied {
$v = \alpha v - \varepsilon \nabla_\theta C(\theta)$
$\theta = \theta + v$
}



Nesterov Momentum

choose an initial parameter θ
$v = 0$
while termination condition not satisfied {
$v = \alpha v - \varepsilon \nabla_\theta C(\theta + \alpha v)$
$\theta = \theta + v$
}



Adam (Kingma and Ba, 2014)

choose an initial parameter θ
$r = 0, \; s = 0, \; t = 0$
while termination condition not satisfied {
$t = t + 1$
$g = \nabla_\theta C(\theta)$
$s = \rho_1 s + (1 - \rho_1) g$
$r = \rho_2 r + (1 - \rho_2) g \odot g$
$\hat{s} = \frac{s}{1 - \rho_1^t}, \quad \hat{r} = \frac{r}{1 - \rho_2^t}$
$v = -\varepsilon \cdot \frac{\hat{s}}{\sqrt{\hat{r}} + \delta}$
$\theta = \theta + v$
}

Adagrad (Duchi et al, 2011)

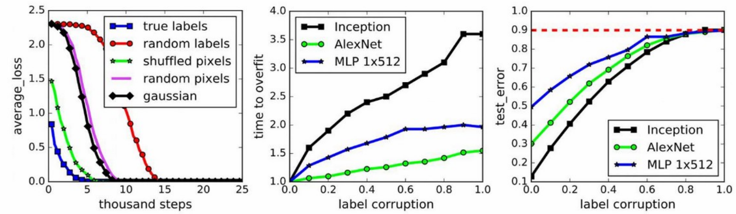
choose an initial parameter θ
$r = 0$
while termination condition not satisfied {
$g = \nabla_\theta C(\theta)$
$r = r + g \odot g$ squared gradient
$v = -\frac{\varepsilon}{\delta + \sqrt{r}} \odot g$ (division and square root applied componentwise)
$\theta = \theta + v$
}

L8 – Example Questions

- What are some problems with plain SGD?
- Write pseudo-code for Nesterov momentum, Adagrad, or Adam
-

L9 Universal Approx. Theorem

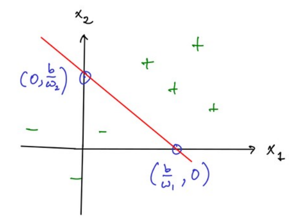
- Universal Approximation
  - Perceptrons (Rosenblatt, 1957)
  - Perceptrons and the XOR function (Minsky & Papert, 1969)
  - Universal Approximation theorem (Cybenko, 1989)
  - Power of depth (Eldan & Shamir, 2016)
- Rethinking generalisation in deep learning



(a) learning curves (b) convergence slowdown (c) generalization error growth

Figure 1: Fitting random labels and random pixels on CIFAR10. (a) shows the training loss of various experiment settings decaying with the training steps. (b) shows the relative convergence time with different label corruption ratio. (c) shows the test error (also the generalization error since training error is 0) under different label corruptions.

$f(x_1, x_2) = \text{sign}(\omega_1 x_1 + \omega_2 x_2 - b)$



Def $\sigma : \mathbb{R} \to \mathbb{R}$ is called discriminatory if

$$\int_{I_n} \sigma(y^T x + \theta)\, d\mu(x) = 0$$

for all $y \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$ implies $\mu = 0$.

Theorem (Cybenko, 1989)

Let σ be any continuous discriminatory function, then for any $f \in C(I_n)$ (i.e., continuous function on $I_n = [0,1]^n$), and any $\varepsilon > 0$, there exists a finite sum on the form

$$G(x) = \sum_{j=1}^{m} \alpha_j \sigma(\omega_j^T x + \theta_j)$$

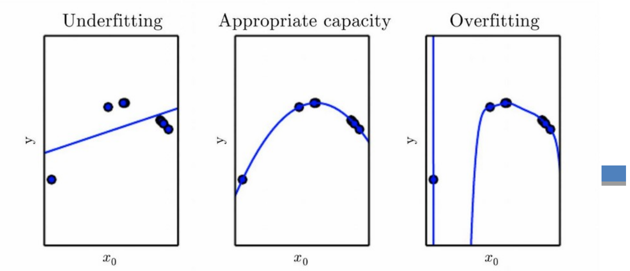
such that

$$|G(x) - f(x)| < \varepsilon \quad \text{for all } x \in I_n.$$

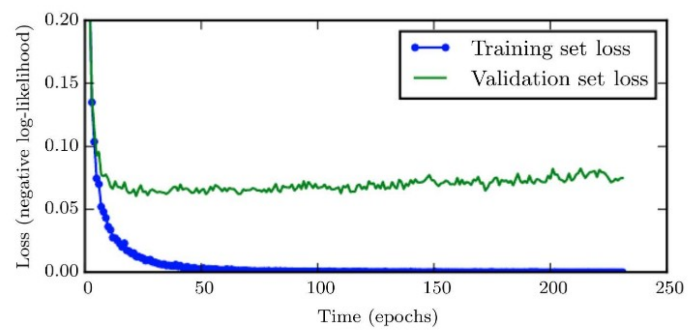
L9 – Example Questions

- Explain why a sigmoid activiation function can be more powerful than the sign function.
- State the universal approximation theorem and explain in plain English what it means
- Explain the power of depth in neural networks

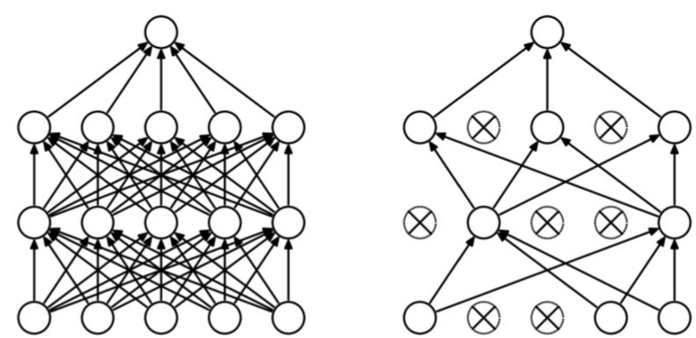
L10 Regularisation



- Model capacity
- Underfitting, overfitting
- Regularisation techniques
  - Data augmentation
  - Early stopping
    - Choose model parameters when validation error was lowest
  - Parameter norm penalties
    - L^2-parameter regularisation
  - Dropout
    - Often used in conjunction with L^2-regularisation.



$$\tilde{C}(\theta; X, y) = C(\theta; X, y) + \alpha\Omega(\theta),$$



(a) Standard Neural Net

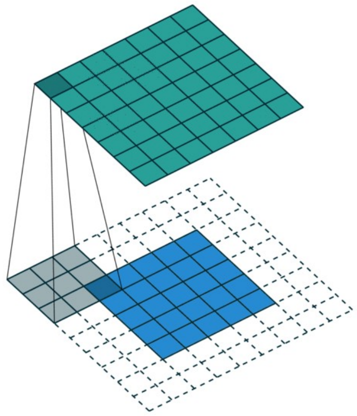
(b) After applying dropout.

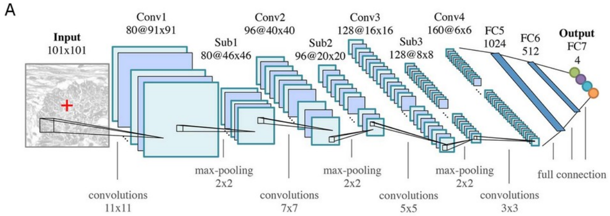
L10 – Example Questions

- Give an example of a model with high capacity, and a model of low capacity.
- What is overfitting?
- What is the aim of regularisation?
- Give four examples of regularisation techniques employed in neural computing

L11-12 Convolutional networks

- Convolution operation
- Convolutional Neural Networks
  - Popular for image recognition, video analysis, natural language processing, etc.
  - Properties
    - sparse interactions, parameter sharing
- Convolutional Layer
  - Convolution stage
  - Detector stage / non-linearity (eg Relu)
  - Pooling stage, e.g. max-pooling with downsampling
- Backpropagation





L11-12 – Example Questions

- Compute the convolution between a vector and a kernel
- State some important properties of CNNs relative to fully connected NNs
- What are typical applications of CNNs?

Compulsory Material

- All material not marked optional on the "Modules" page on Canvas is compulsory reading, e.g.,
  - Lecture notes
  - Reading lists
  - Videos
  - Exercises, lab sheets
- Exam can cover anything covered by compulsory material