AARHUS UNIVERSITY

Data Science 5

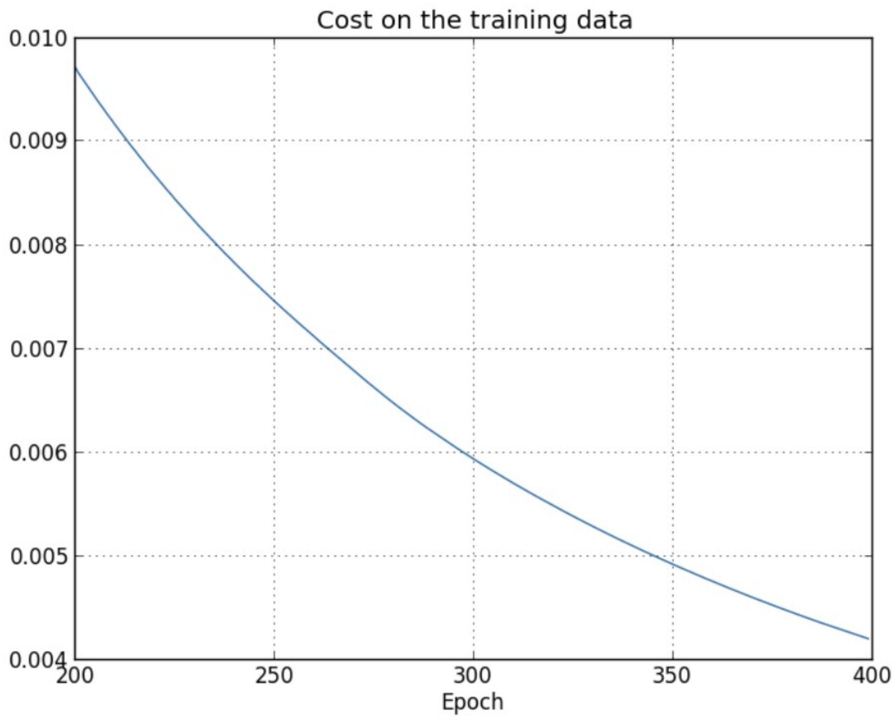
Chris Mathys

Master's Degree Programme in Cognitive Science

Spring 2022

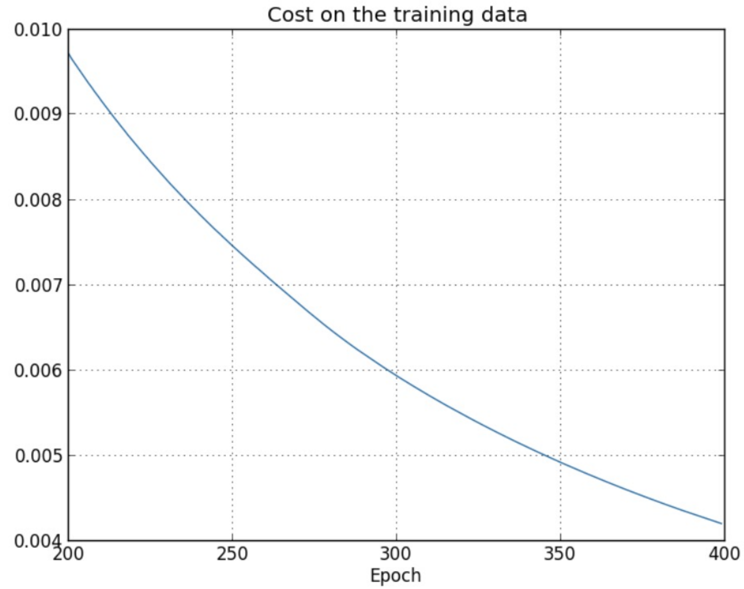
Regularization and overfitting in neural networks

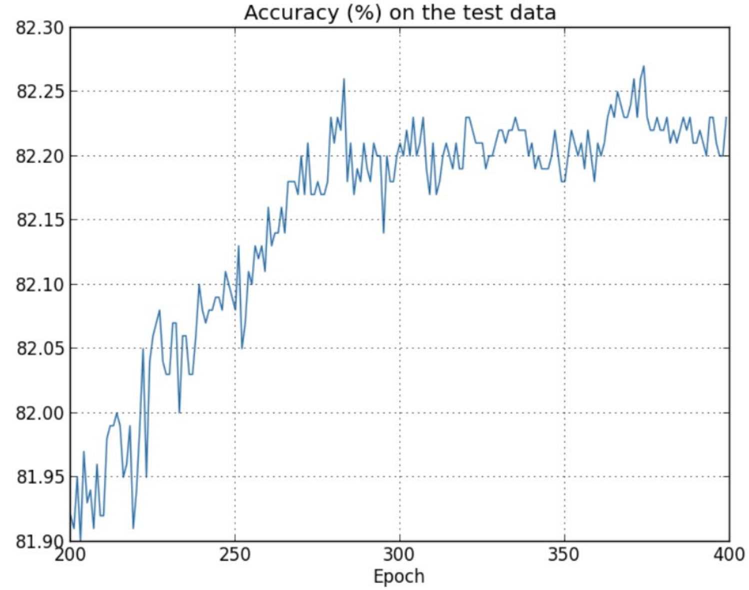
```
>>> import mnist_loader
>>> training_data, validation_data, test_data = \
... mnist_loader.load_data_wrapper()
>>> import network2
>>> net = network2.Network([784, 30, 10], cost=network2.CrossEntropyCost)
>>> net.large_weight_initializer()
>>> net.SGD(training_data[:1000], 400, 10, 0.5, evaluation_data=test_data,
... monitor_evaluation_accuracy=True, monitor_training_cost=True)
```



Source: Michael Nielsen, http://neuralnetworksanddeeplearning.com

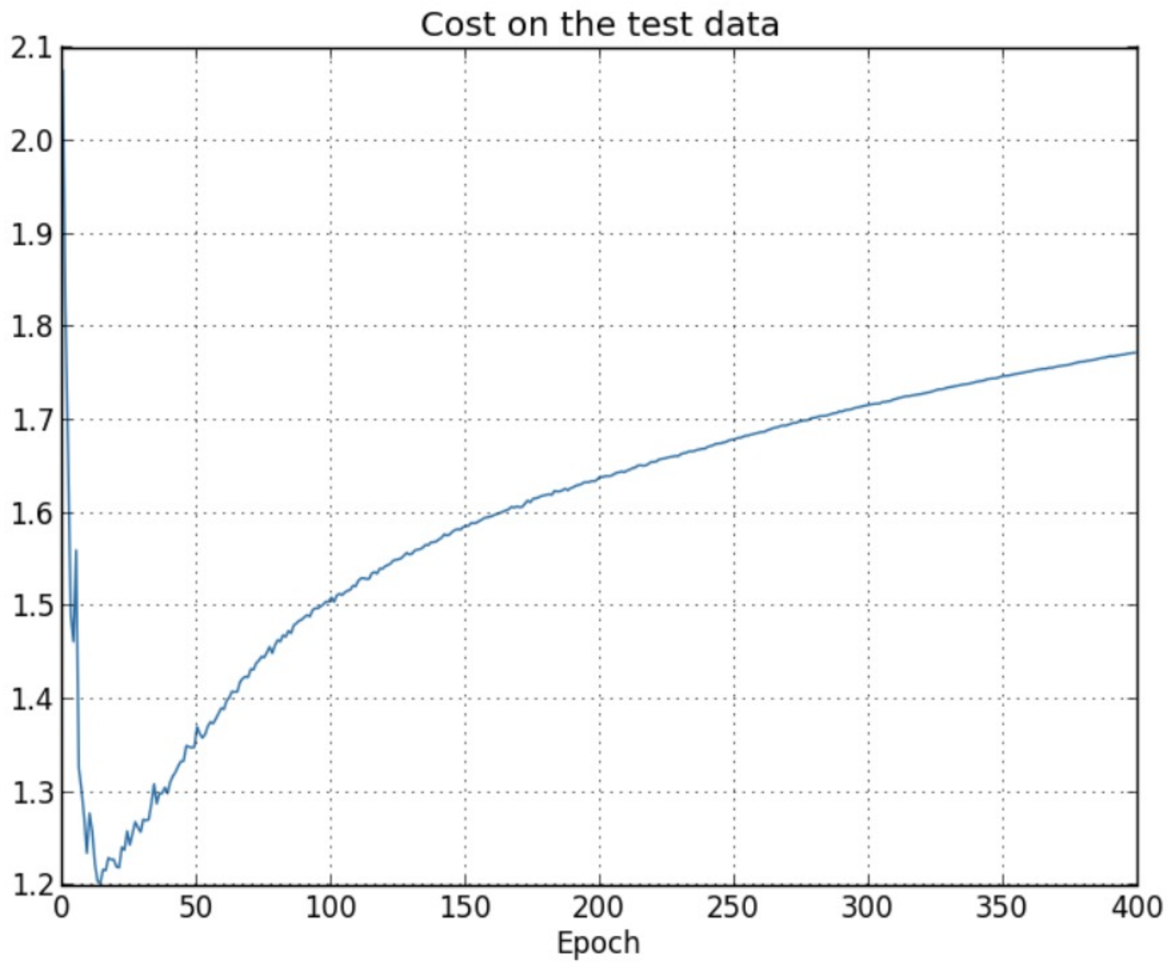
Regularization and overfitting in neural networks





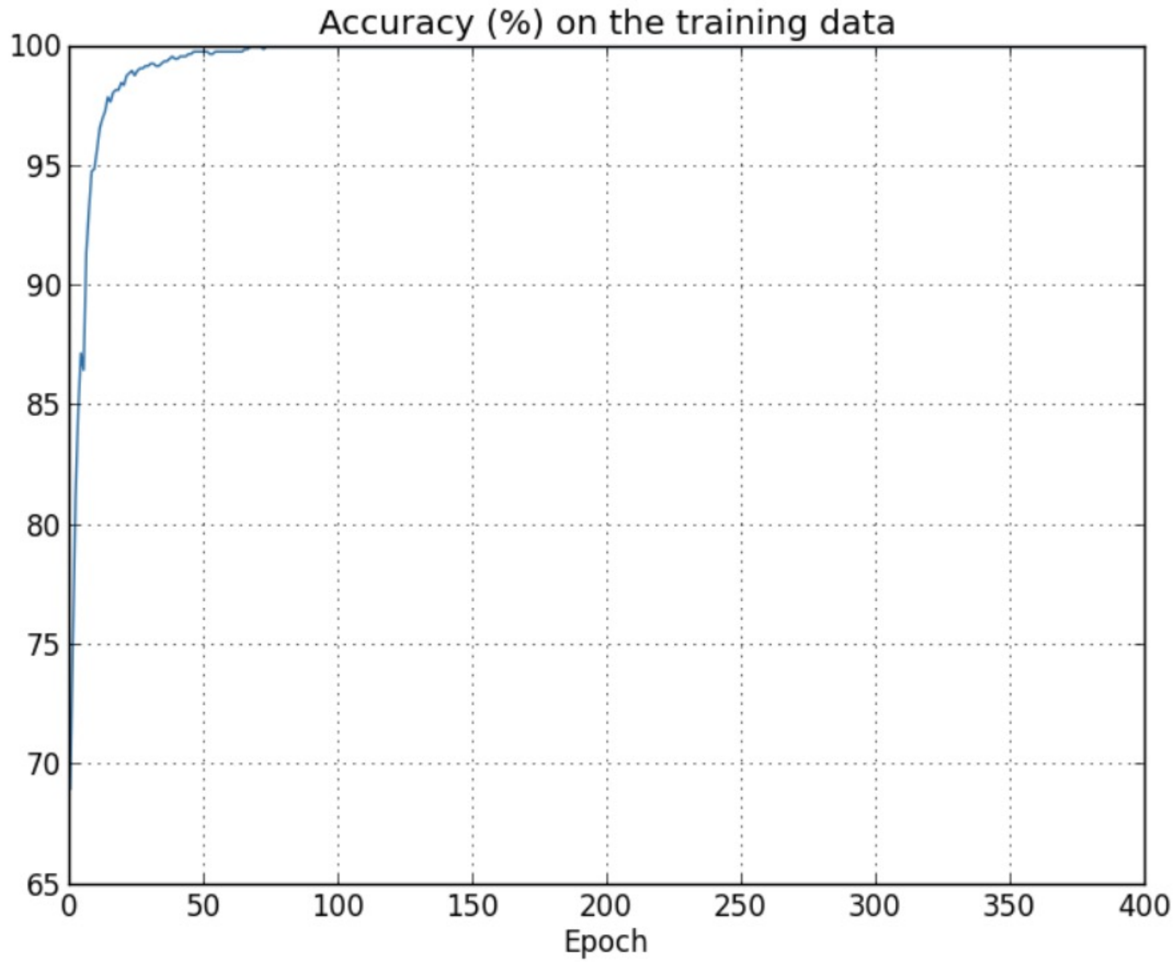
Source: Michael Nielsen, http://neuralnetworksanddeeplearning.com

Regularization and overfitting in neural networks



Source: Michael Nielsen, http://neuralnetworksanddeeplearning.com

Regularization and overfitting in neural networks



Source: Michael Nielsen, http://neuralnetworksanddeeplearning.com

Regularization and overfitting in neural networks

- Use validation data to detect overfitting and *stop early*:

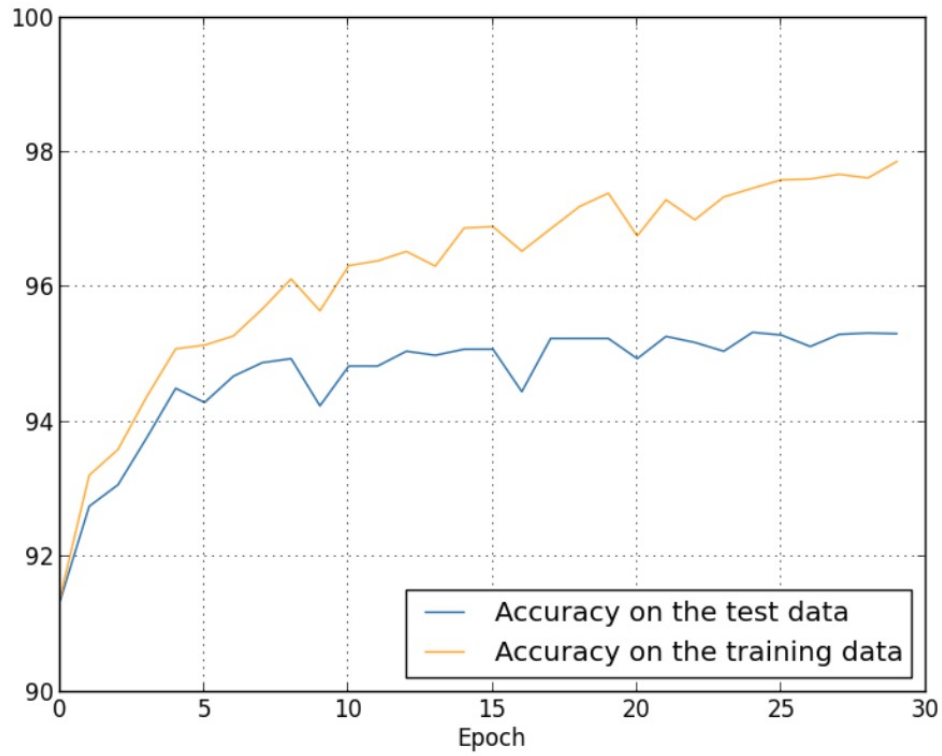
```
>>> import mnist_loader
>>> training_data, validation_data, test_data = \
... mnist_loader.load_data_wrapper()
```

- In general: use validation data to tune hyperparameters
- Why not use the test data?
- Because we'd end up overfitting the hyperparameters to the test data

Source: Michael Nielsen, http://neuralnetworksanddeeplearning.com

Regularization and overfitting in neural networks

- Having more training data reduces overfitting:



Source: Michael Nielsen, http://neuralnetworksanddeeplearning.com

Regularization and overfitting in neural networks

- Regularization *(weight decay):*

$$C = -\frac{1}{n} \sum_{xj} \left[y_j \ln a_j^L + (1 - y_j) \ln(1 - a_j^L) \right] + \frac{\lambda}{2n} \sum_w w^2. \qquad (85)$$

$$C = \frac{1}{2n} \sum_x \|y - a^L\|^2 + \frac{\lambda}{2n} \sum_w w^2. \qquad (86)$$

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2, \qquad (87)$$

Source: Michael Nielsen, http://neuralnetworksanddeeplearning.com

Regularization and overfitting in neural networks

- Regularization *(weight decay):*

$$C = -\frac{1}{n} \sum_{xj} \left[y_j \ln a_j^L + (1 - y_j) \ln(1 - a_j^L) \right] + \frac{\lambda}{2n} \sum_w w^2. \qquad (85)$$

$$C = \frac{1}{2n} \sum_x \|y - a^L\|^2 + \frac{\lambda}{2n} \sum_w w^2. \qquad (86)$$

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2, \qquad (87)$$

Source: Michael Nielsen, http://neuralnetworksanddeeplearning.com

Regularization and overfitting in neural networks

- Regularization *(weight decay):*

The learning rule for the weights becomes:

$$w \rightarrow w - \eta \frac{\partial C_0}{\partial w} - \frac{\eta\lambda}{n} w \tag{91}$$

$$= \left(1 - \frac{\eta\lambda}{n}\right) w - \eta \frac{\partial C_0}{\partial w}. \tag{92}$$

Source: Michael Nielsen, http://neuralnetworksanddeeplearning.com

Regularization and overfitting in neural networks

- **Regularization is equivalent to placing a prior on the weights**
- Maximum likelihood cost function:

$$
\begin{aligned}
\hat{\beta}_{\mathbf{MLE}} &= \arg\min_{\beta} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i,1} + \ldots + \beta_p x_{i,p}))^2 \\
&= \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2
\end{aligned}
$$

Source: Brian Keng, https://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/

Regularization and overfitting in neural networks

- Regularized cost functions:

$$\hat{\beta}_{\mathbf{L1}} = \arg\min_{\beta} \Big(\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i,1} + \ldots + \beta_p x_{i,p}))^2 + \lambda \sum_{j=0}^{p} |\beta_j| \Big)$$

$$\hat{\beta}_{\mathbf{L2}} = \arg\min_{\beta} \Big(\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i,1} + \ldots + \beta_p x_{i,p}))^2 + \lambda \sum_{j=0}^{p} |\beta_j|^2 \Big)$$

Source: Brian Keng, https://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/

Regularization and overfitting in neural networks

- Regularized cost functions:

$$\hat{\beta}_{\mathbf{L1}} = \arg\min_{\beta} \Big(\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i,1} + \ldots + \beta_p x_{i,p}))^2 + \lambda \sum_{j=0}^{p} |\beta_j| \Big)$$

$$\hat{\beta}_{\mathbf{L2}} = \arg\min_{\beta} \Big(\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i,1} + \ldots + \beta_p x_{i,p}))^2 + \lambda \sum_{j=0}^{p} |\beta_j|^2 \Big)$$

Source: Brian Keng, https://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/

Regularization and overfitting in neural networks

- How does the maximum likelihood cost function arise?

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{P(y)}$$

$$\text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$$

$$\begin{aligned}
\mathcal{L}(\beta|\mathbf{y}) &:= P(\mathbf{y}|\beta) \\
&= \prod_{i=1}^{n} P_Y(y_i|\beta, \sigma^2) \\
&= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2}{2\sigma^2}}
\end{aligned}$$

$$\hat{\theta}_{\mathbf{MLE}} = \arg\max_{\theta} \log P(y|\theta)$$

Source: Brian Keng, https://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/

Regularization and overfitting in neural networks

- Maximum-a-posteriori (MAP): regularized because of priors

$$
\begin{aligned}
\hat{\theta}_{\mathbf{MAP}} &= \arg\max_{\theta} P(\theta|y) \\
&= \arg\max_{\theta} \frac{P(y|\theta)P(\theta)}{P(y)} \\
&= \arg\max_{\theta} P(y|\theta)P(\theta) \\
&= \arg\max_{\theta} \log(P(y|\theta)P(\theta)) \\
&= \arg\max_{\theta} \log P(y|\theta) + \log P(\theta)
\end{aligned}
$$

Source: Brian Keng, https://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/

Regularization and overfitting in neural networks

- **Gaussian priors** on coefficients (i.e., weights) lead to ***L2 regularization***:

$$\arg\max_{\beta} \Big[\log \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2}{2\sigma^2}} + \log \prod_{j=0}^{p} \frac{1}{\tau\sqrt{2\pi}} e^{-\frac{\beta_j^2}{2\tau^2}} \Big]$$

$$= \arg\max_{\beta} \Big[-\sum_{i=1}^{n} \frac{(y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2}{2\sigma^2} - \sum_{j=0}^{p} \frac{\beta_j^2}{2\tau^2} \Big]$$

$$= \arg\min_{\beta} \frac{1}{2\sigma^2} \Big[\sum_{i=1}^{n} (y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2 + \frac{\sigma^2}{\tau^2} \sum_{j=0}^{p} \beta_j^2 \Big]$$

$$= \arg\min_{\beta} \Big[\sum_{i=1}^{n} (y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2 + \lambda \sum_{j=0}^{p} \beta_j^2 \Big]$$

Source: Brian Keng, https://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/

Regularization and overfitting in neural networks

- **Laplacean priors** on coefficients (i.e., weights) lead to ***L1 regularization***:

$$Laplace(\mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}$$

$$\arg\max_{\beta} \Big[\log \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2}{2\sigma^2}} + \log \prod_{j=0}^{p} \frac{1}{2b} e^{-\frac{|\beta_j|}{b}} \Big]$$

$$= \arg\max_{\beta} \Big[-\sum_{i=1}^{n} \frac{(y_i - (\beta_0 + \beta_1 x_{i,1} + \ldots + \beta_p x_{i,p}))^2}{2\sigma^2} - \sum_{j=0}^{p} \frac{|\beta_j|}{b} \Big]$$

$$= \arg\min_{\beta} \frac{1}{2\sigma^2} \Big[\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i,1} + \ldots + \beta_p x_{i,p}))^2 + \frac{2\sigma^2}{b} \sum_{j=0}^{p} |\beta_j| \Big]$$

$$= \arg\min_{\beta} \Big[\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i,1} + \ldots + \beta_p x_{i,p}))^2 + \lambda \sum_{j=0}^{p} |\beta_j| \Big]$$

Source: Brian Keng, https://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/