CS840a
Learning and Computer Vision
Prof. Olga Veksler

Lecture 2

Some Slides on Optical flow are from Gary Bradski

Sebastian Thrun

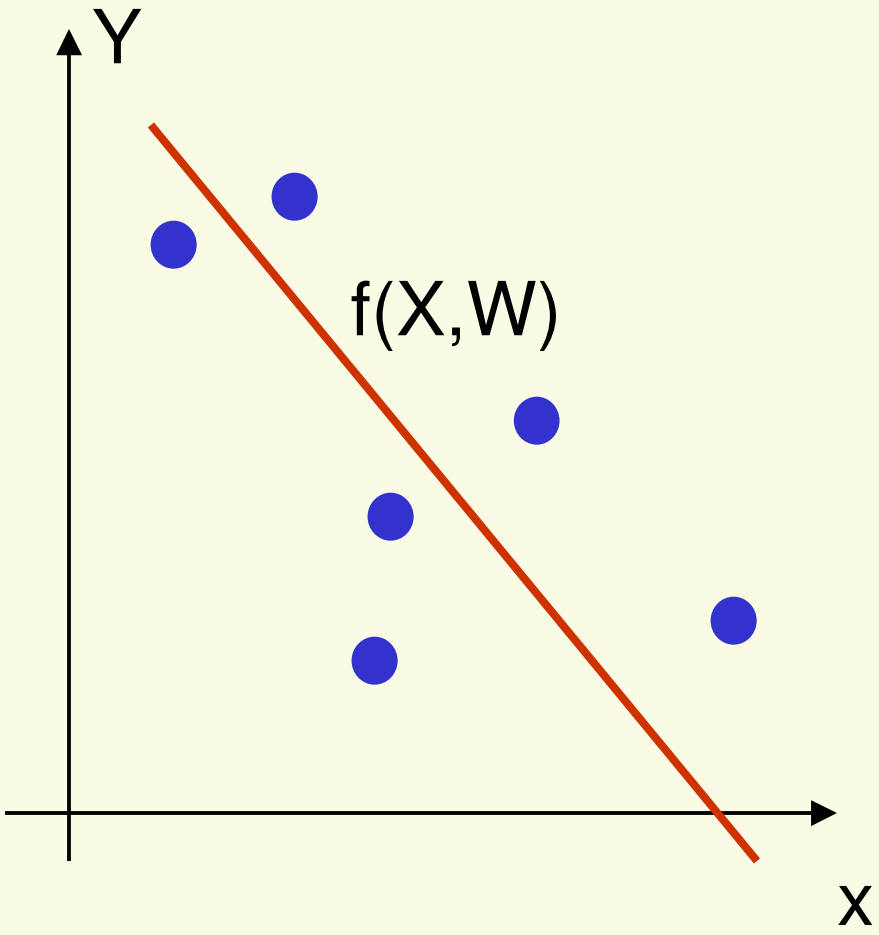
Today

- Continue introduction to Machine Learning
- Linear Machines
- Start preparation for the first paper
  - *"Recognizing Action at a Distance" by A. Efros, A.Berg, G. Mori, Jitendra Malik*
  - there should be a link to PDF file on our web site
- Next time:
  - Discuss the paper
  - Prepare for the second paper

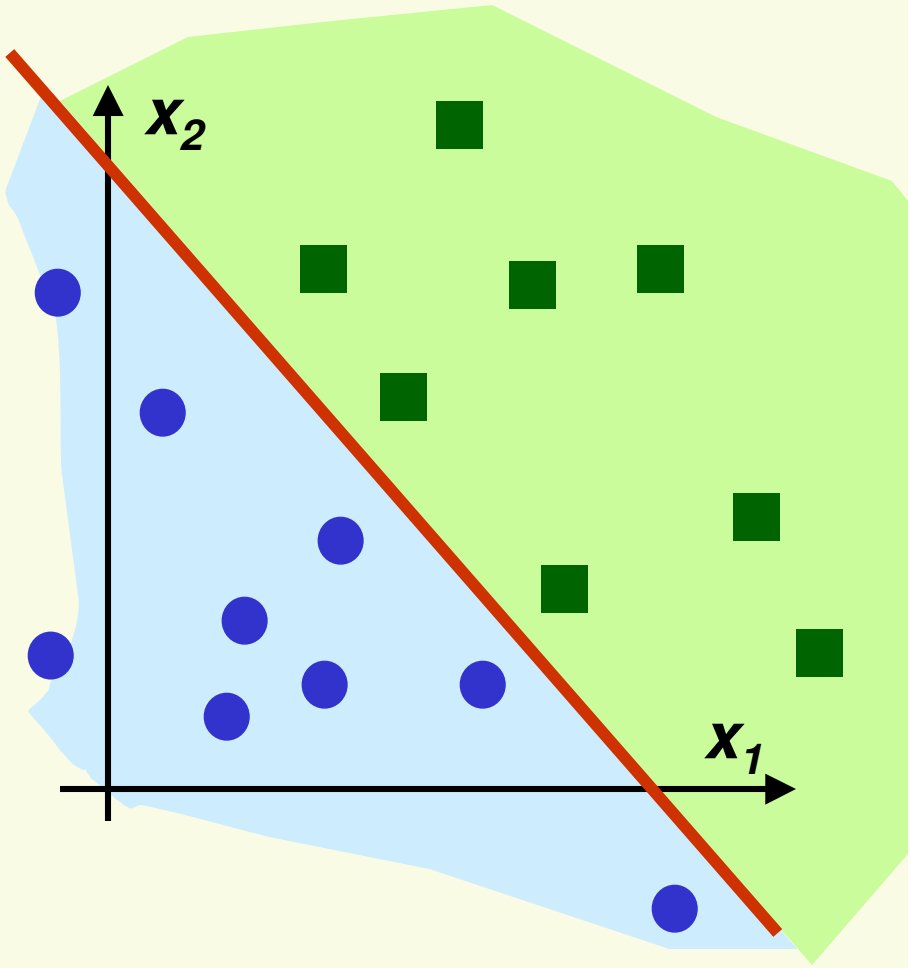
Last Time: Supervised Learning

- Training samples (or examples) $X^1, X^2, \ldots X^n$
- Each example is typically multi-dimensional
  - $X^i_1, X^i_2, \ldots, X^i_d$ are typically called *features*, X^i is sometimes called a *feature vector*
  - **How many features and which features do we take?**
- Know desired output for each example (labeled samples) $Y^1, Y^2, \ldots Y^n$
  - This learning is supervised ("teacher" gives desired outputs).
  - Y^i are often one-dimensional, but can be multidimensional
- Two types of supervised learning:
  - Classification:
    - Y^i takes value in finite set and typically called a *label* or a *class*
    - Example: $Y \in$ {sunny,cloudy,raining}
  - Regression, or function fitting:
    - Y^i continuous. In this case, it is typically called an *output value*
    - Example: Y=temperature $\in$ [-60,60]

Regression vs. Classification



- Learn (fit) function f(X,W)



- Convenient to define decision boundary between classes
- X is on decision boundary if f(X,W) is discontinuous at X

Last Time: Supervised Learning

- Wish to design a *machine* f(X,W) s.t. f(X,W) = true output value at X
  - In classification want f(X,W) = label of X
  - **How do we choose f?**
    - when we choose a particular f, we are making implicit assumptions about our problem
  - W is typically multidimensional vector of weights (also called *parameters*) which enable the machine to "learn"
    - $W = [w_1, w_2, \ldots w_k]$

Training and Testing

- There are 2 phases, training and testing
  - Divide all labeled samples $X^1, X^2, \ldots X^n$ into 2 sets, *training* set and *testing* set
  - Training phase is for "teaching" our machine (finding optimal weights W)
  - Testing phase is for evaluating how well our machine works on unseen examples
- Training phase
  - Find the weights W s.t. $f(X^i, W) = Y^i$ "as much as possible" for the *training* samples X^i
  - "as much as possible" needs to be defined
  - Training can be quite complex and time-consuming

Testing

- Testing phase
  - The goal is to design machine which performs well on unseen examples (which are typically different from labeled examples)
  - Evaluate the performance of the trained machine f(X,W) on the testing samples (unseen labeled samples)
  - Testing the machine on unseen labeled examples lets us approximate how well it will perform in practice
  - If testing results are poor, may have to go back to the training phase and redesign f(X,W)

Loss Function

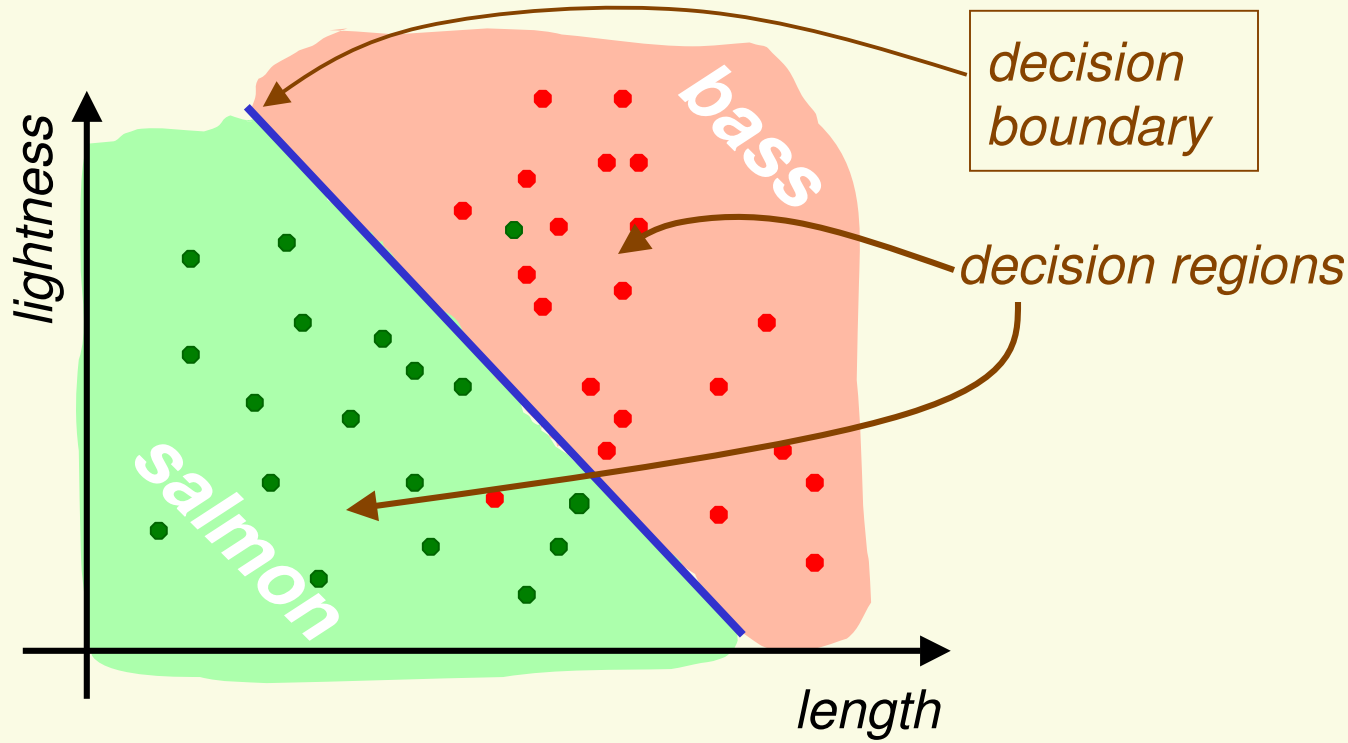
- How do we quantify what it means for the machine f(X,W) do well in the training and testing phases?
- f(X,W) has to be "close" to the true output on X
- Define Loss (or Error) function L
  - This is up to the designer (that is you)
- Typically first define per-sample loss $L(X^i,Y^i,W)$
  - Some examples:
    - for classification, $L(X^i,Y^i,W) = \mathbf{I}[f(X^i,W) \neq Y^i]$, where $\mathbf{I}[\text{true}] = 1$, $\mathbf{I}[\text{false}] = 0$
      - we just care if the sample has been classified correctly
    - For continuous Y, $L(X^i,Y^i,W) = || f(X^i,W) - Y^i ||^2$,
      - how far is the estimated output from the correct one?
- Then loss function $L = \Sigma_i L(X^i,Y^i,W)$
  - Number of missclassified example for classification
  - Sum of distances from the estimated output to the correct output

Generalization and Overfitting

- *Generalization* is the ability to produce correct output on previously unseen examples
  - In other words, low error (loss) on unseen examples
  - **Good generalization is the main goal of ML**
- Low train error does not necessarily imply that we will have low test error
  - Very easy to produce f(X,W) which is perfect on training samples
    - "memorize" all the training samples and output their correct label
    - random label on unseen examples
    - No training error but horrible test error
- *Overfitting*
  - when the machine performs well on training data but poorly on testing data

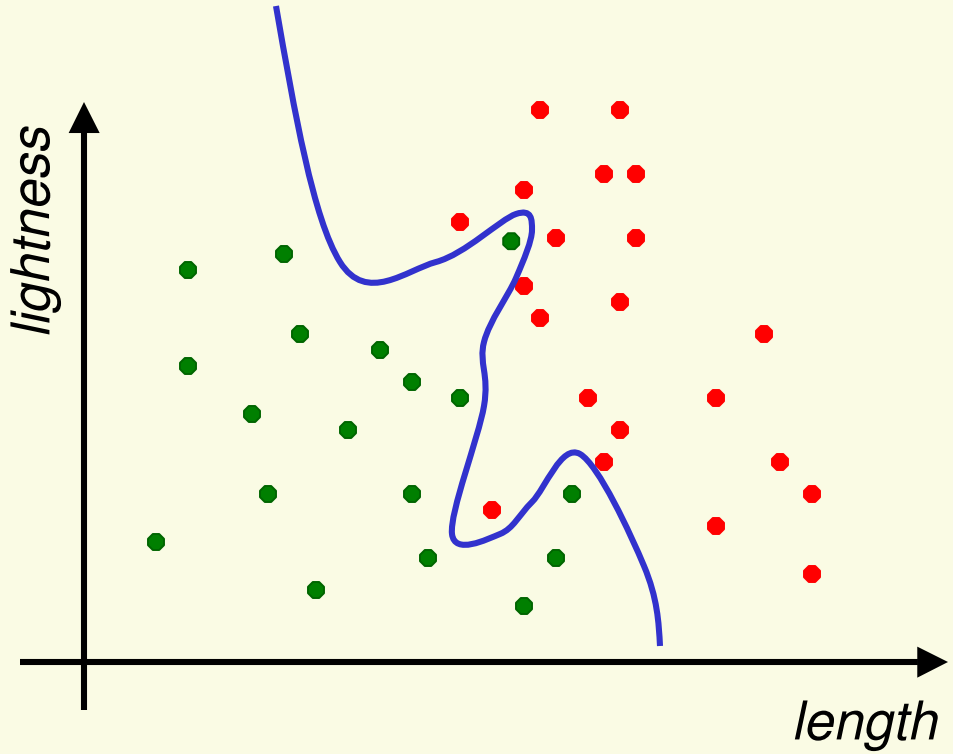
Separating Salmon from Bass

- Use *length* and *lightness* as features
- Feature vector [*length*,*lightness*]



- Classification error 4%

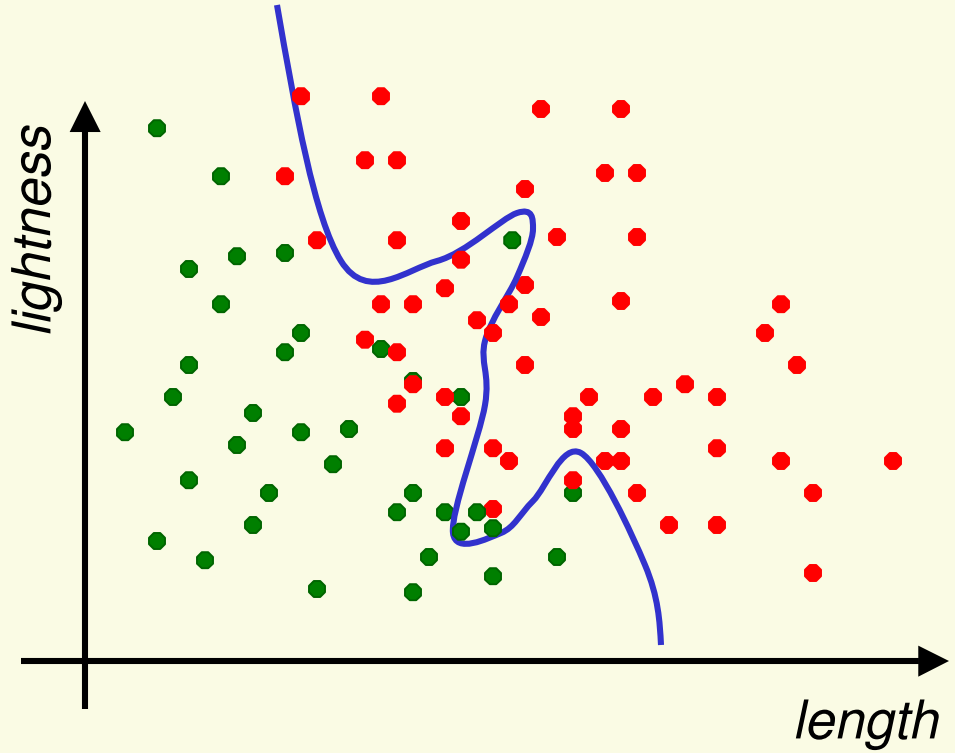
Better decision boundary



- Ideal decision boundary, 0% classification error

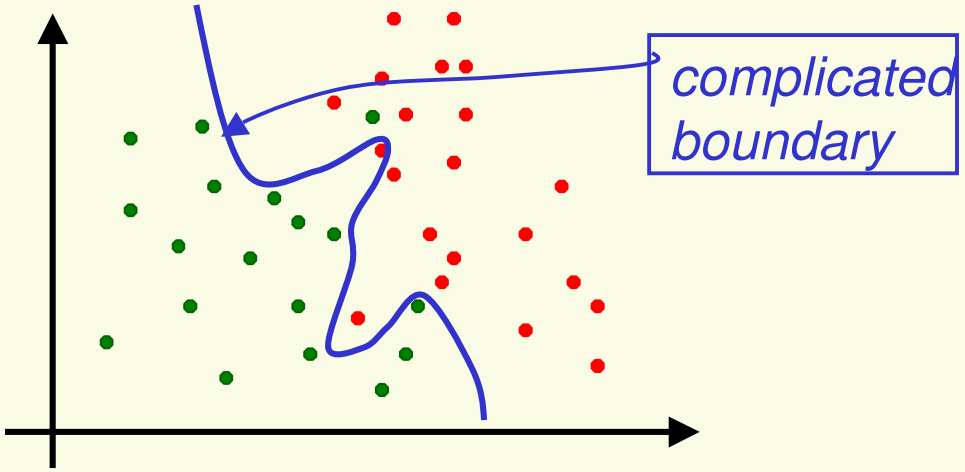
Test Classifier on New Data

- Classifier should perform well on new data
- Test “ideal” classifier on new data: 25% error



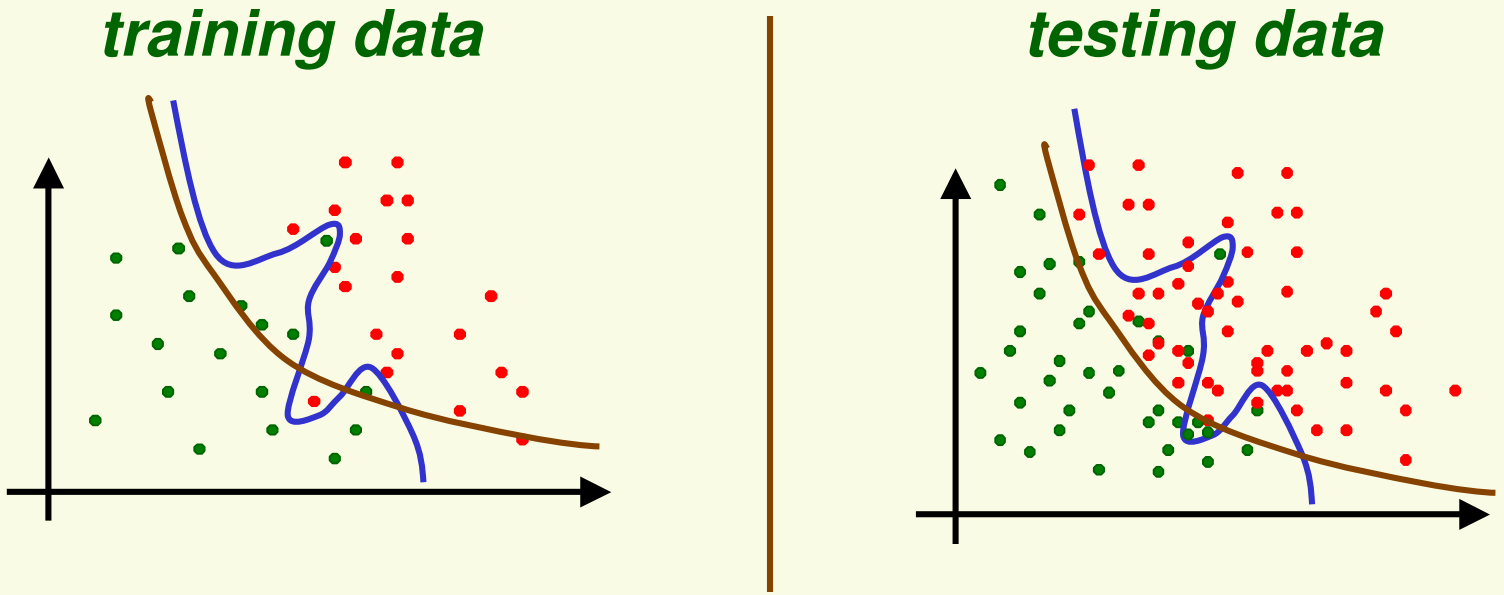
What Went Wrong?

- Poor ***generalization***



- Complicated boundaries do not generalize well to the new data, they are too "tuned" to the particular training data, rather than some true model which will separate salmon from sea bass well.
  - This is called overfitting the data

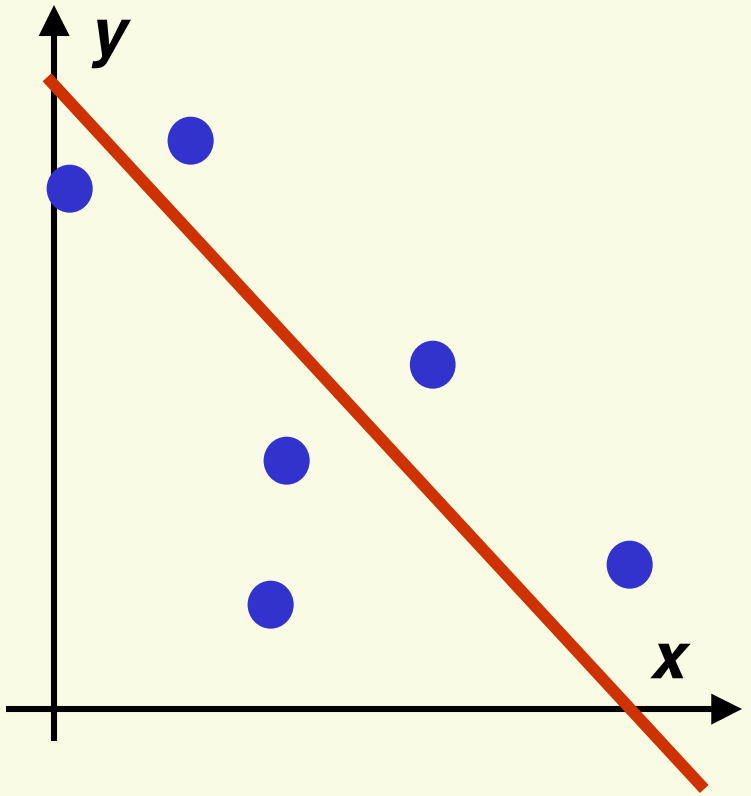
Generalization



- Simpler decision boundary does not perform ideally on the training data but generalizes better on new data
- Favor simpler classifiers
  - William of Occam (1284-1347): “entities are not to be multiplied without necessity”

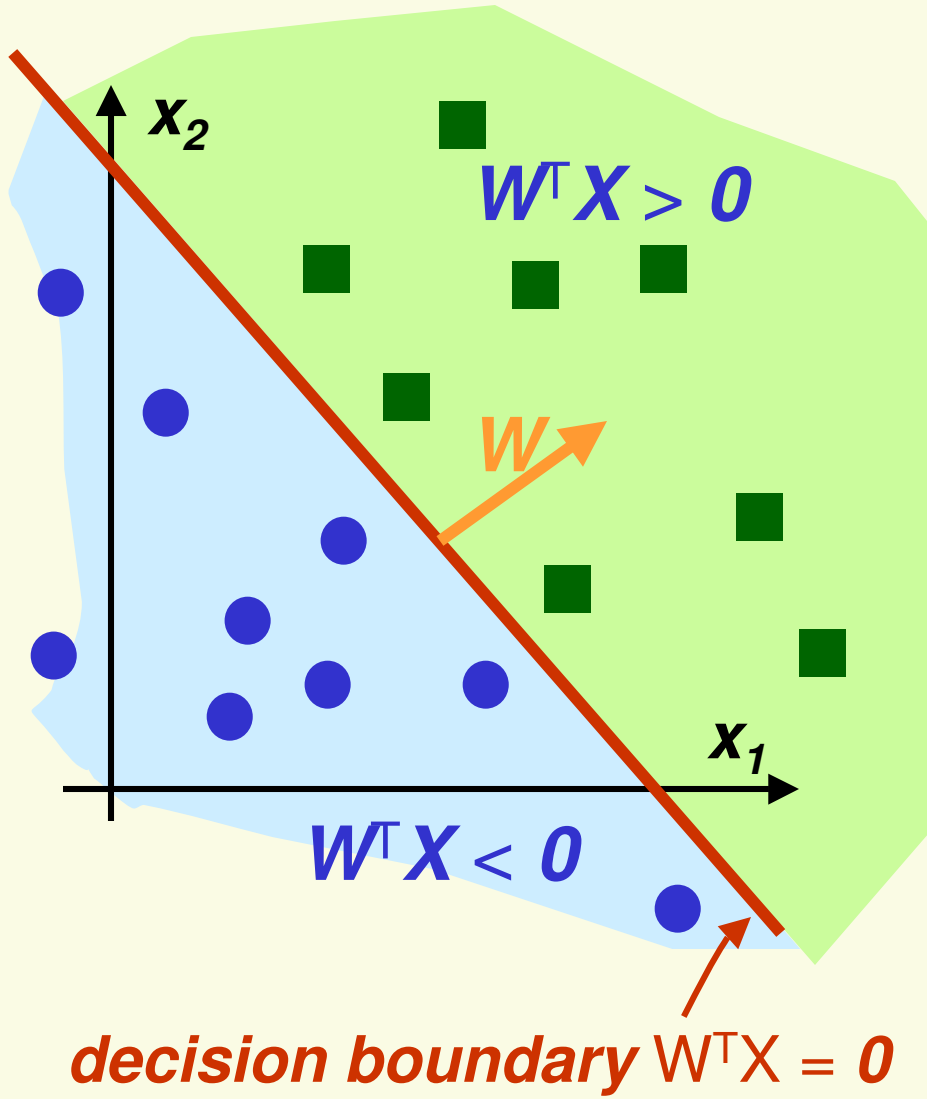
Linear Machine, Continuous Y

- $f(X,W) = w_0+\Sigma_{i=1,2,...d}\ w_i x_i$
  - w_0 is called bias
- In vector form, if we let $X = (1,x_1,x_2,\ldots,x_d)$, then $f(X,W) = W^T X$
  - notice abuse of notation
- This is standard linear regression (line fitting)
  - assume $L(X^i,Y^i,W) = || f(X^i,W) - Y^i ||^2$
  - optimal W can be found by solving linear system of equations $W^* = [\Sigma X^i (X^i)^T]^{-1} \Sigma Y^i X^i$



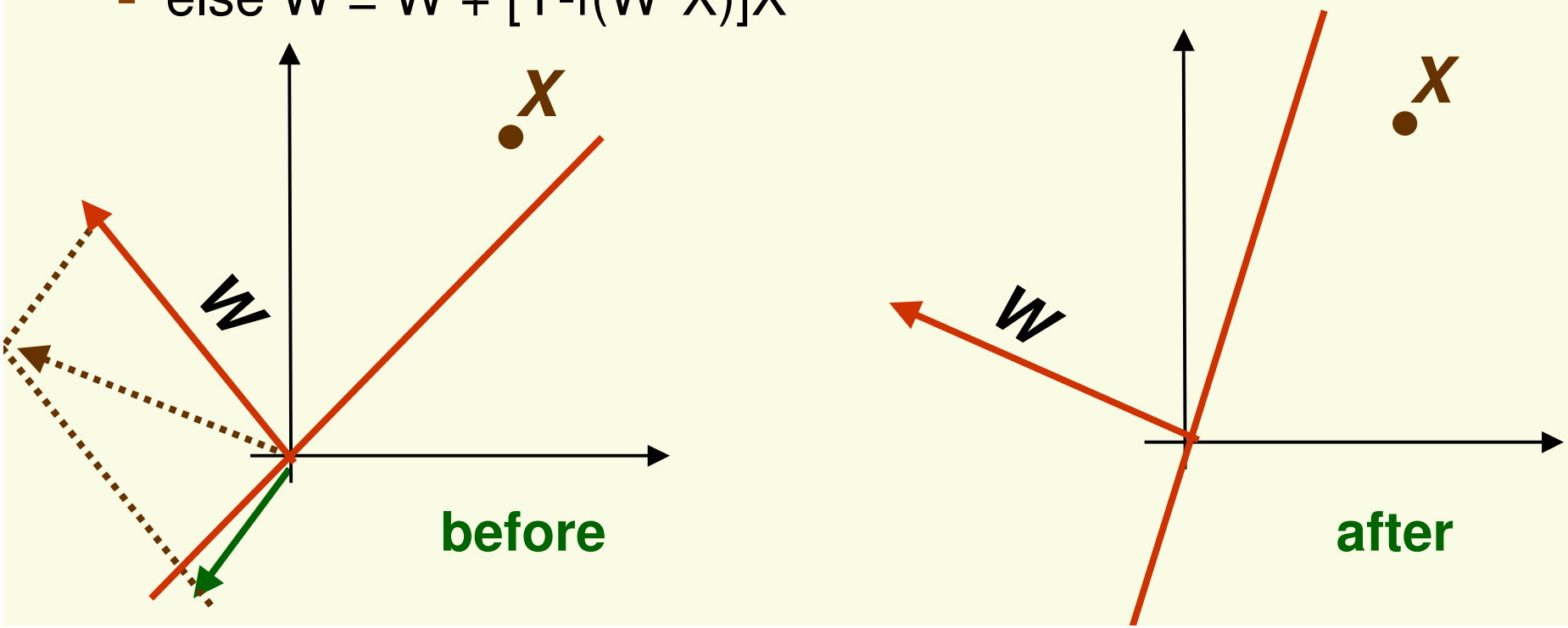
Linear Machine: binary Y

- $f(X,W) = \text{sign}(w_0+\Sigma_{i=1,2,...d}\ w_i x_i)$
  - sign(positive) = 1, sign(negative) = -1
  - w_0 is called bias
- In vector form, if we let $X = (1,x_1,x_2,\ldots,x_d)$ then $f(X,W) = \text{sign}(W^T X)$



Perceptron Learning Procedure (Rosenblatt 1957)

- $f(X,W) = \text{sign}(w_0+\Sigma_{i=1,2,...d}\ w_i x_i)$
- Let $L(X^i,Y^i,W) = \mathbf{I}[f(X^i,W) \neq Y^i]$. How do we learn W?
- A solution:
- Iterate over all training samples
  - if f(X,W)=Y (correct label), do nothing
  - else $W = W + [Y-f(W^T X)]X$



Perceptron Learning Procedure (Rosenblatt 1957)

- Amazing fact: If the samples are linearly separable, the perceptron learning procedure will converge to a solution (separating hyperplane) in a finite amount of time
- Bad news: If the samples are not linearly separable, the perceptron procedure will not terminate, it will go on looking for a solution which does not exist!
- For most interesting problems the samples are not linearly separable
- Is there a way to learn W in non-separable case?
  - Remember, it's ok to have training error, so we don't have to have "perfect" classification

Optimization

- Need to minimize a function of many variables

$$J(x) = J(x_1, \ldots, x_d)$$

- We know how to minimize $J(x)$
  - Take partial derivatives and set them to zero

$$\begin{bmatrix} \frac{\partial}{\partial x_1} J(x) \\ \vdots \\ \frac{\partial}{\partial x_d} J(x) \end{bmatrix} = \nabla J(x) = 0$$

gradient

- However solving analytically is not always easy
  - Would you like to solve this system of nonlinear equations?

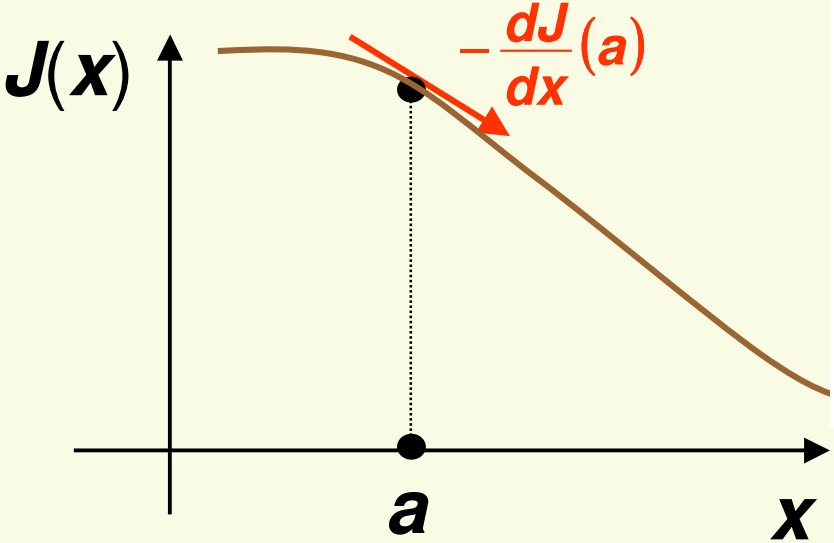
$$\begin{cases} \sin\left(x_1^2 + x_2^3\right) + e^{x_4^2} = 0 \\ \cos\left(x_1^2 + x_2^3\right) + \log\left(x_5^3\right)^{x_4^2} = 0 \end{cases}$$

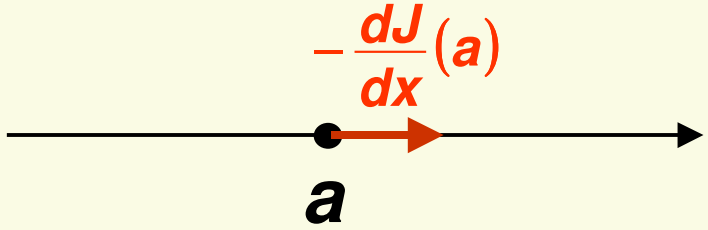
  - Sometimes it is not even possible to write down an analytical expression for the derivative, we will see an example later today

Optimization: Gradient Descent

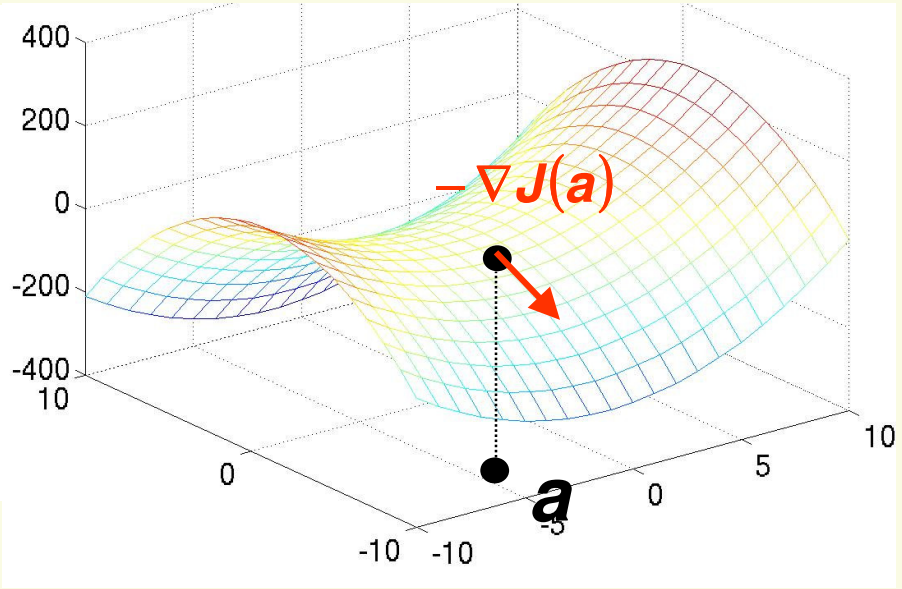
- Gradient $\nabla J(\boldsymbol{x})$ points in direction of steepest increase of $J(\boldsymbol{x})$, and $-\nabla J(\boldsymbol{x})$ in direction of steepest decrease

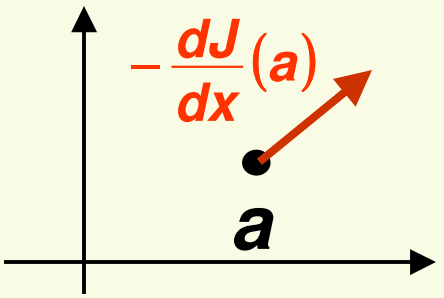
one dimension



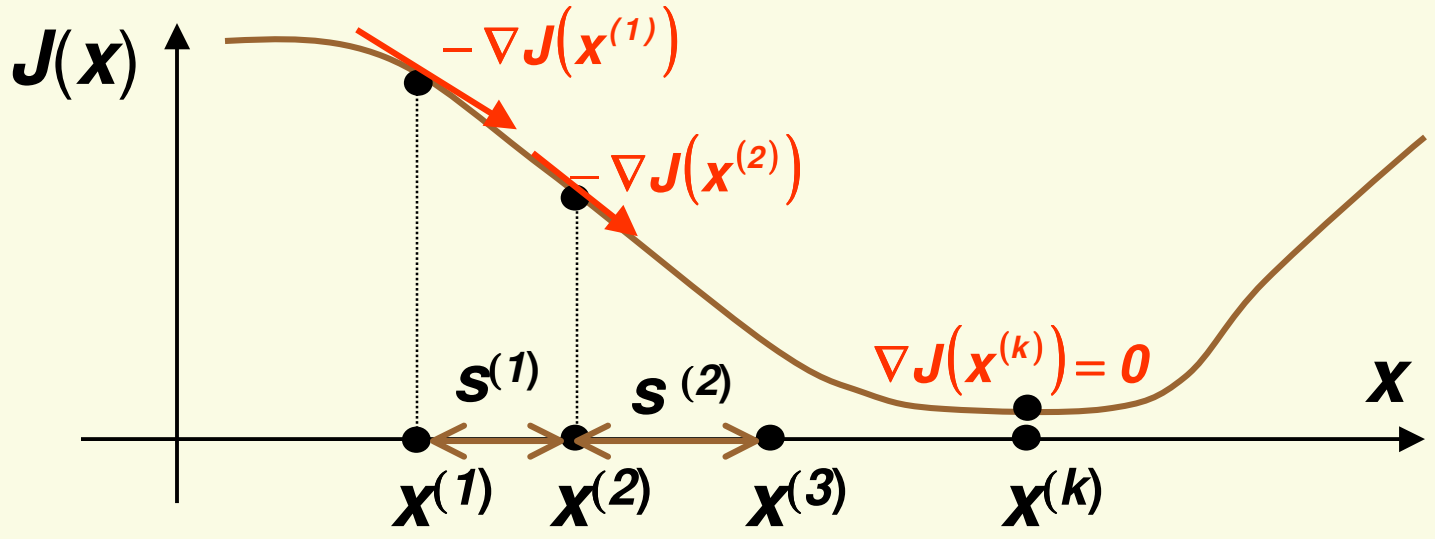


two dimensions





Optimization: Gradient Descent



Gradient Descent for minimizing any function $J(x)$

set $k = 1$ ***and*** $x^{(1)}$ to some initial guess for the weight vector

while $\eta^{(k)} \left| \nabla J\left(x^{(k)}\right) \right| > \varepsilon$

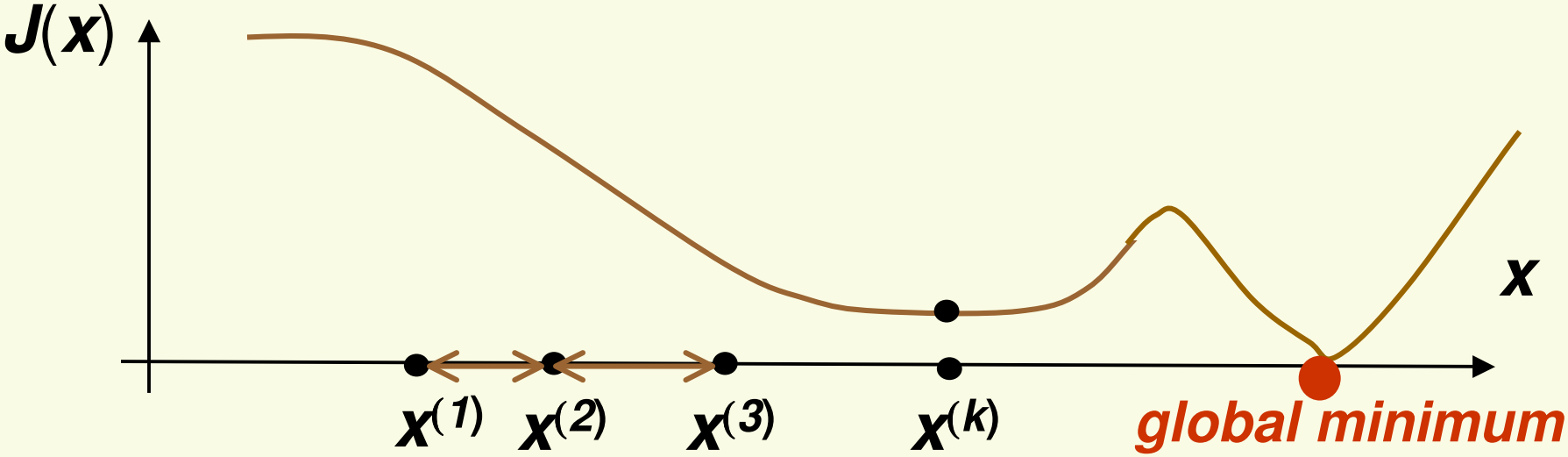
choose learning rate $\eta^{(k)}$

$$x^{(k+1)} = x^{(k)} - \eta^{(k)} \nabla J(x)$$ (***update rule***)

$$k = k + 1$$

Optimization: Gradient Descent

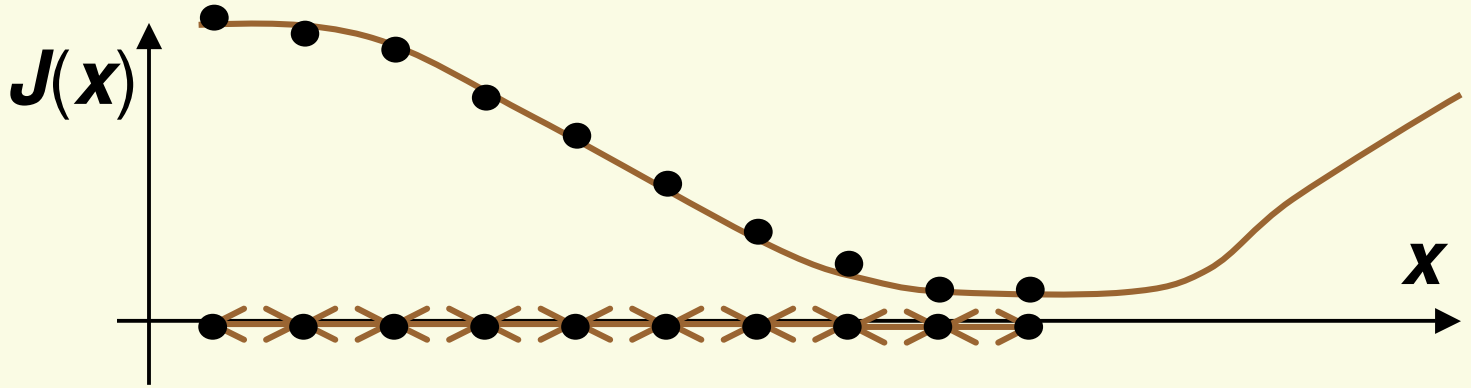
- Gradient descent is guaranteed to find only a local minimum



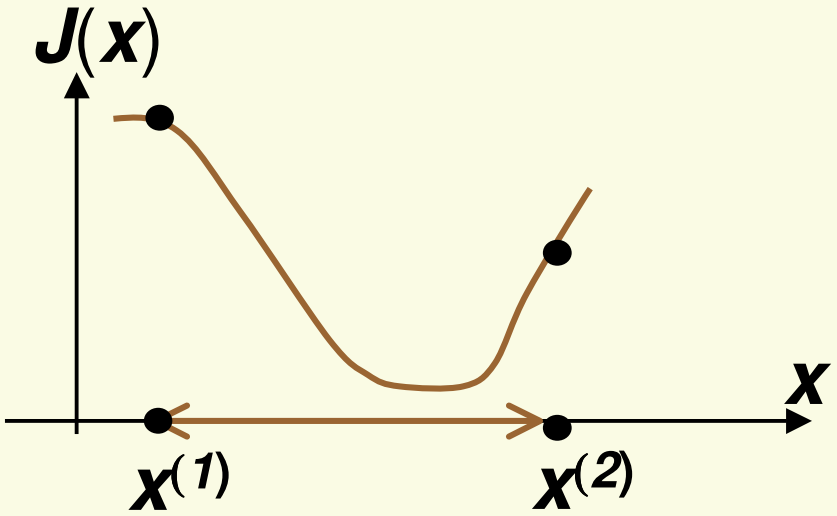
- Nevertheless gradient descent is very popular because it is simple and applicable to any differentiable function

Optimization: Gradient Descent

- Main issue: how to set parameter η (***learning rate***)
- If η is too small, need too many iterations



- If η is too large may overshoot the minimum and possibly never find it (if we keep overshooting)

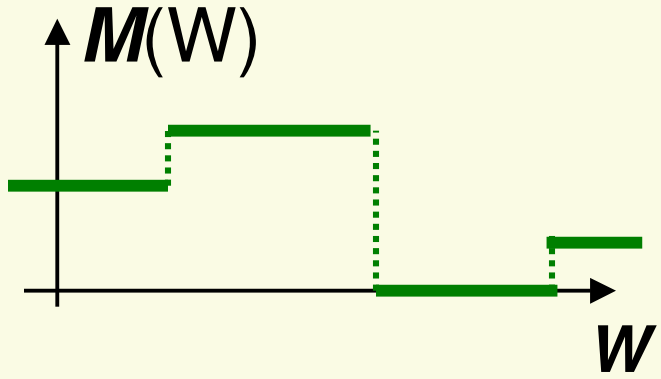


"Optimal" W with Gradient Descent

- $f(X,W) = \text{sign}(w_0+\Sigma_{i=1,2,...d}\ w_i x_i)$
- If we let $L(X^i,Y^i,W) = \mathbf{I}[f(X^i,W) \neq Y^i]$, then L(W) is the number of missclassified examples
- Let ***M*** be the set of examples misclassified by ***W***

$$\boldsymbol{M(W) = \{ sample\ \ X^i\ \ s.t.\ \ W^T X^i \neq Y^i \}}$$

- Then L(W) = |M(W)|, the size of M(W)
- L(W) is piecewise constant, gradient descent is useless

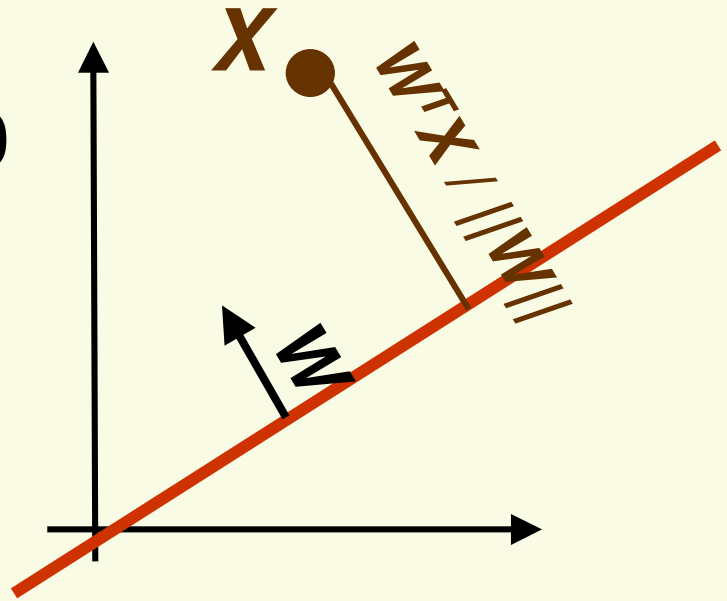


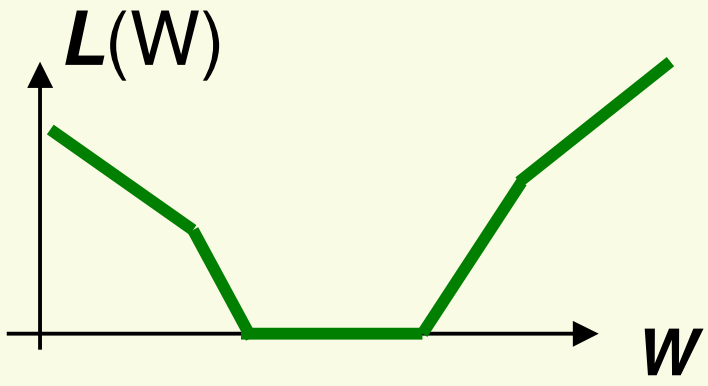
"Optimal" W with Gradient Descent

- Better choice:

$$L(W) = \sum_{X^i \in M} \left(-W^T X^i\right) Y^i$$

- If X^i is misclassified, $(W^T X^i) Y^i \leq 0$
- Thus $L(W, X^i, Y^i) \geq 0$
- $L(W, X^i, Y^i)$ is proportional to the distance of misclassified example to the decision boundary
- $L(W) = \Sigma L(W, X^i, Y^i)$ is piecewise linear and thus suitable for gradient decent





Batch Rule

$$L(W, X^i, Y^i) = \sum_{X \in M} (-W^T X) Y$$

- Gradient of L is $\nabla L(W) = \sum_{X \in M} (-X) Y$
  - M are samples misclassified by W
  - It is not possible to solve ∇L(W) = 0 analytically
- Update rule for gradient descent: $x^{(k+1)} = x^{(k)} - \eta^{(k)} \nabla J(x)$
- Thus *gradient decent batch update rule* for L(W) is:

$$W^{(k+1)} = W^{(k)} + \eta^{(k)} \sum_{Y \in M} XY$$

- It is called batch rule because it is based on all misclassified examples

Single Sample Rule

- Thus *gradient decent single sample rule* for $\boldsymbol{L}(W)$ is:

$$\boldsymbol{W}^{(k+1)} = \boldsymbol{W}^{(k)} + \eta^{(k)}(\boldsymbol{XY})$$

  - apply for any sample X misclassified by $\boldsymbol{W}^{(k)}$
  - must have a consistent way of visiting samples

Convergence

- If classes are linearly separable, and $\eta^{(k)}$ is fixed to a constant, i.e. $\eta^{(1)} = \eta^{(2)} = \ldots = \eta^{(k)} = \boldsymbol{c}$ (*fixed learning rate*)
  - *both single sample and batch rules converge to a correct solution* (could be any $\boldsymbol{W}$ in the solution space)
- If classes are not linearly separable:
  - Single sample algorithm does not stop, it keeps looking for solution which does not exist
  - However by choosing appropriate learning rate, heuristically stop algorithm at hopefully good stopping point

$$\eta^{(k)} \to \mathbf{0} \;\; \boldsymbol{as} \;\; \boldsymbol{k} \to \infty$$

  - for example, $\eta^{(k)} = \dfrac{\eta^{(1)}}{\boldsymbol{k}}$
  - for this learning rate convergence in the linearly separable case can also be proven

Learning by Gradient Descent

- Suppose we suspect that the machine has to have functional form f(X,W), not necessarily linear
- Pick differentiable per-sample loss function $L(X^i,Y^i,W)$
- We need to find W that minimizes $L = \Sigma_i L(X^i,Y^i,W)$
- Use gradient-based minimization:
  - Batch rule: $W = W - \eta \nabla L(W)$
  - Or single sample rule: $W = W - \eta \nabla L (X^i,Y^i,W)$

Important Questions

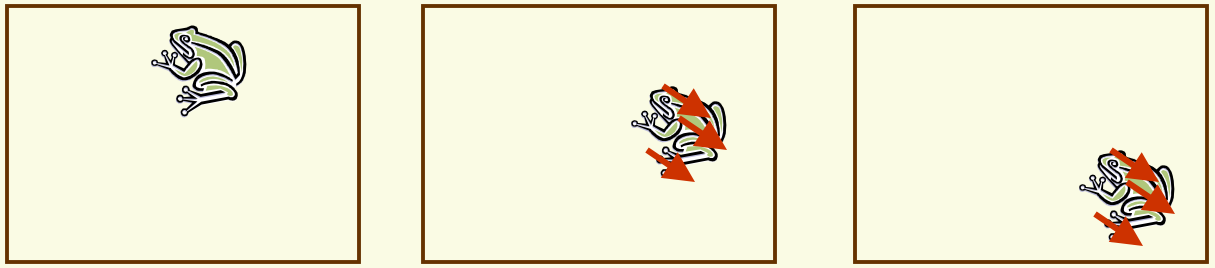
- How do we choose the feature vector X?
- How do we split labeled samples into training/testing sets?
- How do we choose the machine f(X,W)?
- How do we choose the loss function $L(X^i,Y^i,W)$?
- How do we find the optimal weights W?

Next Time

- Paper: *"Recognizing Action at a Distance" by A. Efros, A.Berg, G. Mori, Jitendra Malik*
- Bring in a typed discussion on this paper
  - Should be only a few paragraphs long, **definitely less than 1 side of a page (typed)**
  - Your discussion should have the following:
    - very short description of the problem paper tries to solve
    - What makes this problem difficult?
    - Short description of the method used in the paper to solve the problem
    - What is the contribution of the paper (what new does it do)?
    - Do the experimental results look "good" to you?

Optical Flow

- Suppose we have a video sequence
- Optical flow is the apparent motion of brightness patterns from one frame to the next



- Many algorithms are available to compute optical flow, however up to date the results are not very reliable

Optical Flow

$$I(x+dx, y+dy, t+dt) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \ldots,$$

$$I(x+dx, y+dy, t+dt) = I(x, y, t),$$

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \ldots = 0.$$

$$\frac{dx}{dt} = u, \qquad \frac{dy}{dt} = v,$$

$$-\frac{\partial I}{\partial t} = \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v,$$

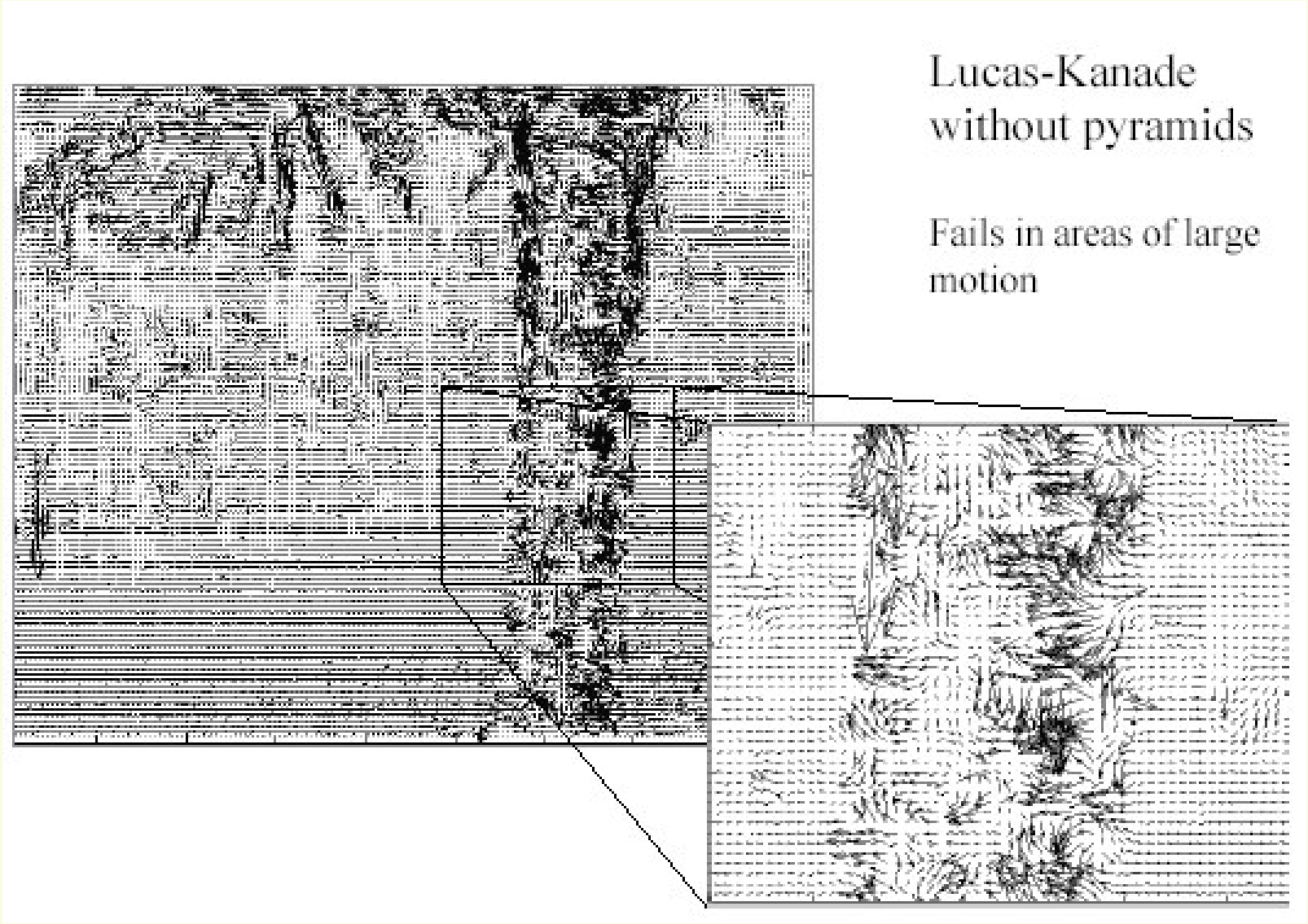
optical flow constraint equation

Video Sequence

*

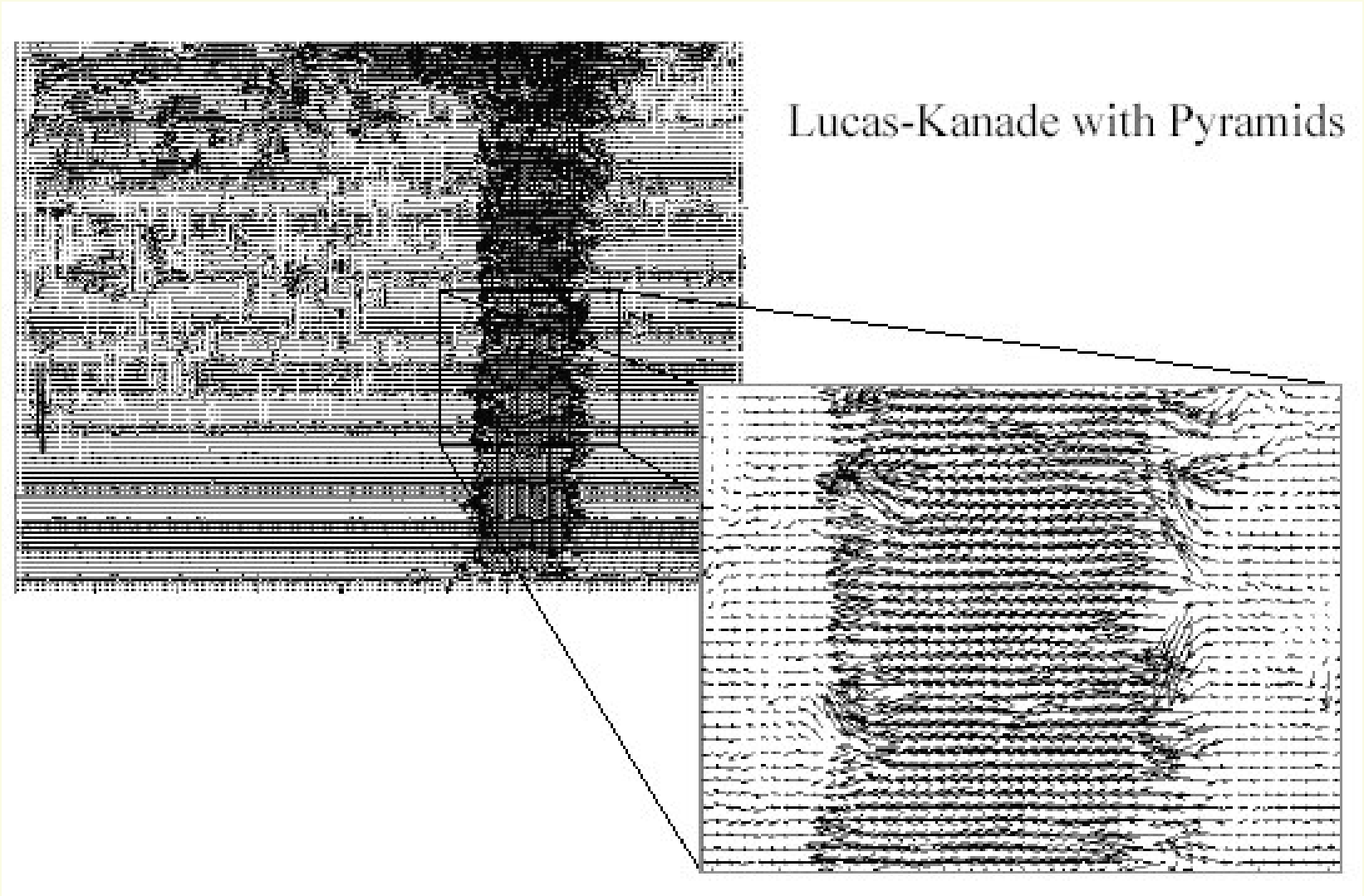
* Picture from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow Results



* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

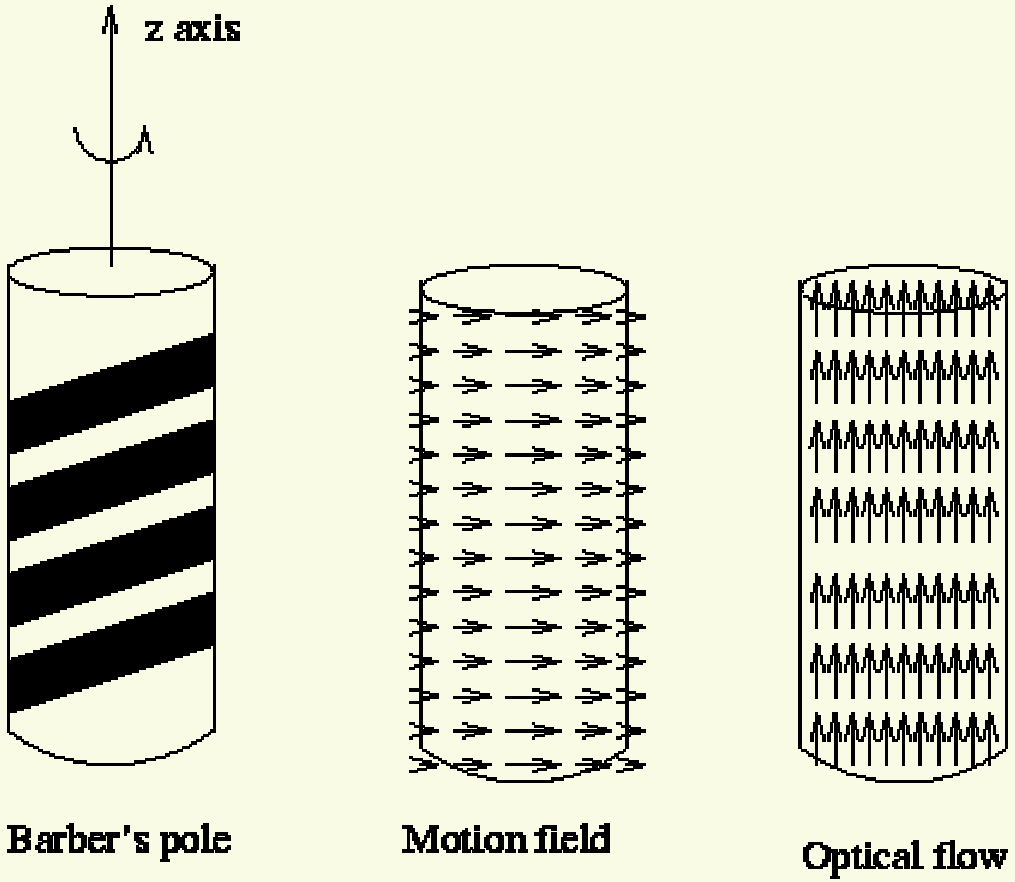
Optical Flow Results



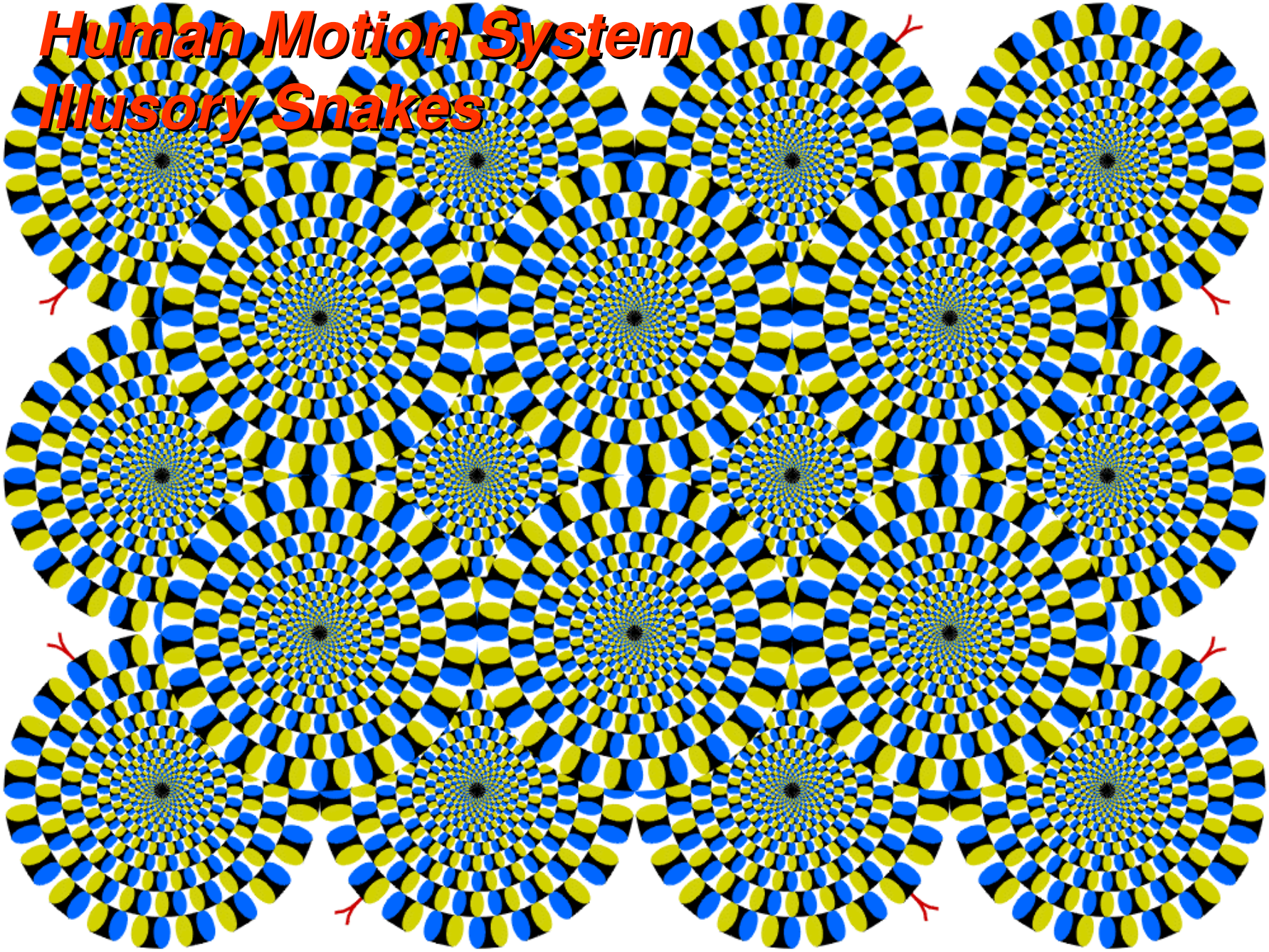
* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow vs. Motion Field

- Often (but not always) optical flow corresponds to the true motion of the scene



Human Motion System
Illusory Snakes



from Gary Bradski and Sebastian Thrun

Other Concepts to Review

- Image gradient
- Cross-correlation
- Convolution
- Gaussian smoothing (blurring)