On Neural Network Architecture Based on Concept Lattices

Sergei O. Kuznetsov, Nurtas Makhazhanov(✉), and Maxim Ushakov

Faculty of Computer Science, Department of Data Analysis and Artificial Intelligence, National Research University Higher School of Economics, Kochnovskiy pr. 3, Moscow, Russia
{skuznetsov,nmahaghanov}@hse.ru, mnushakov_1@edu.hse.ru
https://www.hse.ru/

Abstract. Selecting an appropriate network architecture is a crucial problem when looking for a solution based on a neural network. If the number of neurons in network is too high, then it is likely to overfit. Neural networks also suffer from poor interpretability of learning results. In this paper an approach to building neural networks based on concept lattices and on lattices coming from monotone Galois connections is proposed in attempt to overcome the mentioned difficulties.

Keywords: Neural network architecture · Formal concept analysis · Optimal NN architecture · Lattice-based NN

1 Introduction

Neural Networks (NN) is one of the most popular approaches to Machine Learning [15]. Fitting of NN architecture to a dataset under study is a standard procedure that major researchers apply to obtain the best performance results. Matching appropriate architecture is important, because if we take too large network for training, the possibility of getting over-trained model increases. Because of too much redundant connections, NN may learn to select required outcome for each example in the training dataset. So, if the trained network is applied on an object with unobserved set of attributes, it is unable to make a correct classification of it. An over-trained network loses generalizing ability, however if one takes a too small network, one misses the ability to detect possible non-linearities in data. Hence, selecting an appropriate topology and a size of neural network is crucial for its performance. Another problem of NNs is a "black box"-problem. Even if the system with NN shows good classification results, one cannot give an intuitively clear explanation of this.

The first attempts to relate FCA and Neural Networks were done in [13–15]. In [2] authors apply FCA for interpretation of neural codes. In this article we propose an approach to generating neural network architecture based on the covering relation (graph of the diagram) of a lattice coming from antitone Galois

M. Kryszkiewicz et al. (Eds.): ISMIS 2017, LNAI 10352, pp. 653–663, 2017.
DOI: 10.1007/978-3-319-60438-1_64

connections (concept lattice) [6] or monotone Galois connections [1]. The motivation for that is two-fold: First, vertices of such neural networks are related to sets of similar objects with similarity given by their common attributes, so easily interpretable. The edges between vertices are also easily interpretable in terms of concept generality (bottom-up) or conditional probability (top-bottom). Second, many well-known indices of concept quality can be used to select "most interesting" concepts [10], thus reducing the size of the resulting network. In this paper we study two different types of lattices underlying network architecture: standard concept lattices [6] and lattices based on monotone Galois connections [1]. The sets of attributes closed wrt. the latter have "disjunctive meaning" in contrast to "conjunctive meaning" of standard FCA intents (sets of attributes closed wrt. antitone Galois connection). This disjunctive understanding of a set of attributes might fit better the principle of threshold function underlying the standard model of a neuron.

2 Formal Concept Analysis and Hypotheses

First of all, let us recall the basic definitions of Formal Concept Analysis [6]. We consider a set G of objects, a set M of attributes and a binary relation $I \subseteq G \times M$ such that $(g, m) \in I$ iff object g has the attribute m. Such a triple $K = (G, M, I)$ is called a *formal context*. Using the *derivation operators*, defined for $A \subseteq G$, $B \subseteq M$ by

$$A' = \{m \in M \mid gIm \, for \, all \, g \in A\},$$
$$B' = \{g \in G \mid gIm \, for \, all \, m \in B\},$$

we can define a *formal concept* of the context K as a pair (A, B) such that $A \in G$, $B \in M$, $A' = B$, $B' = A$. A is called the *extent* B is called the *intent* of the concept (A, B). These concepts, ordered by

$$(A_1, B_1) \geq (A_2, B_2) \Longleftrightarrow A_1 \supseteq A_2$$

form a complete lattice, called *the concept lattice* of $K = (G, M, I)$.

Next, we recall concept-based hypotheses [4,5,7–9], which originate from JSM-hypotheses [3]. A *target attribute* $\omega \notin M$ partitions the set G of all objects into $c + 1$ subsets, where c is the number of values of the target attribute. The set G_i of those objects that are known to belong to ω_i class. The set $G_\tau \subseteq G$ consists of undetermined examples, i.e., of those objects, for which it is unknown what class they belong to.

Further on we consider the case of binary target and two respective classes, which we call positive and negative. By G_+ we denote the set of objects that are known to have the property of ω and G_-, the set of objects for which it is known that they do *not* have the target attribute ω, and G_τ is the set of objects for which it is not known if they have or do not have target attribute ω. Then, $K_+ = (G_+, M, I_+)$, $K_- = (G_-, M, I_-)$, and $K_\tau = (G_\tau, M, I_\tau)$ are positive, negative, and undefined contexts, respectively. Consider an example in Table 1.

Table 1. Example of formal context. Here we use the following abbreviations: "w" for white, "y" for yellow, "g" for green, "b" for blue, "s" for smooth, "r" for round, "f" for firm, and we use "$\bar{m}$" to denote negation of a binary attribute m.

Context	N	G/M	Color				Form		Firm		Smooth		Target
			w	y	g	b	r	$\bar{r}$	f	$\bar{f}$	s	$\bar{s}$	
K_+	1	Apple		×			×			×	×		×
	2	Grapefruit		×			×			×		×	×
	3	Kiwi			×		×			×		×	×
	4	Plum				×	×			×	×		×
K_-	5	Toy cube			×			×	×		×		
	6	Egg	×				×		×		×		
	7	Tennis ball	×				×			×		×	
K_τ	8	Mango		×			×			×	×		?

3 Neural Network Based on Concept Lattice

In this section we introduce an algorithm for the construction of neural network architecture from the diagram of formal concept lattice. The first step is generating the set of concepts. For the generation of the covering relation of the concept lattice we apply *Add Extent* algorithm, a dual version of *Add Intent* from [12].

The first k upper layers of the lattice can be used as a neural network, where concepts stay for neurons and links between them stay for network connections. By adding extra output layer for classes we obtain an architecture of the neural network. The last hidden layer (with most specific concepts) is connected with the output layer, so that the activations of this layer determine the class of the undefined object. We call the upper part of the lattice diagram with intents of size $\leq k$ the diagram of level k. To attain a less number of classification errors by the network, one can try to select the "best concepts," retain the corresponding vertices in the network and discard the remaining ones. There are many ways to define what are "best concepts". Here we consider two measures, assuming that higher values of them correspond to "better concepts."

– F-value:

$$F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

– Score accounted on Precision and Recall:

$$Score(h_i) = \alpha Precision(h_i) + (1 - \alpha) Recall(h_i),$$

where $\alpha \in [0, 1]$. We will consider different values of α.

– purity of formal concept, which is the maximal part of objects from one class in the concept extent;

To form the set of best concepts H, we start from the empty set, and then iteratively add a concept with the highest score (one of those given above) calculated on examples uncovered by H. This procedure stops when the set H covers all examples from the training set.

In [11,16] authors proposed their approaches for applying Concept Lattices to constructing NN's architecture. Here, we will describe another possible implementation of building interpretable NN using FCA. Previously, we build a diagram of level k, and after that we discard all formal concepts from the last hidden layer that are not considered to be good classifiers. All the neurons above that are not connected with the remaining concepts in the last hidden layer are also removed from the network. We connect the remaining part of the diagram with the input and output layers, obtaining the final architecture of the neural network.

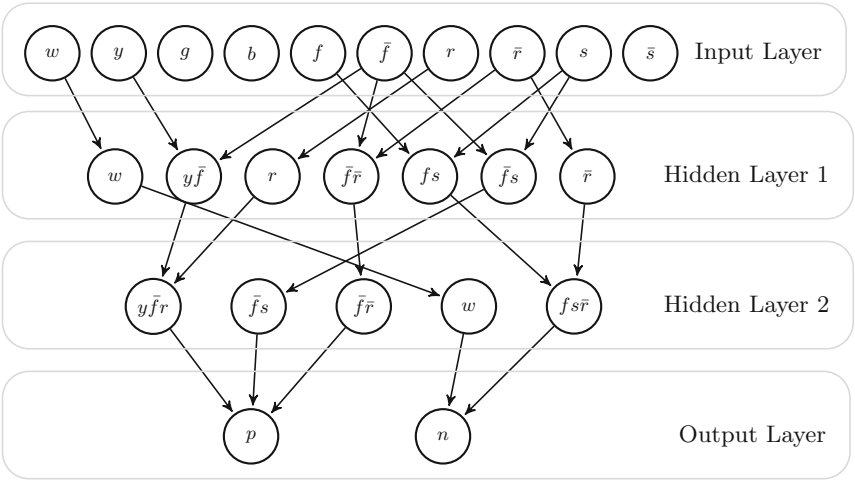

Fig. 1. Architecture of feedforward neural network based on a concept lattice.

Assume that we have a context $K = (G, M, I)$ and diagram D of its concept lattice. Then the network structure can be build as follows:

- The input layer Inp consists of neurons which represent attributes $m \in M$ of the context K.
- Hidden layers Hid_i consist of neurons which represent formal concepts of the context K. The connections between neurons in the hidden layers are the same as in the diagram D (so, two neurons are connected if the corresponding formal concepts are neighbours in the diagram of formal concept lattice). The lower (most specific) concepts are connected to neurons staying for classes. In Fig. 1 you can see the architecture of NN that has been constructed by the method described above.

– The output layer *Out* consists of neurons representing classes. It is connected to the last hidden layer representing the most specific concepts.

In Fig. 1 you can see that in the output layer every neuron is related to each class. There are neurons related to formal concepts in the last hidden layer. We can interpret each weight on the link connecting a concept from the last hidden layer with the output layer as importance of this concept. Here, we want to note that one has to see the proportion of such weights in each particular output neuron. So, these weights do not allow us to compare the concepts related to different classes.

4 Neural Networks Based on Monotone Galois Connections

In this section we describe an approach to constructing neural networks from lattices arising from monotone Galois connections between powersets of objects and attributes. The motivation for this approach comes from the basic properties of standard formal concepts and related closed sets of attributes (intents).

This properties may result in problems when we take intents for nodes of a neural network and covering relation of the concept lattice for connections (arcs) of the neural network. For example, consider a neural network with node C, which have two parents, node A and node B, connected with corresponding weights w_{AC} and w_{BC}. Assume that for some object g neuron A is activated, but neuron B is not. According to neural network model, neuron C also will be activated: $w_{AC} \cdot 1 + w_{BC} \cdot 0 = w_{AC}$ (here we use linear activation function). However, if the given neurons are formal concepts with intents $\hat{A}$, $\hat{B}$ and $\hat{C}$, respectively, then object g does not have some attributes from $\hat{B}$ (as B is not activated), and, as a result, object g does not have some attributes from $\hat{C}$. The latter means that object g does not belong to the extent of concept C, but neuron C is activated for g. This difference may lead to a problem when neurons, which are not supposed to be activated, have significant weights in the constructed neural network.

In this section we will use monotone Galois connection [1,17,18] for building network architecture, which will help us to deal with the problem of 'conjunctivity' of formal concepts, i.e., the property of the concept intent that one has to have all attributes when having the intent.

Consider a formal context (G, M, I), then monotone Galois connections are defined as

$$A' = \{b \mid \nexists a \in G \setminus A \quad such \quad that \quad aIb\},$$
$$B' = \{a \mid \exists b \in B \quad such \quad that \quad aIb\},$$

where $A \subseteq G$ is a set of objects and $B \subseteq M$ is a set of attributes. Further we will call a pair (A, B), where $A' = B$, $B' = A$, a *disjunctive formal concept*.

As in the case of standard formal concepts, disjunctive formal concepts form a lattice with operation $\cup$, defined as $(A_1, B_1) \cup (A_2, B_2) = (A_1 \cup A_2, (B_1 \cup B_2)'')$.

So, we can use the diagram of the concept lattice for building neural network (in the same way as we did it in the previous section).

To show advantages of this approach, consider the previous example, but with disjunctive formal concepts instead of the standard ones. Consider that neuron A is activated, but neuron B is not. So, object g has some attributes from $\hat{A}$, and does not have any attribute from $\hat{B}$. As $\hat{A} \subset \hat{C}$, then object g has some attributes from $\hat{C}$ too. Thus, object g belongs to the extent of $\hat{C}$, so neuron C is activated (we do not have contradictions between disjunctive formal concepts and neural network model).

In order to compute the set of disjunctive formal concepts in a context $K = (G, M, I)$, we need to perform three steps:

1. Compute the complement of the initial relation (replacing all zeros by ones and vice versa);
2. Run *Add Extent* algorithm (a dual version of *Add Intent* from [12]) to compute formal concepts and the covering relation on them;
3. Replace all extents A by $G \setminus A$.

Besides determining the network structure, to train the network one needs to set initial weights on neuron connections. Here we consider several ways of initializing weights.

First, consider the following assignment of weights:

1. $w_1((A_1, B_1), (A_2, B_2)) = \frac{|A_1|}{|A_2|}$ The weights of this kind just give the confidence of the association rule $B_1 \to B_2$.

   Second, one can use similar weights, but with the zero mean:
2. $w_2((A_1, B_1), (A_2, B_2)) = \frac{|A_1|}{|A_2|} - 0.5$.

   The idea of the following kind of weights is quite simple: the more general the concept, the less important a connection from any concept to it:
3. $w_3((A_1, B_1), (A_2, B_2)) = \frac{1}{|A_2|}$.

Here $w((A_1, B_1), (A_2, B_2))$ is the weight of the connection from concept (A_1, B_1) to (A_2, B_2). For edges connecting the last hidden layer and the output layer we initialize weights $w((A, B), i)$ as follows:

$$w((A, B), i) = \frac{|\{a : a \in A, l(a) = i\}|}{|A|},$$

where i is a class, and $l(\cdot)$ is a function that takes any object g to its class $l(g)$.

5 Experiments

We have performed experiments with the following six datasets from the open source UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/index.html) (Table 2):

1. Breast Cancer

2. Credit Card Default
3. Heart Disease
4. Mammographic Mass Data
5. Seismic Bumps.

Table 2. Basic characteristics of the datasets

Dataset	Train sample	Test sample	Number of variables	Number of classes in target variable
Breast cancer	512	57	30	2
CreditCard default	27000	3000	23	2
Heart disease	273	31	13	5
Mammographic mass data	865	96	5	2
Seismic bumps	2326	258	18	2

5.1 Experiments with Different ML Methods

First, we consider the performance of Neural Networks without any prior information about dataset. For all of the five datasets above we have learnt neural networks with different architectures. We apply Adam stochastic optimization strategy because of small number of parameters required for tuning.

As you can see in Fig. 2, neural networks are very unstable wrt. size. For each dataset, it is required to select its own best performing architecture. Even small modifications in the network structure can dramatically affect the results.

Now, let us compare the performance of various ML methods and FCA-based NN algorithms. We constructed a network with one hidden layer and random initial weights. As you can see in Table 3, on Breast Cancer and on Mammographic Mass datasets we achieve same results or that comparable with other algorithms. On Credit Card Default and Seismic Bumps datasets the network based on antitone Galois connections perform better than other ML methods. On the other hand, FCA-based NN models shows worse performance on Heart Disease dataset. At the same time, you can see that the performance of fully-connected NN on this sample is higher than ours. If we consider FCA-based NN as a simple neural network without redundant connections, then we can suppose that it can achieve performance comparable with fully-connected neural networks. The reason why we have obtained worse results may reside in poor selection of the "best concepts". Nevertheless, the main goal of this work was to construct neural network architecture, which can give interpretable results. In future work we would like to attain performance of lattice-based neural networks close to results obtained with simple feed-forward neural networks.

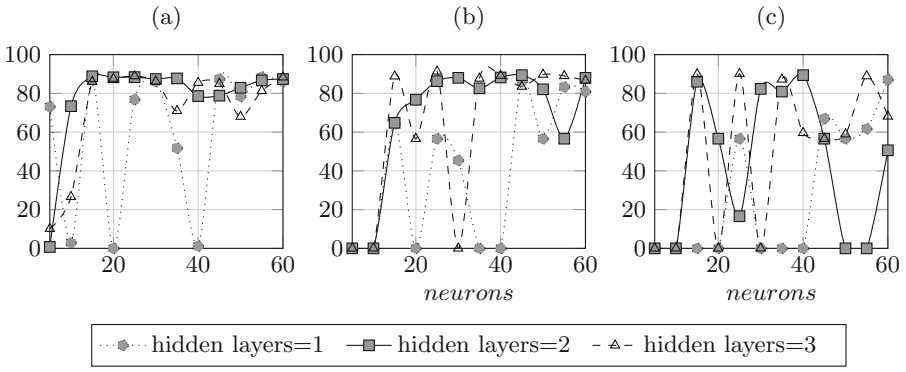

Fig. 2. Performance of NNs with different architectures on breast cancer data. On x-scale number of neurons in each hidden layer, on y-scale F-values. Each NN were learned applying Adam stochastic optimization strategies with initial learning rates equal: (a) 0.01, (b) 0.001, (c) 0.0001.

Table 3. Performance of machine learning methods.

Method	F-value				
	Breast cancer	CreditCard default	Heart disease	Mammogr mass	Seismic bumps
Nearest neighbour	89.0%	27.6%	7.8%	77.2%	5.5%
Decision tree	92.4%	40.5%	34.7%	76.7%	18.5%
Random forest	91.9%	41.9%	28.2%	80.0%	13.2%
Neural network (for the best architecture)[a]	92.9%	37.1%	48.5%	81.7%	12.7%
FCA with AGC based NN	91.8%	50.1%	37.3%	83.4%	23.2%
FCA with MGC based NN	91.8%	34.6%	39.2%	80.5%	10.2%

[a]For Breast Cancer and CC Default datasets: 2 layers, 25 neurons; for Heart Disease: 3 layers, 10 neurons; for Mammographic Mass data: 1 layer, 40 neurons; for Seismic Bumps: 2 layers, 10 neurons

5.2 Comparing Different Methods of Pretraining Neural Network

In Sect. 4 we have proposed three methods of pretraining initial weights of the model based on the properties of disjunctive formal concepts. We have compared

their efficiency on Car Evaluation dataset for different activation functions. To this end, we have constructed the network architecture from the 4-level diagram of disjunctive formal concepts, by initializing weights according to the formulas above and training the model. The table below shows the results of the experiments:

As you can see, the first method of initializing weights for disjunctive concepts $w_1((A_1, B_1), (A_2, B_2))$ shows the worst results, almost like coming from random choice. The reason can be in high values of preinitialized weights, which result in high values of activation function in class nodes, so it is difficult for neural network to fit the data (Table 4).

This model is significantly less accurate than the previous one. The reason of such difference may reside in the monotone nature of disjunctive concepts (the size of extents increases with the size of intent). The top level of the lattice gives very general concepts with big extents and intents, which are not so good for classifying objects.

Table 4. Performance of disjunctive formal concepts with various initial weights and activation functions

Activation function	Predefined weights		
	w_1	w_2	w_3
Sigmoid	32.7%	43.1%	41%
Rectify	33.8%	45.1%	41.9%
Softmax	29.2%	30.9%	31.2%

6 Conclusion

In this paper we have proposed an approach for constructing neural networks based on lattices coming from antitone Galois connections (standard concept lattices) and monotone Galois connections.

Neural networks that are based on concept lattices are very sparse compared to standard fully-connected networks. All neurons in the last hidden layer are related to concepts coming from the dataset. Another advantage of neural networks based on concept lattices is their interpretability, which is very significant in domains like medical decision making and credit scoring. One can both predict the probability of default of applicants, but also implement specific rules and then weight them according to their importance for predicting target variable. Thus, NNs based on concept lattices can be implemented in domains where it is important to explain why objects are assigned to particular classes.

We have presented some results of experiments with different heuristics and parameters of the model. We have calculated performances of simple neural networks with different number of hidden layers and neurons. Also, we have evaluated performance of other learning algorithms in order to compare them

with neural networks based on concept lattices for different datasets. We can conclude that on some datasets we have achieved results comparable with those obtained by other learning approaches. Our further research will be on the study of methods for selecting best concepts for better network performance.

Acknowledgments. The paper was prepared within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) and supported within the framework of a subsidy by the Russian Academic Excellence Project 5-100.

References

1. Düntsch, I., Gediga, G.: Approximation operators in qualitative data analysis. In: Swart, H., Orłowska, E., Schmidt, G., Roubens, M. (eds.) Theory and Applications of Relational Structures as Knowledge Instruments. LNCS, vol. 2929, pp. 214–230. Springer, Heidelberg (2003). doi:10.1007/978-3-540-24615-2_10
2. Endres, D., Foldiak, P.: Interpreting the neural code with formal concept analysis. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems 21, pp. 425–432. MIT Press, Cambridge (2009)
3. Finn, V.K.: Plausible reasoning in systems of JSM type. Itogi Nauki i Tekhniki, Seriya Informatika, Moscow (1991, in Russian)
4. Ganter, B., Kuznetsov, S.O.: Hypotheses and version spaces. In: Ganter, B., De Moor, A., Lex, W. (eds.) ICCS-ConceptStruct 2003. LNCS, vol. 2746, pp. 83–95. Springer, Heidelberg (2003). doi:10.1007/978-3-540-45091-7_6
5. Ganter, B., Kuznetsov, S.O.: Formalizing hypotheses with concepts. In: Ganter, B., Mineau, G.W. (eds.) ICCS-ConceptStruct 2000. LNCS, vol. 1867, pp. 342–356. Springer, Heidelberg (2000). doi:10.1007/10722280_24
6. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)
7. Kuznetsov, S.O.: Mathematical aspects of concept analysis. J. Math. Sci. **80**(2), 1654–1698 (1996)
8. Kuznetsov, S.O.: Machine learning and formal concept analysis. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 287–312. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24651-0_25
9. Kuznetsov, S.O.: Fitting pattern structures to knowledge discovery in big data. In: Cellier, P., Distel, F., Ganter, B. (eds.) ICFCA 2013. LNCS (LNAI), vol. 7880, pp. 254–266. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38317-5_17
10. Kuznetsov, S.O., Makhalova, T.P.: On interestingness measures of formal concepts. Inf. Sci. (2017) (accepted for publication)
11. Nguifo, E.M., Tsopze, N., Tindo, G.: M-CLANN: multiclass concept lattice-based artificial neural network. In: Franco, L., Elizondo, D.A., Jerez, J.M. (eds.) Constructive Neural Networks. Studies in Computational Intelligence, vol. 258, pp. 103–121. Springer, Heidelberg (2009)
12. Merwe, D., Obiedkov, S., Kourie, D.: AddIntent: a new incremental algorithm for constructing concept lattices. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 372–385. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24651-0_31
13. Norris, E.M.: Maximal rectangular relations. In: Karpiński, M. (ed.) FCT 1977. LNCS, vol. 56, pp. 476–481. Springer, Heidelberg (1977). doi:10.1007/3-540-08442-8_118

14. Rudolph, S.: Using FCA for encoding closure operators into neural networks. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS-ConceptStruct 2007. LNCS, vol. 4604, pp. 321–332. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73681-3_24
15. Shavlik, W.J., Towell, G.G.: KBANN: knowledge based artificial neural networks. Artif. Intell. **70**, 119–165 (1994)
16. Tsopze N., Nguifo, E.M., Tindo G., CLANN: concept-lattices-based artificial neural networks. In: Proceedings of 5th International Conference on Convcept Lattices and Applications (CLA 2007), pp. 157–168, Montpellier, France, 24–26 October 2007
17. Vimieiro, R., Moscato, P.: Disclosed: an efficient depth-first, top-down algorithm for mining disjunctive closed itemsets in high-dimensional data. Inf. Sci. **280**, 171–187 (2014)
18. Zhao, L., Zaki, M.J., Ramakrishnan, N.: BLOSOM: a framework for mining arbitrary Boolean expressions. In: KDD 2006, Philadelphia USA (2006)