# Bifurcations of Recurrent Neural Networks in Gradient Descent Learning [*]

Kenji Doya
doya@crayfish.ucsd.edu
Department of Biology
University of California, San Diego
La Jolla, CA 92093-0322, USA

February 1, 1993

**Abstract**

Asymptotic behavior of a recurrent neural network changes qualitatively at certain points in the parameter space, which are known as "bifurcation points". At bifurcation points, the output of a network can change discontinuously with the change of parameters and therefore convergence of gradient descent algorithms is not guaranteed. Furthermore, learning equations used for error gradient estimation can be unstable. However, some kinds of bifurcations are inevitable in training a recurrent network as an automaton or an oscillator. Some of the factors underlying successful training of recurrent networks are investigated, such as choice of initial connections, choice of input patterns, teacher forcing, and truncated learning equations.

## 1 Introduction

Recurrent neural networks are expected to have versatile capabilities for modeling and controlling dynamical systems. From the fact that multi-layer neural networks can approximate arbitrary mappings [13], it is easy to show that recurrent neural networks can model arbitrary dynamical systems [3]. Back-propagation learning schemes for multi-layer feed-forward networks have been successfully applied to a wide range of problems. In contrast, since gradient descent learning algorithms for recurrent networks became popular several years ago [19, 5, 18, 25], not many cases have been reported about their successful application to large scale problems. One reason for this is the large cost for gradient computation [26].

However, another critical issue in training recurrent networks is "bifurcation" of the network dynamics. In general, asymptotic behavior of a nonlinear dynamical system changes qualitatively at certain points in its parameter space [10, 24]. For example, a stable fixed point can change into an unstable fixed point or even disappear with a continuous change of

[*]submitted to: *IEEE Transactions on Neural Networks*

network parameters. This is a very different situation compared to the case of feed-forward networks, in which the output is a continuous function of the weights if each unit has a smooth output function such as a sigmoid.

Although it is formally possible to define error gradient learning schemes for recurrent networks just as in static, feed-forward networks, it is practically very important to understand the special problems that can arise in training recurrent networks [4, 20]. In the following sections, we will investigate what kinds of bifurcations are expected in training recurrent networks, what problems can arise from bifurcations, and what measures we can take to avoid those problems.

## 2 Bifurcations in Recurrent Networks

In this section, we consider the dynamics of continuous-time recurrent networks that are defined by differential equations, for example,

$$\tau \dot{\boldsymbol{x}}(t) = \boldsymbol{x}(t) + S(W\boldsymbol{x}(t)), \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbf{R}^n$ is the state of the network, $W$ is a weight matrix, and $S : \mathbf{R}^n \to \mathbf{R}^n$ is a component-wise squashing function, for example, $S(\boldsymbol{x})_i = 1/(1+\exp[-x_i])$. This model can be converted to a standard discrete-time model

$$\boldsymbol{x}(t+\Delta t) = S(W\boldsymbol{y}(t)) \tag{2}$$

by finite difference approximation. Therefore, the discussion below also applies to discrete-time cases, although there can be some additional stability problems from discretization with large time steps.

### 2.1 A trivial example

First, we consider a very trivial example of a bifurcation of a recurrent network to see what kind of problems can arise with bifurcations. Suppose a recurrent network consisting of a single unit

$$\tau \dot{x}(t) = -x(t) + 1/(1+\exp[-(wx(t)+b)]), \tag{3}$$

with a positive recurrent connection $w$ to itself. The output of this network converges to a fixed point solution that satisfies $x = 1/(1+\exp[-(wx+b)])$. Figure 1 shows the change of the fixed points with the change of the bias $b$ and self connection $w = 5$. The solid and dashed curves represent the stable and unstable fixed points respectively.

Suppose that the desired output is $x_d = 0.6$ and that the bias is initially set as $b = 0$. At first trajectory converges to the upper branch of the stable fixed point and then any gradient descent algorithm decreases $b$ so that the fixed point moves toward $x_d = 0.6$. However, when $b$ becomes smaller than $b_1$, the stable fixed point on the upper branch vanishes and the state jumps to another stable fixed point in the lower branch. Therefore the error is increased abruptly even with a very small change in $b$. The point $b = b_1$ is one-dimensional case of "saddle-node" bifurcation, which is explained below. Note that the gradient of the output with respect to the parameter goes to infinity at the bifurcation point. Suppose an "on-line" version of learning scheme is taken, in which the parameters are updated concurrently with the network dynamics. Then the network state remains on the lower branch of the stable fixed point and $b$ is increased to move the fixed point toward $x_d$, until another saddle-node bifurcation occurs at $b = b_2$. Therefore, the learning process falls into an infinite loop between $b_1$ and $b_2$.

Although this examples uses a very simple network and a trivial task, it is possible that similar problems—sudden increase of the error, explosion of the error gradient, and never converging learning process—are encountered in training large scale recurrent networks on practical tasks. Actually in many simulations, we found that the error curves have several steep jumps [5]. Such increase in error is often considered as a numerical artifact, for example, that the trajectory in the parameter space is bouncing between steep cliffs of the error surface, which is frequently encountered in feedforward case [22]. However, in the case of recurrent networks, the error increase is much larger and can happen even when the learning rate is very small.

## 2.2 Codimension-one bifurcations

Now, let us investigate what kind of bifurcations are likely to occur in training recurrent networks using the theory of bifurcations. Consider a general differential equation system

$$\dot{\boldsymbol{x}}(t) = F(\boldsymbol{x}(t), \boldsymbol{\mu}), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{4}$$

where $\boldsymbol{x}(t) \in \mathbf{R}^n$ and $\boldsymbol{\mu} \in \mathbf{R}^k$ are the state and the parameter of the system and $F(\,)$ is a vector field which is continuous with respect to $\boldsymbol{x}$ and $\boldsymbol{\mu}$. The basic theorem of differential equations assures that this equation has a unique solution $\boldsymbol{x}(t)$ and that the solution curve changes continuous with the change of the initial state $\boldsymbol{x}_0$ and the parameter $\boldsymbol{\mu}$. What characterizes the dynamical system is its "invariant" trajectories, such as fixed points and closed orbits. If we gradually change the parameter $\boldsymbol{\mu}$ of the vector field, we may expect that such points and orbits move continuously. However, that is not always true. Such invariant sets can emerge, collide, disappear, or change stability at certain points in the parameter space, which are called "bifurcation points". The following are examples of the most typical bifurcations.

**saddle-node bifurcation** A saddle point and a node (an attractor or a repellor) emerge in a pair, or coalesce and disappear (Figure 2 (a)).

**Hopf bifurcation** An equilibrium changes it stability and a new limit cycle appears around it (Figure 2 (b) and (c)).

**homoclinic bifurcation** A limit cycle collides with a saddle point and disappears (Figure 2 (d)).

In general, bifurcations occur when the vector field $F$ satisfies some constraints. The above cases can be seen on a $k$-1 dimensional surface in $k$ dimensional parameter space and therefore called "codimension-one" bifurcations. Note that there are many other types of codimension-one bifurcations. For complete theory of bifurcations and their codimensions, see textbooks such as [10, 24].

If we randomly pick one point in the parameter space, it is very unlikely that it is a bifurcation point. However, it we change the parameters continuously by learning, there is a non-negligible chance that we encounter codimension-one bifurcations. The parameter space of a recurrent network is divided into many regions with different qualitative structure of the state space, for example, the regions in which the state space has only one attractor point, one limit cycle, two attractor points, one attractor point and one limit cycle, and so on. Those qualitatively different regions are bordered by bifurcation points. Therefore, some types of bifurcations are inevitable steps in constructing a network with an interesting dynamical behavior. However, when the network goes through a bifurcation, either purposefully or by a mishap, gradient descent algorithms can have problems as shown below.

## 2.3 Bifurcations in learning multiple attractors

One of the popular tasks for recurrent networks is to construct an automaton from given input-output sequences. If a network is to have non-transient memory, it must be implemented as isolated attractors in the state space, such as fixed points [12] and limit cycles [6].

In many cases, the connection weights are initialized with small random values. Then, the state space has only one fixed point, which is a global attractor. In order to increase the number of attractors, the most typical bifurcations is *saddle-node bifurcation*, in which a pair of a saddle and an attractor point are generated (Figure 2 (a)). However, as we saw in the above example, the output of the network can change discontinuously with saddle-node bifurcation and it can make the use of gradient descent algorithms inappropriate.

Learning an automaton involves basically two kinds of tasks. One is to construct an appropriate state transition mechanism and another is to assign required outputs to each state. However, until the network has a required state transition mechanisms, only the average value of the target outputs tend to be learned.

## 2.4 Bifurcations in learning oscillation patterns

Another popular theme of learning in recurrent networks is oscillation. The most typical bifurcation that makes a limit cycle from a fixed point is *Hopf bifurcation*. There are two sub-types of Hopf bifurcation depending on the higher order terms of the vector field. In a *supercritical* Hopf bifurcation (Figure 2 (b)), an attractor point changes into an unstable fixed points and a stable limit cycle appears around it. In this case, a damped oscillator changes into an autonomous oscillator. However, in *subcritical* Hopf bifurcation (Figure 2 (c)), the fixed point point becomes unstable but there is no stable limit cycle in the neighborhood.

Another possible bifurcation in training a limit cycle is *homoclinic bifurcation* (Figure 2 (d)). Suppose we have successfully got a small limit cycle. The next task is to grow the limit cycle into a desired size and shape. However, if there is a saddle point in the neighborhood, the limit cycle may collide with it and disappear.

In both cases, the network state jumps to some other attractor in the state space, as in the case of saddle-node bifurcation. After such a jump to a new attractor, the behavior of the network becomes so different from the previous stage that the network may have to start the learning again.

## 2.5 Instability of learning equations

In this section, we investigate the stability of typical learning equations for error gradient calculation. Most of the learning algorithms for recurrent networks are based on the "variation equation" of the network dynamics. If this equation becomes unstable, which is the case at bifurcation points, the estimate of gradient becomes very large. This causes a large jump in the parameter space if the gradient descent algorithm is used with a fixed learning rate.

Let us represent a network dynamics as

$$\dot{\boldsymbol{x}}(t) = F(\boldsymbol{x}(t)). \tag{5}$$

We want to estimate how the solution $x(t)$ changes when we add a small perturbation $\boldsymbol{\epsilon}(t)$ to the network as

$$\dot{\tilde{\boldsymbol{x}}}(t) = F(\tilde{\boldsymbol{x}}(t)) + \boldsymbol{\epsilon}(t).$$

The variation $\boldsymbol{p}(t) = \tilde{\boldsymbol{x}}(t) - \boldsymbol{x}(t)$ can be estimated by the linear equation

$$\dot{\boldsymbol{p}}(t) = A(t)\boldsymbol{p}(t) + \boldsymbol{\epsilon}(t), \tag{6}$$

where $A(t) = \frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}=\boldsymbol{x}(t)}$ is a Jacobian matrix. In order to estimate the effect of a small change in a parameter, for example, a weight $w$, we use the perturbation $\boldsymbol{\epsilon}(t) = \frac{\partial F(\boldsymbol{x})}{\partial w}$. In "on-line" or "real-time" versions of learning algorithms, the variation equation (6) for each of the parameters are integrated along with the network dynamics (5) [25, 5].

Another version of the learning scheme uses the solution of an "adjoint system" of the variation equation (6)

$$\dot{\boldsymbol{q}}(t) = -A^T(t)\boldsymbol{q}(t) - \boldsymbol{\delta}(t), \tag{7}$$

where $\boldsymbol{\delta}(t)$ is actual error in the output units and $\boldsymbol{q}(t)$ represents equivalent errors in all the units including hidden units. Note that the stability of the adjoint system (7) is the same as (6) if it is solved backward in time, as in "back-propagation through time" algorithms [23, 19, 18].

The variation equation (6) is stable at a stable fixed point. However, when the fixed point has a bifurcation, the eigenvalue of Jacobian matrix $A$ has a zero real part and therefore the asymptotic stability of the learning equation (6) is not guaranteed.

If the network (5) has a non stationary solution $\boldsymbol{x}(t)$, for example, a limit cycle, then the corresponding variation equation (6) is not asymptotically stable [11, 21]. We can show that $\boldsymbol{p}_a(t) = a\dot{\boldsymbol{x}}(t)$ for any $a \in \mathbf{R}$ is a solution of the homogeneous equation $\dot{\boldsymbol{p}}(t) = A(t)\boldsymbol{p}(t)$ as below.

$$\dot{\boldsymbol{p}}_a(t) = a\frac{d}{dt}F(\boldsymbol{x}(t)) = a\frac{\partial F}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} = aA(t)\dot{\boldsymbol{x}} = A(t)\boldsymbol{p}_a(t).$$

This means that (6) can have a solution with arbitrarily large amplitude, depending on the initial state and the input $\boldsymbol{\epsilon}(t)$.

In feedforward networks, the error gradient is bounded by

$$(\text{output error}) \times \{(\text{gain of output function}) \times (\text{maximum weight})\}^{(\text{number of layers})}.$$

In recurrent networks, the estimate of error gradient can grow exponentially in time when the learning equation becomes unstable. If a gradient descent algorithm with a fixed learning rate is used, a very large error gradient causes a long jump in the parameter space. It can even lead to a numerical overflow. Even if those are not the case, gradient descent does not work efficiently when the size of gradient varies so much in different directions.

## 3 Conditions underlying successful training

In spite of the possible problems considered in the previous section, there are successful cases in which recurrent networks have been trained to have fairly complex dynamics. In this section, we investigate factors or techniques that are used in those successful cases.

### 3.1 Preprogramming of the state space

When we have an *a priori* knowledge about the required qualitative structure of the state space, for example, the number of attractors, it is possible to preprogram it by the initial connection weights. For example, if we want to train a network to have multiple limit cycle attractors, it is helpful to set the initial connection weights to have multiple attractor points using autocorrelation matrix [6]. In such a case, problems with saddle-node bifurcations are avoided and changing fixed points into limit cycles is not difficult if "teacher forcing" is used, as described later.

Actually in many cases, recurrent networks are trained as an adaptive nonlinear filters [1, 17, 15]. In these cases, the main issue of learning is to fit the transient response of the network to the desired curves and the network should have a single attractor point. This can be realized without any bifurcations from weak random initial connections. Convergence of gradient descent learning is theoretically guaranteed in this case [16].

### 3.2 Choice of input sequences

In the previous section, we considered the bifurcations of autonomous networks. However, when a network is trained as an automaton, time-dependent inputs play an important role. If the external inputs can successfully control the state of the network and modulate the bifurcation boundary, it is possible to train a network to have multiple attractor points without apparently having the problems from "saddle-node" bifurcations.

It has been reported that recurrent networks could be successfully trained to predict or to discriminate fairly long and complex sequences generated by formal grammars [8, 9]. However in these cases, the choice of training patterns are critical for successful training. At first, only a small part of the training set [9] or sequences with limited length [8] were used for training. As the network become capable of processing them, more examples or longer sequences are incrementally used for training. This suggests that a recurrent network with external inputs can create new attractors, but only a limited number at any one time.

### 3.3 Teacher forcing

In training autonomous behaviors of recurrent networks, such as limit cycles, it has been reported that a technique called "teacher forcing" is required [20, 5, 25]. In teacher forcing, the actual outputs $x_i(t)$ of the output units are replaced by the desired output $x_{di}(t)$ when they appear in the right-hand side of the equations (1) or (2). If the actual outputs $x_i(t)$ becomes very close to $x_{di}(t)$ by learning, we can expect that the autonomous system will have a similar trajectory when teacher forcing is cut off.

In training an associative memory network using gradient descent, teacher forcing was required to avoid trajectories from different initial states to converge to the same single attractor [20]. It also avoids the problems from saddle-node bifurcations. Teacher forcing is also necessary in training a network as an oscillatory for several reasons.

1. **Synchronization with the teacher**: Suppose the network has nearly learned the desired waveform and is running autonomously. Unless the frequency of the network oscillation is exactly equal to that of the teacher signal and the initial phase is exactly matched, the phases of the network output and the teacher signal are not kept equal. This makes an apparent large error and destroys the nearly desired structure of the network.

2. **Hopf bifurcations**: A fixed point of an autonomous system becomes a limit cycle when it is teacher forced. Therefore, a desired oscillation pattern can be achieved by just changing the shape of the limit cycle, without having Hopf bifurcations.

3. **Stability of learning equations**: As mentioned in section 2.5, the variation equation of an autonomous limit-cycle is not asymptotically stable. This can be avoided by making the system into a forced system.

### 3.4 Truncated learning equations

One simple way to avoid the instability of the learning equations is to use non-recurrent learning equations. Actually, learning rules in which recurrent connections are neglected in error gradient calculation have been successfully applied to sequence generation [5, 14] and sequence prediction tasks [2, 7].

In the algorithm called "truncated back-propagation through time" [26], the adjoint equation (7) is calculated only for some limited steps backward in time. Although this algorithm was developed mainly to reduce the amount of computation, it also avoids the problem of instability of the learning equations.

## 4 Conclusions

Gradient descent learning in recurrent networks is not as reliable as in feed-forward networks. In order to achieve successful learning, many factors other than gradient calculation need to be considered. Insights from the theories of dynamical systems and bifurcations are very important for this purpose.

## Acknowledgments

The author thanks Steve Biafore, Mike Casey, Gary Cottrell, Peter Rowat, Fu-Sheng Tsung, and Shuji Yoshizawa for their helpful comments. Part of this study was supported by ONR grant N00014-91-J-1720.

## References

[1] T. J. Anastasio. Neural network models of velocity storage in the horizontal vestibulo-ocular reflex. *Biological Cybernetics*, 64:187–196, 1991.

[2] A. Cleeremans, D. Servan-Schreiber, and J. L. McClelland. Finite state automata and simple recurrent networks. *Neural Computation*, 1:372–381, 1989.

[3] K. Doya. Universality of fully connected recurrent neural networks. (submitted).

[4] K. Doya. Bifurcations in the learning of recurrent neural networks. *Proceedings of 1992 IEEE International Symposium on Circuits and Systems*, 6:2777–2780, 1992.

[5] K. Doya and S. Yoshizawa. Adaptive neural oscillator using continuous-time back-propagation learning. *Neural Networks*, 2:375–386, 1989.

[6] K. Doya and S. Yoshizawa. Memorizing oscillatory patterns in the analog neuron network. *Proceedings of IJCNN 1989 in Washington D.C.*, I:27–32, 1989.

[7] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[8] J. L. Elman. Incremental learning, or the importance of starting small. Center for Research in Language, Technical Report 9101, University of California, San Diego, La Jolla, CA, 1991.

[9] C. L. Giles, C. B. Miller, D. Cheng, H. H. Chen, G. Z. Sun, and Y. C. Lee. Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4:393–405, 1992.

[10] J. Guckenheimer and P. Holmes. *Nonlinear Oscillation, Dynamical Systems, and Bifurcations of Vector Fields.* Springer-Verlag, New York, NY, 1983.

[11] P. Hartman. *Ordinary Differential Equations.* Birkhäuser., 1982.

[12] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of National Academy of Science*, 79, 1982.

[13] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.

[14] M. I. Jordan. Serial order: A parallel distributed processing approach. In J. L. Elman and D. E. Rumelhart, editors, *Advances in Connectionist Theory: Speech.* Erlbaum, 1989.

[15] R. J. Krauzlis and S. G. Lisberger. Visual motion commands for pursuit eye movements in the cerebellum. *Science*, 253:568–571, 1991.

[16] C. Kuan, K. Hornik, and H. White. Some convergence results for learning in recurrent neural networks. Discussion Paper 90-42, UCSD Department of Economics, 1990.

[17] S. R. Lockery, Y. Fang, and T. J. Sejnowski. A dynamic neural network model of sensorimotor transformation in the leech. *Neural Computation*, 2:274–282, 1990.

[18] B. A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1:263–269, 1989.

[19] F. J. Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59:2229–2232, 1987.

[20] F. J. Pineda. Dynamics and architecture for neural computation. *Journal of Complexity*, 4:216–245, 1988.

[21] L. S. Pontriagin. *Ordinary Differential Equations.* Addison-Wesley, 1962. (translated from Russian by L. Kacinskas and W. B. Counts).

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In J. L. McClelland, D. E. Rumelhart, and the PDP research group, editors, *Parallel Distributed Processing*, volume 1. MIT Press, Cambridge, MA, 1986.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[24] S. Wiggins. *Introduction to applied nonlinear dynamical systems and chaos.* Springer-Verlag, New York, NY, 1990.

[25] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

[26] R. J. Williams and D. Zipser. Gradient based learning algorithms for recurrent connectionist networks. College of Computer Science Technical Report NU-CCS-90-9,, Northeastern University, 1990.

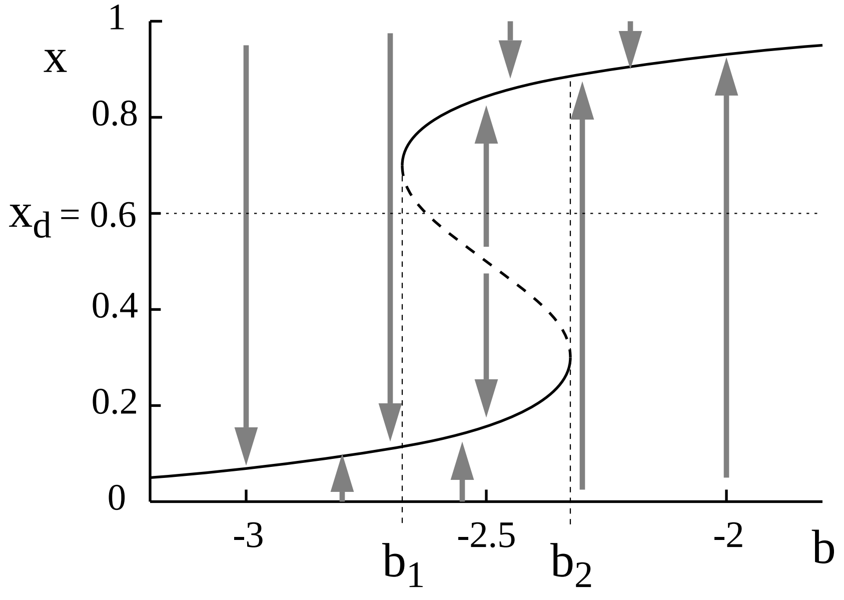

Figure 1: The bifurcation diagram of a self-recurrent network ($w = 5$).

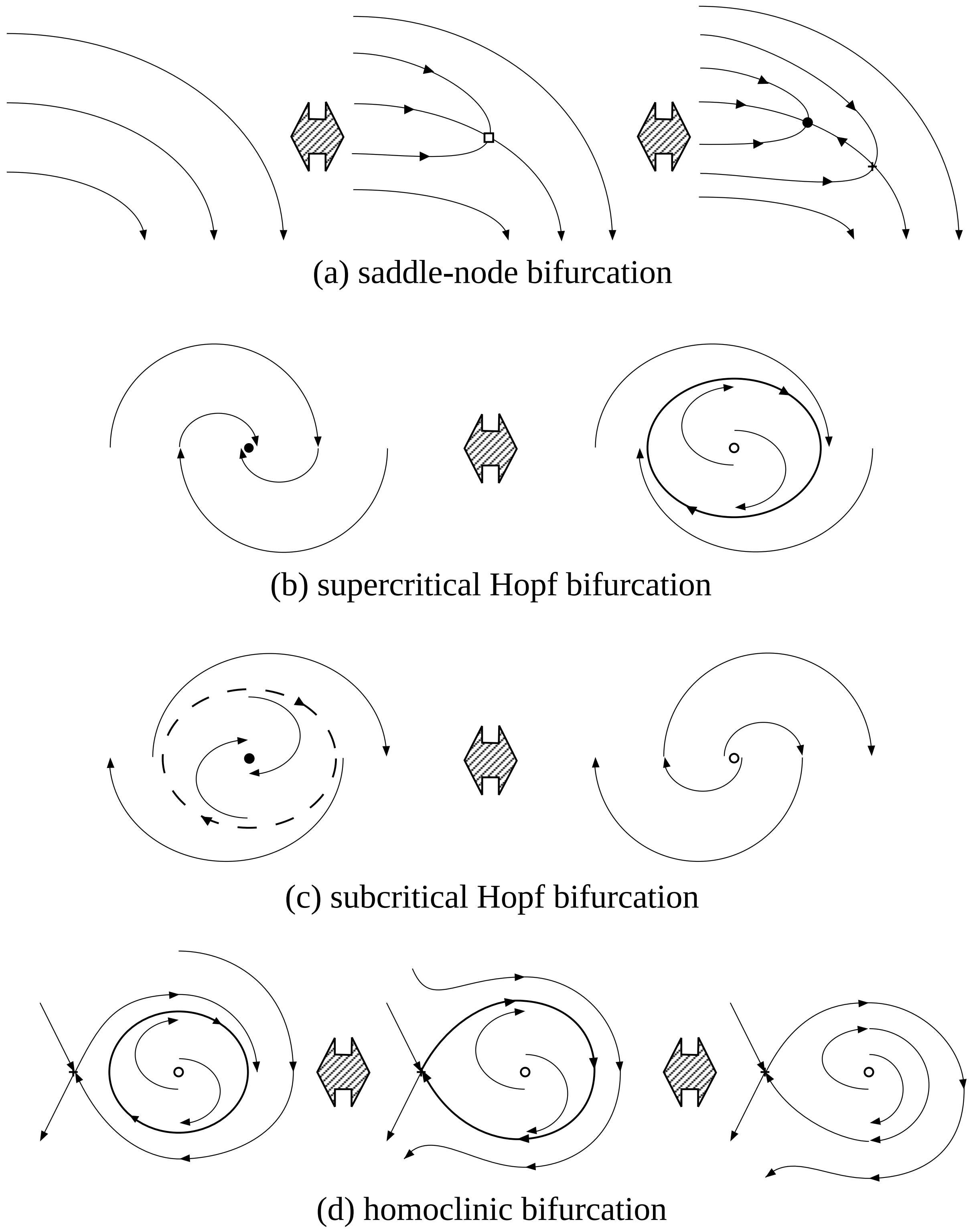
(a) saddle-node bifurcation
(b) supercritical Hopf bifurcation
(c) subcritical Hopf bifurcation
(d) homoclinic bifurcation

Figure 2: Codimension-one bifurcations.