CS 1674: Intro to Computer Vision

Visual Recognition

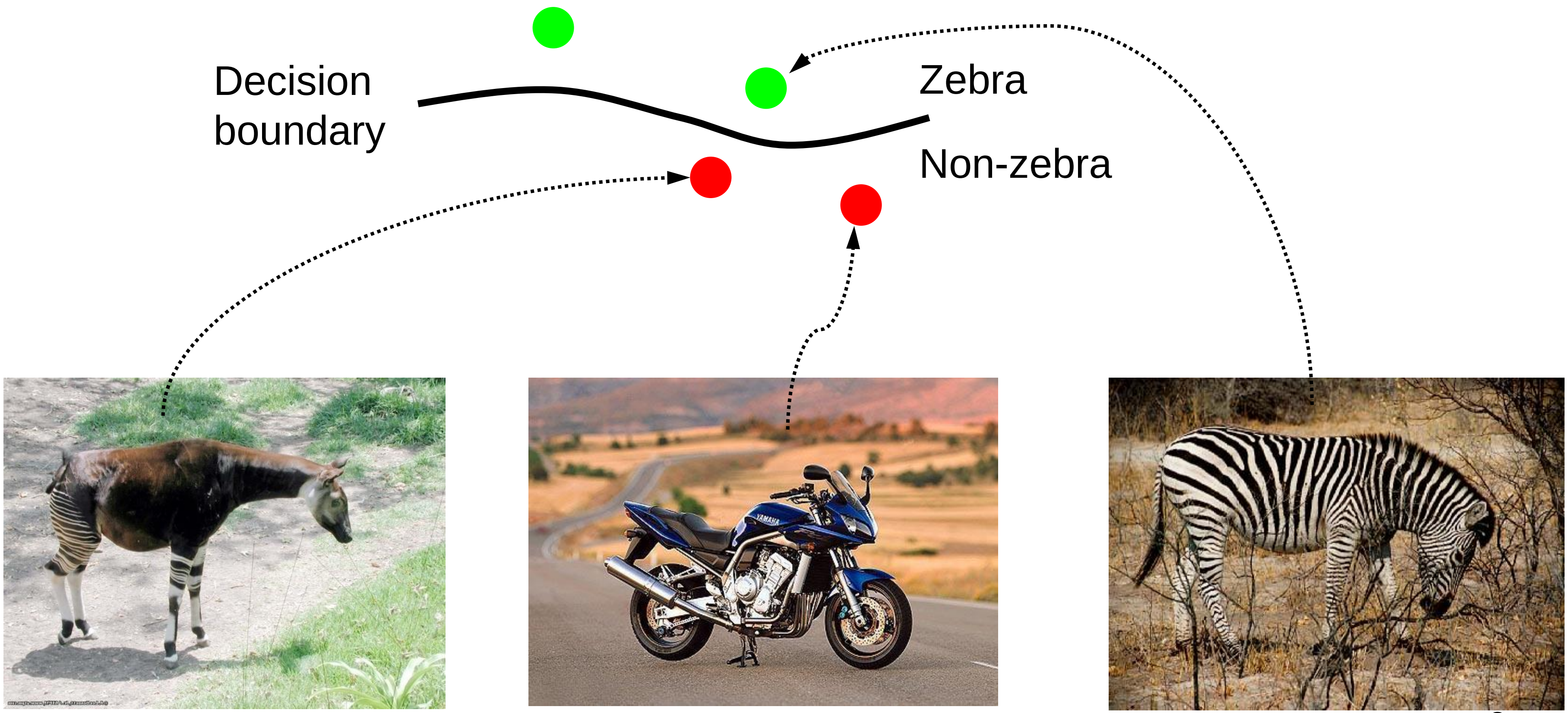
Prof. Adriana Kovashka
University of Pittsburgh
March 1, 2022

Plan for this lecture

- What is recognition?
  - a.k.a. classification, categorization
- Support vector machines
  - Separable case / non-separable case
  - Linear / non-linear (kernels)
- Example approach for scene classification
- The importance of generalization
  - The bias-variance trade-off (applies to all classifiers)

Classification

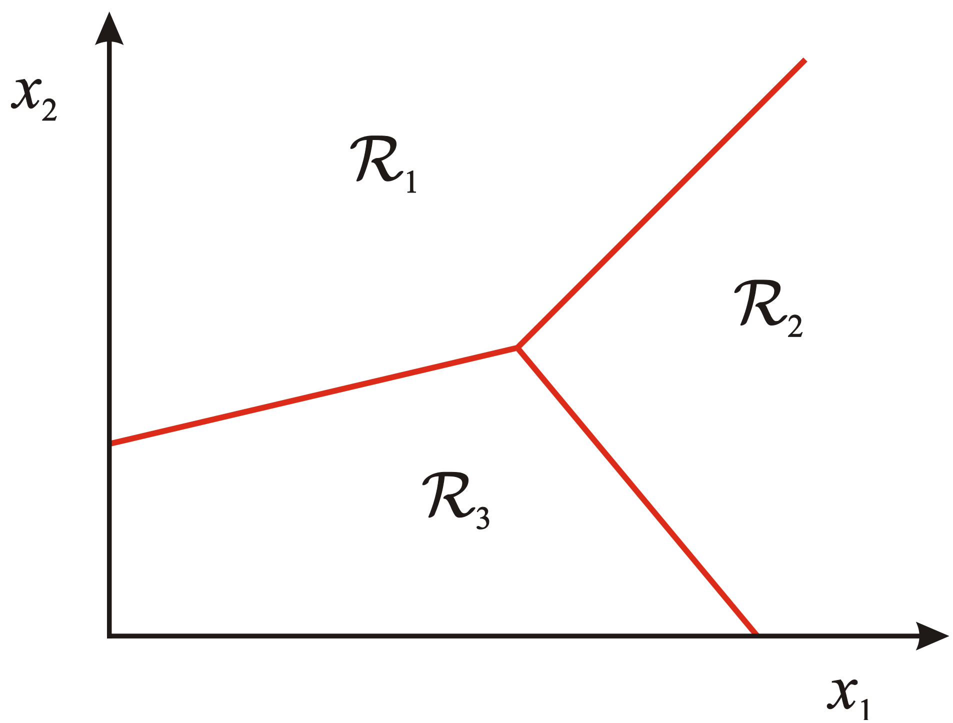
- Given a feature representation for images, how do we learn a model for distinguishing features from different classes?



Slide credit: L. Lazebnik

Classification

- Assign input vector to one of two or more classes
- Input space divided into *decision regions* separated by *decision boundaries*



Slide credit: L. Lazebnik

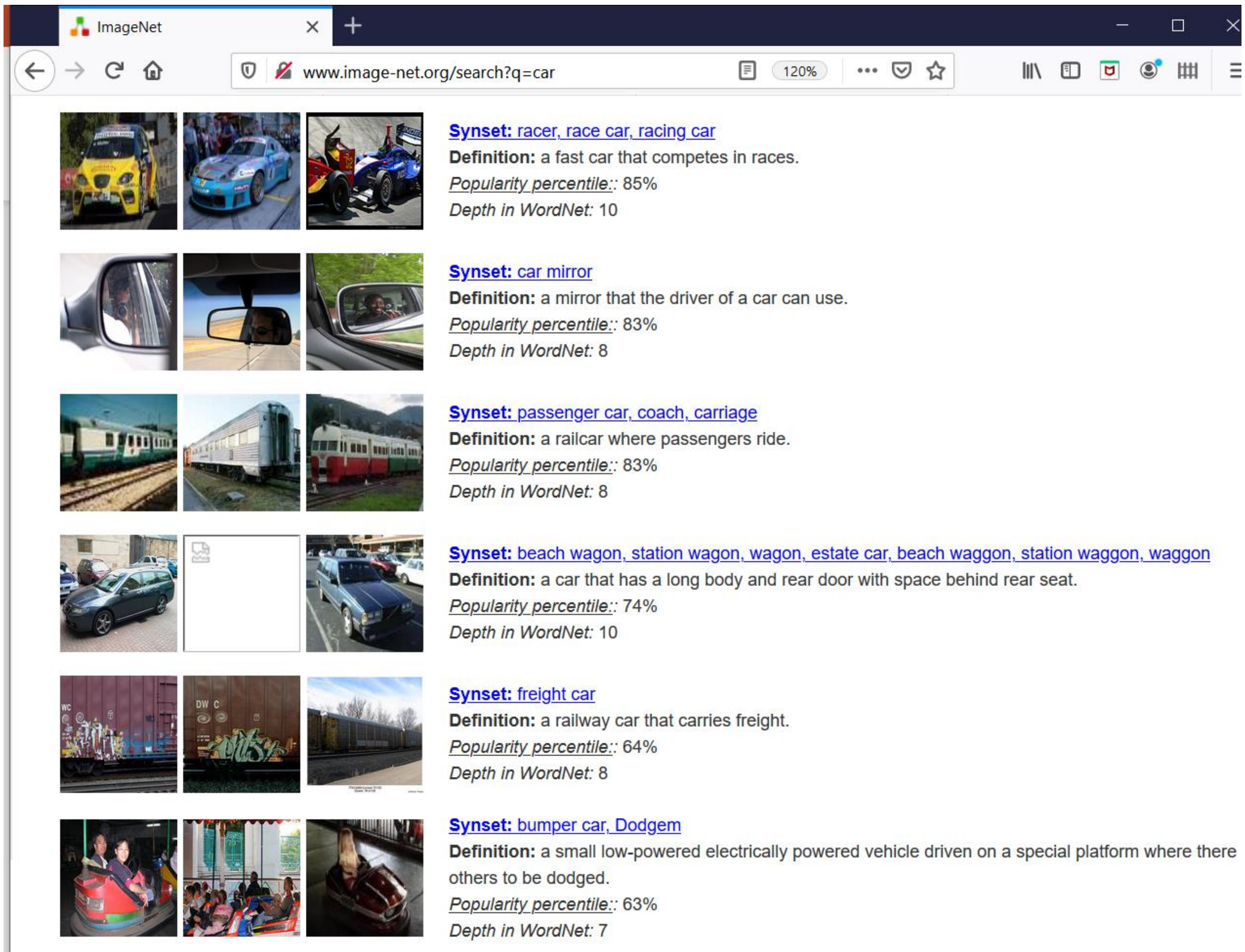
Examples of image classification

- Two-class (binary): Cat vs Dog

Adapted from D. Hoiem

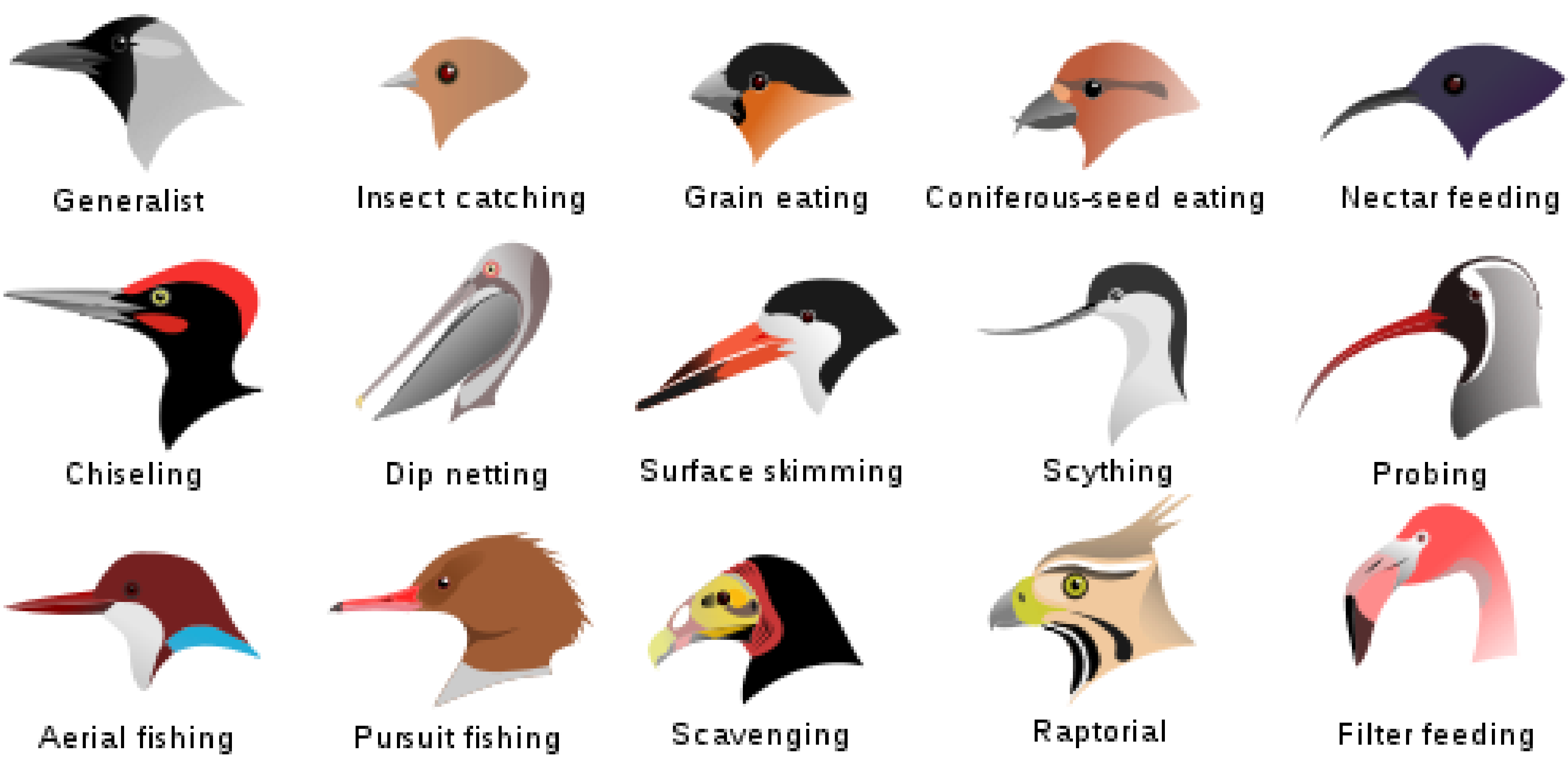
Examples of image classification

- Multi-class (often): Object recognition



Examples of image classification

- Fine-grained recognition



Visipedia Project

Slide credit: D. Hoiem

Examples of image classification

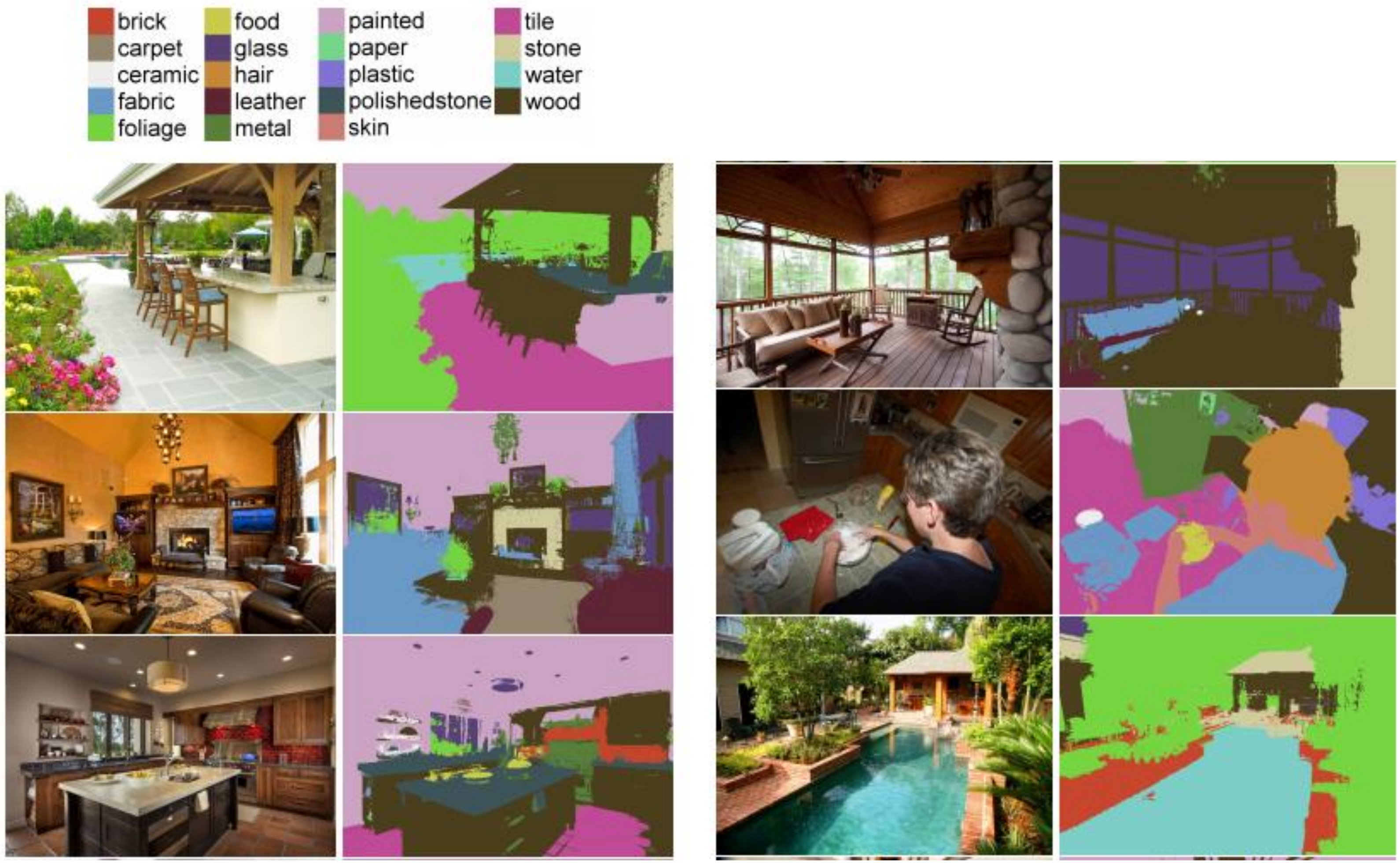
- Place recognition

Places Database [Zhou et al. NIPS 2014]

Slide credit: D. Hoiem

Examples of image classification

- Material recognition



[Bell et al. CVPR 2015]

Slide credit: D. Hoiem

Examples of image classification

- Dating historical photos

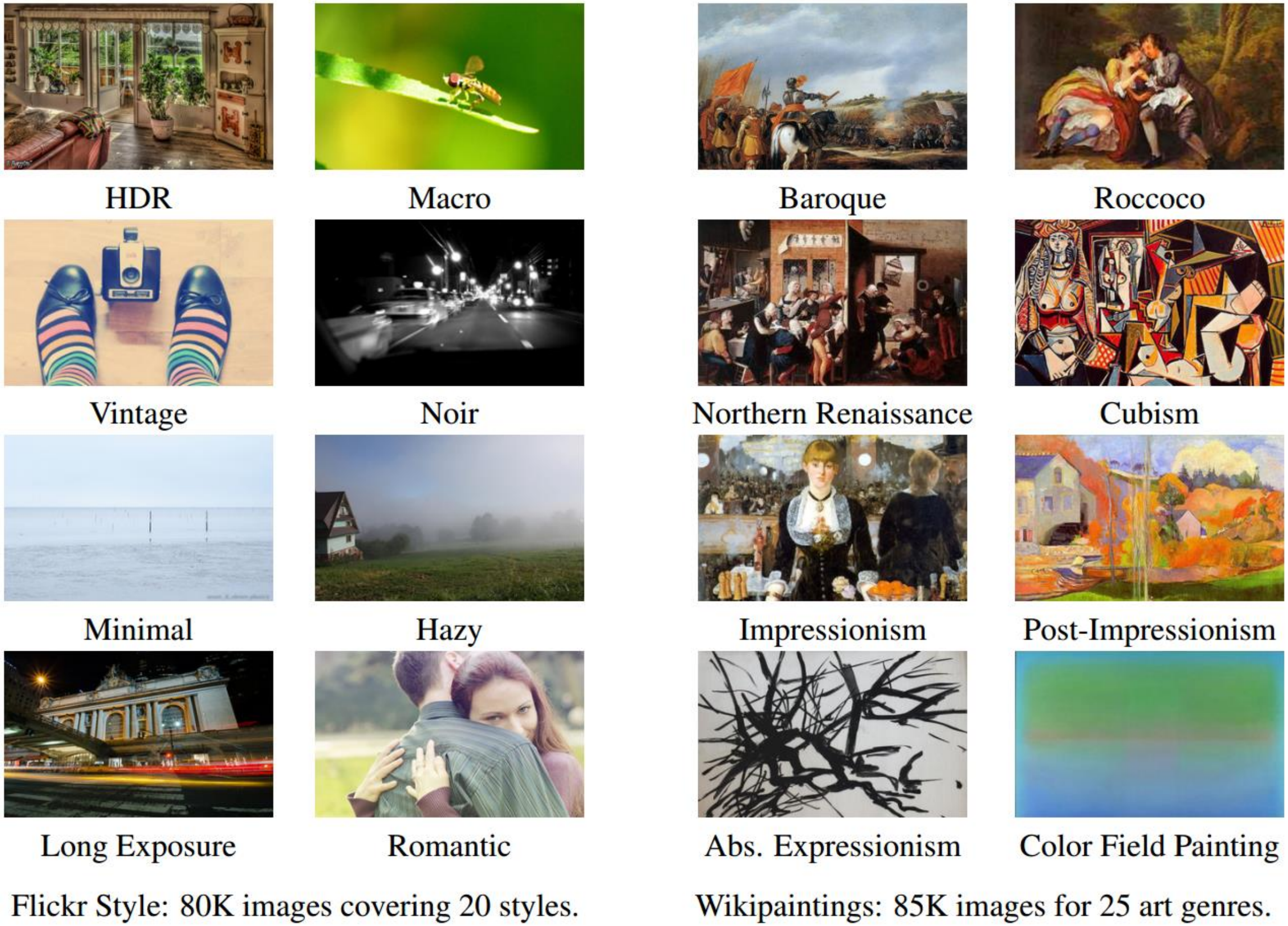
1940 1953 1966 1977

[Palermo et al. ECCV 2012]

Slide credit: D. Hoiem

Examples of image classification

- Image style recognition



[Karayev et al. BMVC 2014]

Slide credit: D. Hoiem

Recognition: A machine learning approach

The machine learning framework

- Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

f() = “tomato”

f() = “cow”

Slide credit: L. Lazebnik

The machine learning framework

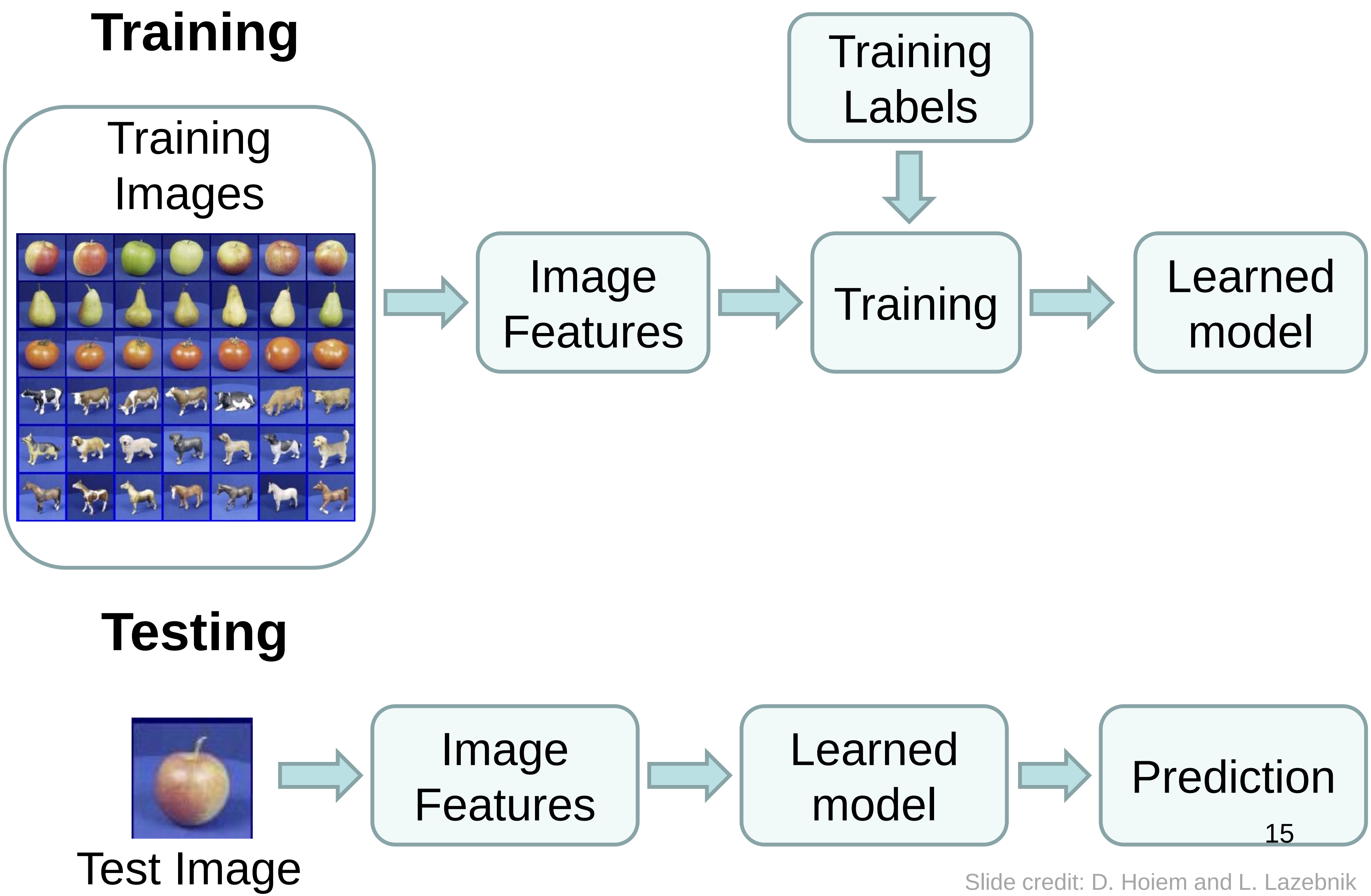
$$y^* = f(\mathbf{x})$$

output (may differ from ground-truth label y) — prediction function — image / image features

- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set, e.g. $|f(\mathbf{x}_i) - y_i|$
  - Evaluate multiple hypotheses f_1, f_2, f_H … and pick the best one as f
- **Testing:** apply f to a never-before-seen *test example* **x** and output the predicted value $y^* = f(\mathbf{x})$

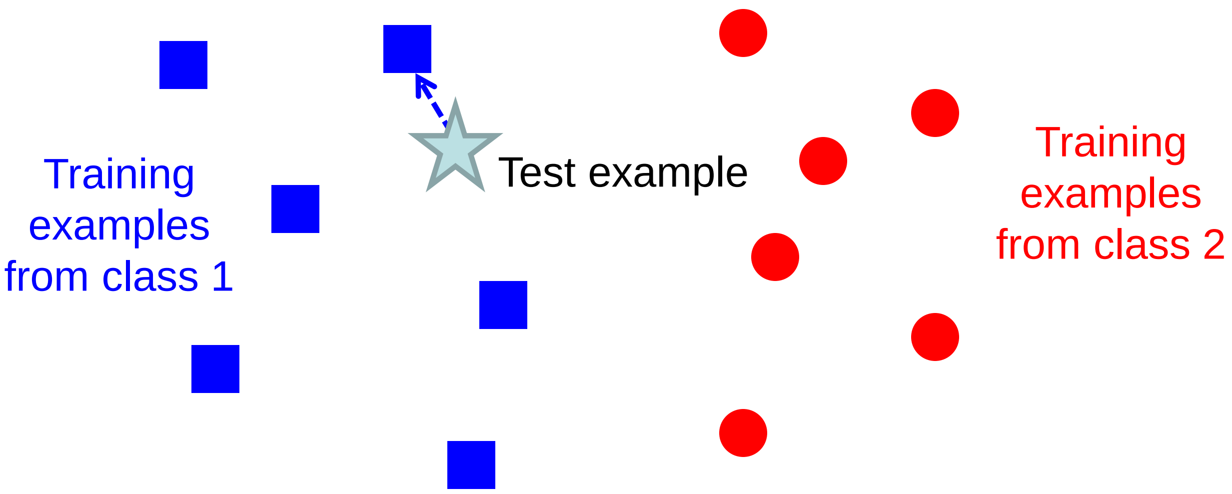
Slide credit: L. Lazebnik

The old-school way



Slide credit: D. Hoiem and L. Lazebnik

The simplest classifier



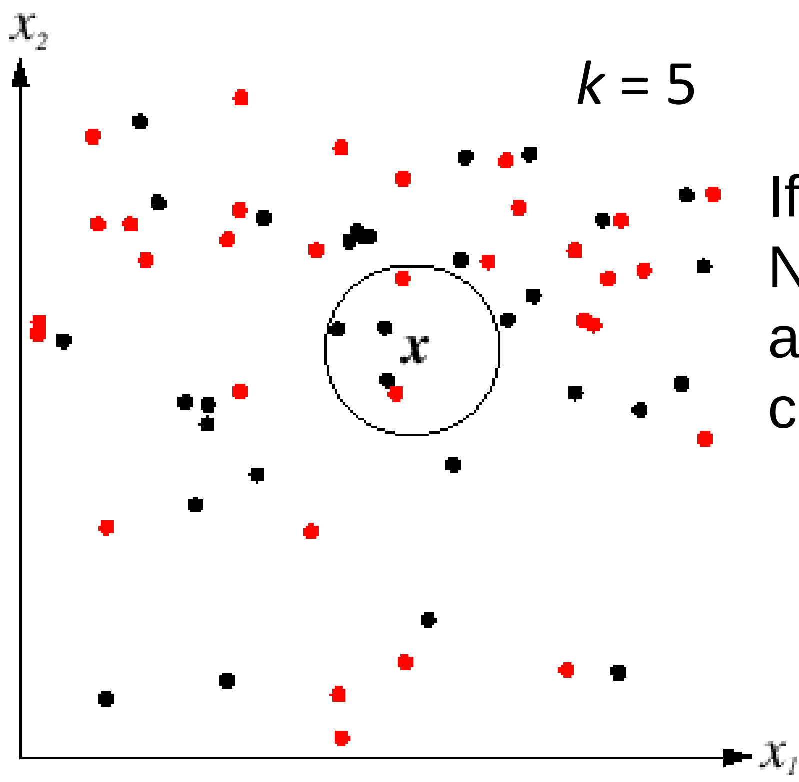
f(**x**) = label of the training example nearest to **x**

- All we need is a distance function for our inputs
- No training required!

K-Nearest Neighbors classification

- For a new point, find the *k* closest points from training data
- Labels of the *k* points "vote" to classify

Black = positive
Red = negative



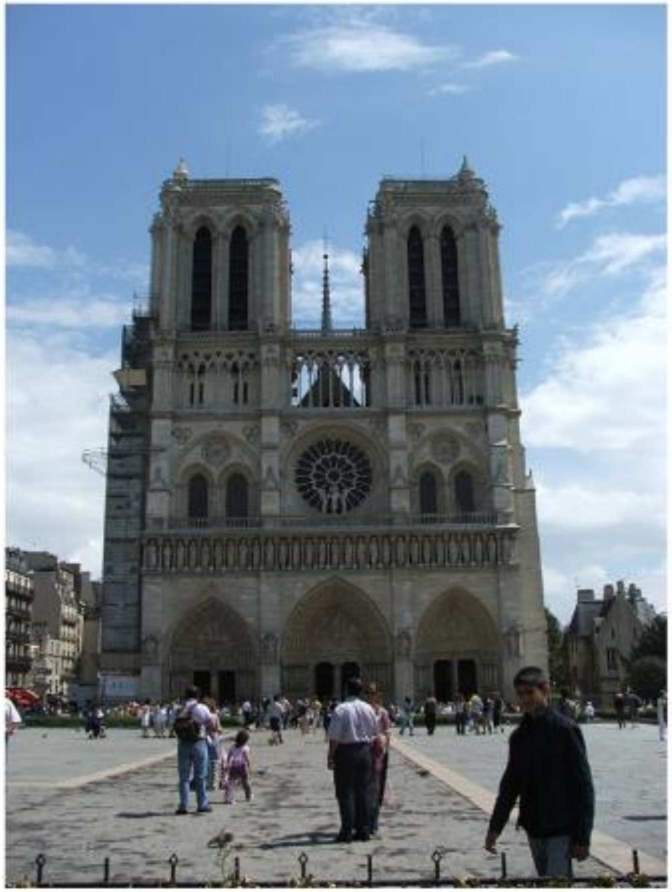
If query lands here, the 5 NN consist of 3 positives and 2 negatives, so we classify it as positive.

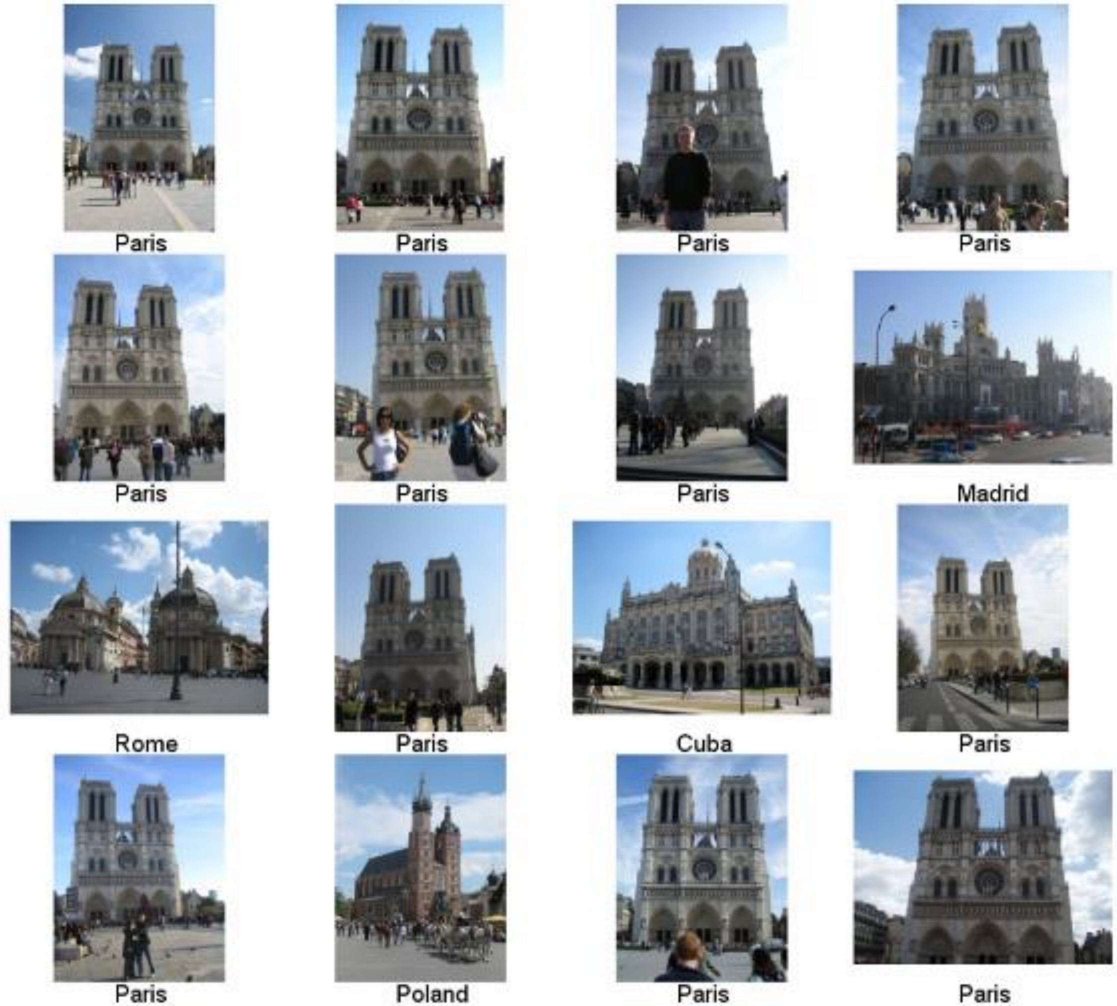
Slide credit: D. Lowe

im2gps: Estimating Geographic Information from a Single Image

James Hays and Alexei Efros, CVPR 2008

Where was this image taken?

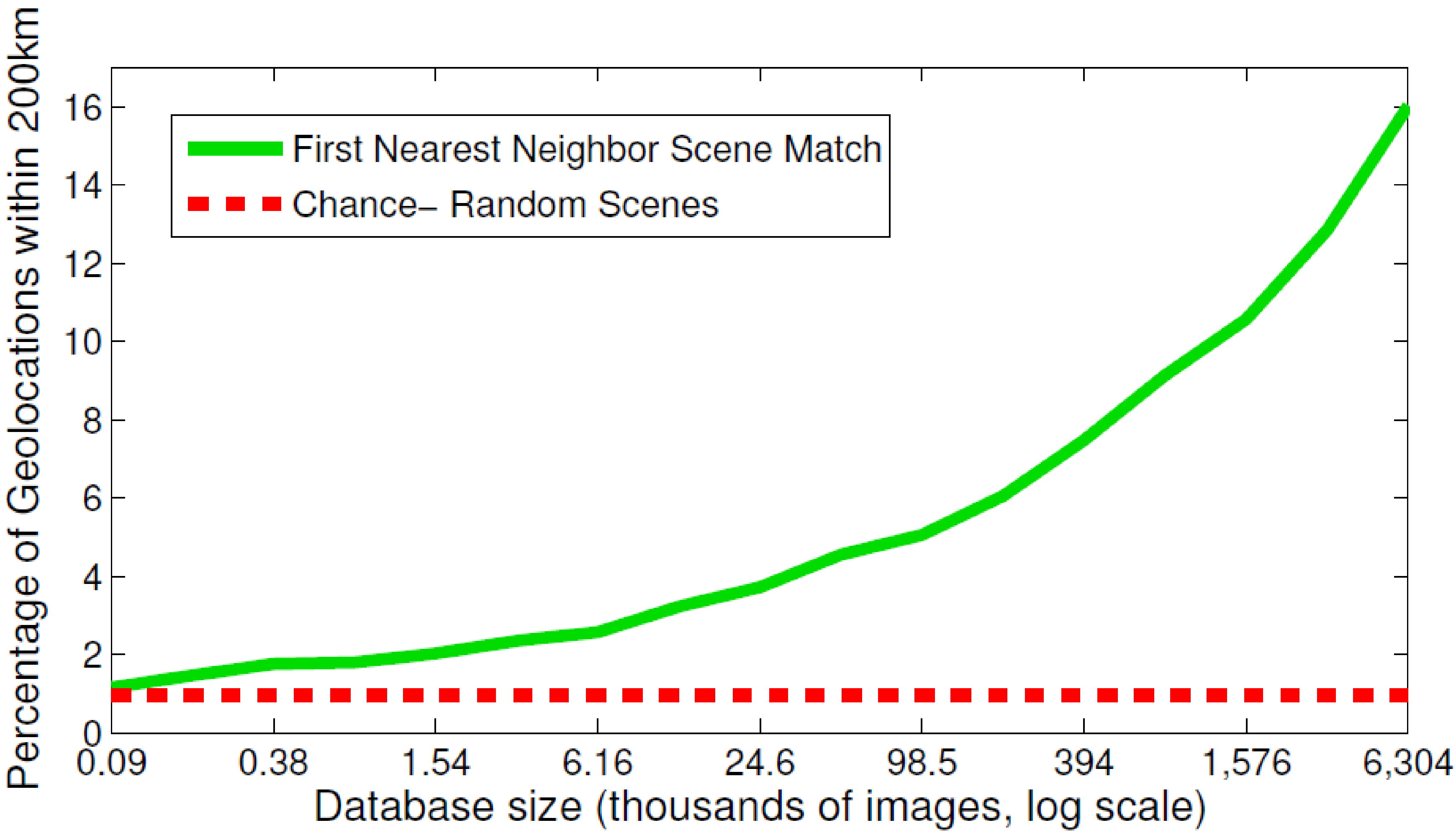




Nearest Neighbors according to BOW-SIFT + color histogram + a few others

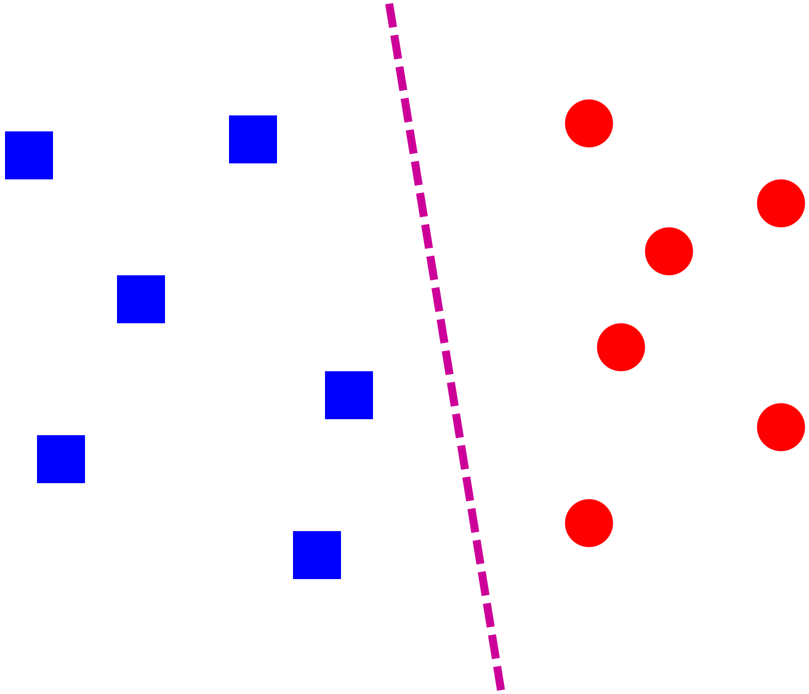
Slide credit: James Hays

The Importance of Data



Slides: James Hays

Linear classifier



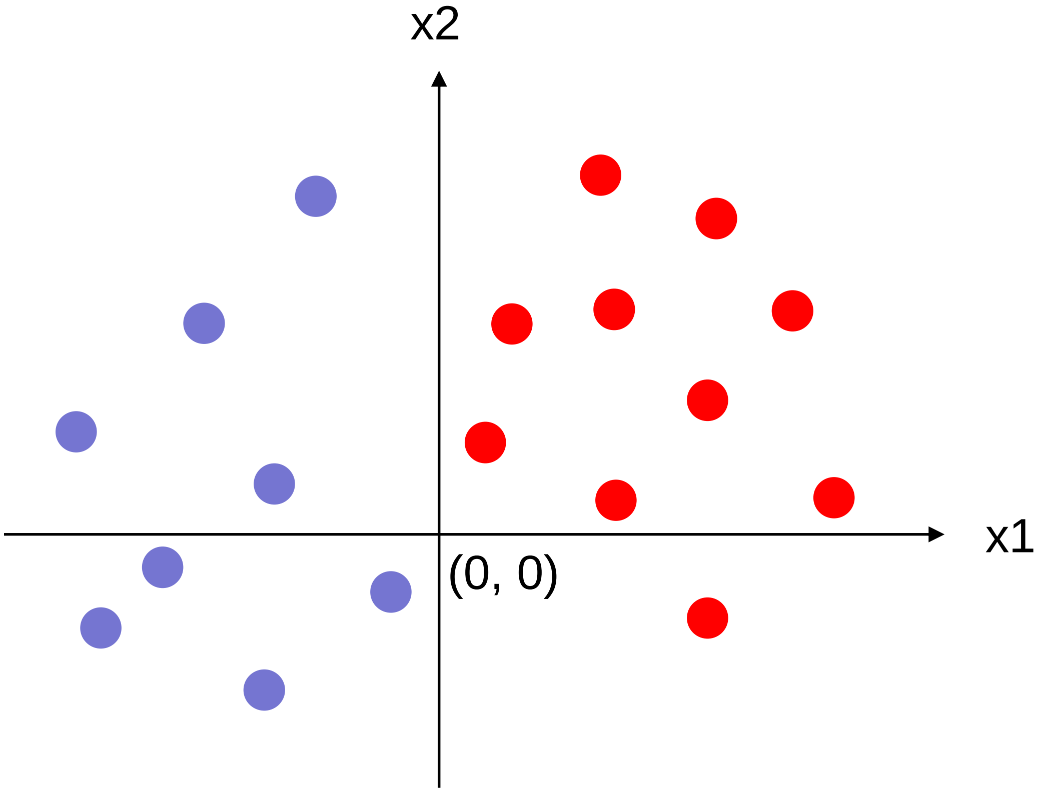
- Find a *linear function* to separate the classes

$$f(\mathbf{x}) = \text{sgn}(w_1x_1 + w_2x_2 + \ldots + w_Dx_D) = \text{sgn}(\mathbf{w} \cdot \mathbf{x})$$

Slide credit: L. Lazebnik

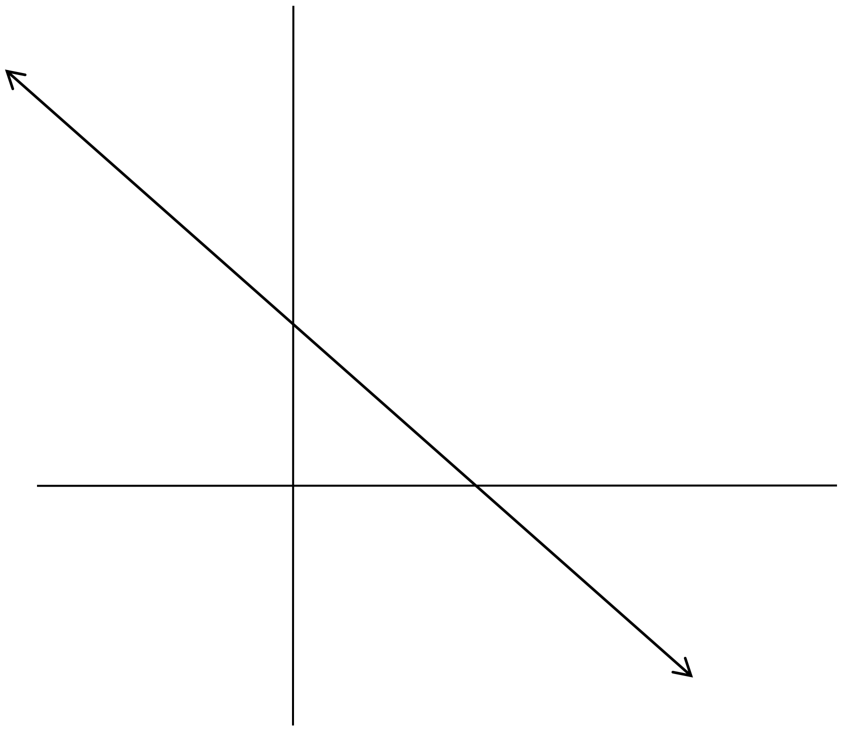
Linear classifier

- Decision = sign($\mathbf{w}^T\mathbf{x}$) = sign(w1*x1 + w2*x2)



- What should the weights be?

Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

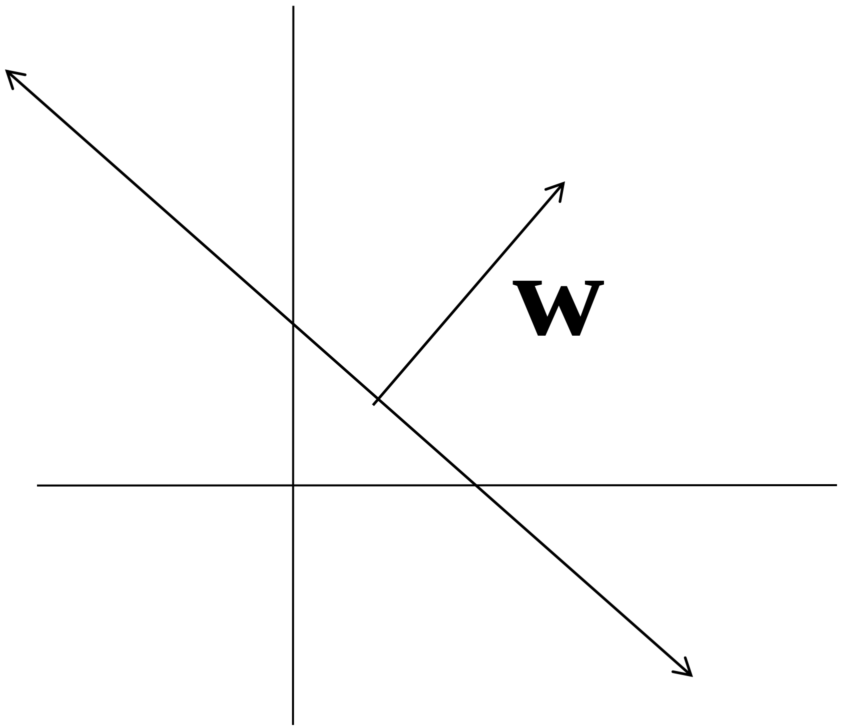
Compare to:
*slope**x + *y-intercept* = y

ax + b = -cy
(-a/c) x + (-b/c) = y

Slope: -a/c
Y-intercept: -b/c

Slope: -a/c
Y-intercept: -b/c

Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

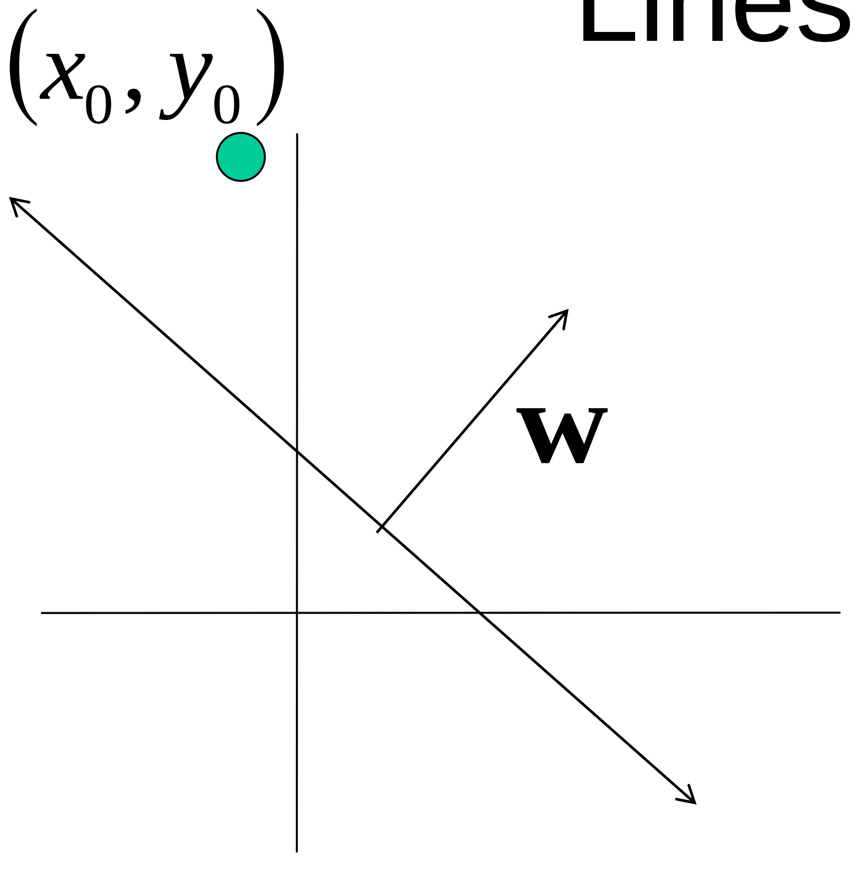
$$\Updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Kristen Grauman

Lines in R^2

Slope: -a/c
Y-intercept: -b/c



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

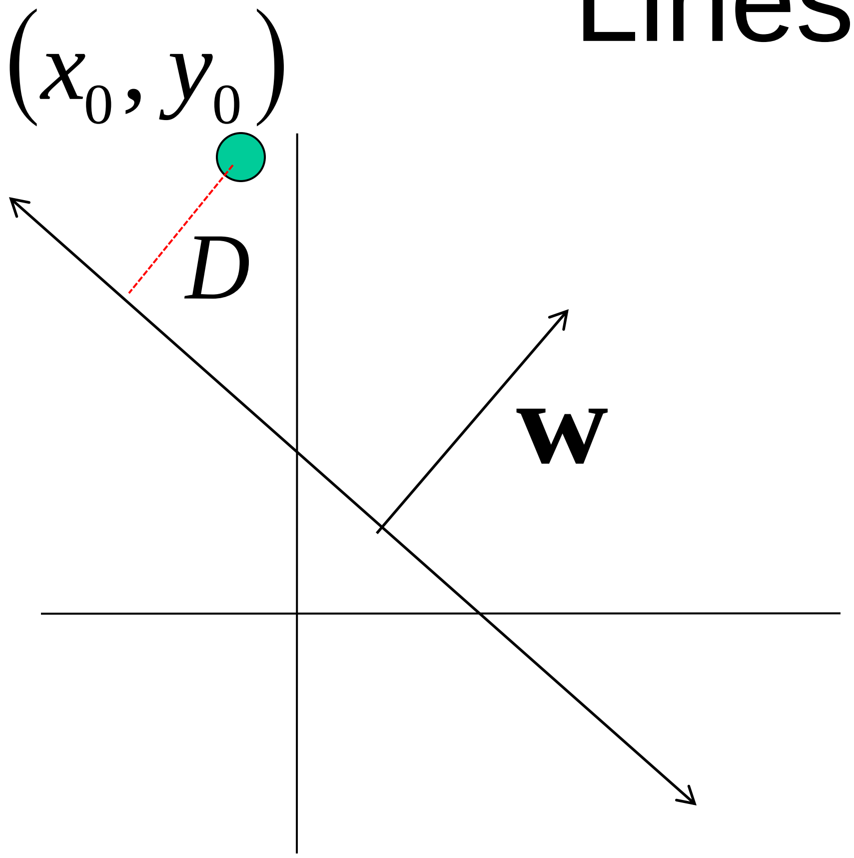
$$ax + cy + b = 0$$

$$\Updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Slope: -a/c
Y-intercept: -b/c

Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\Updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

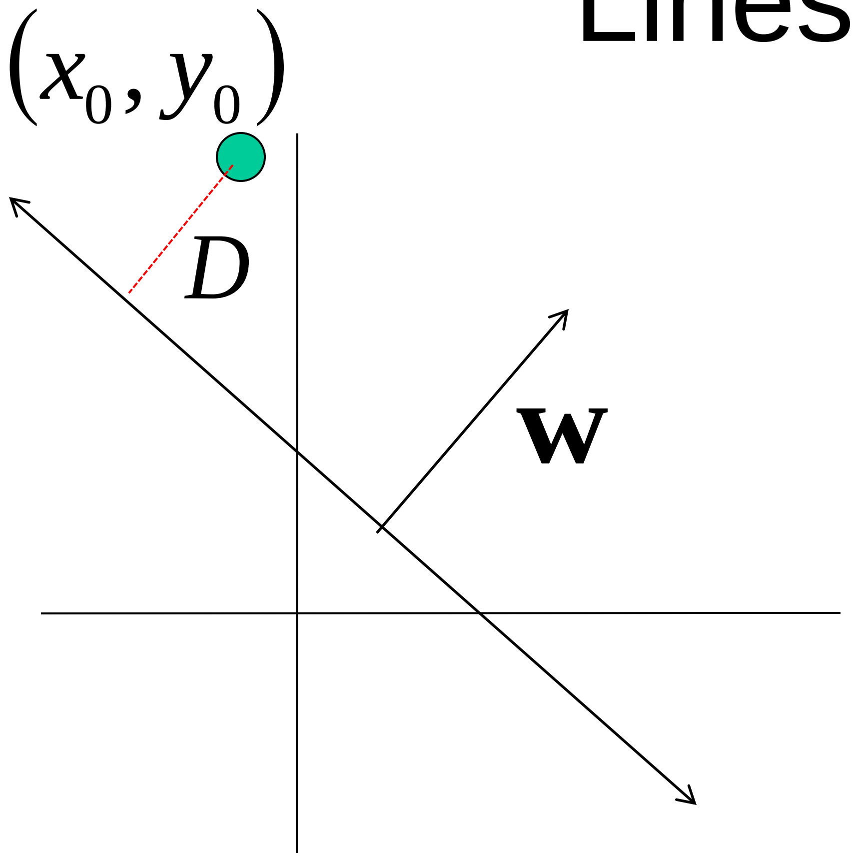
$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}}$$

} distance from point to line

Kristen Grauman

Slope: -a/c
Y-intercept: -b/c

Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\Updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

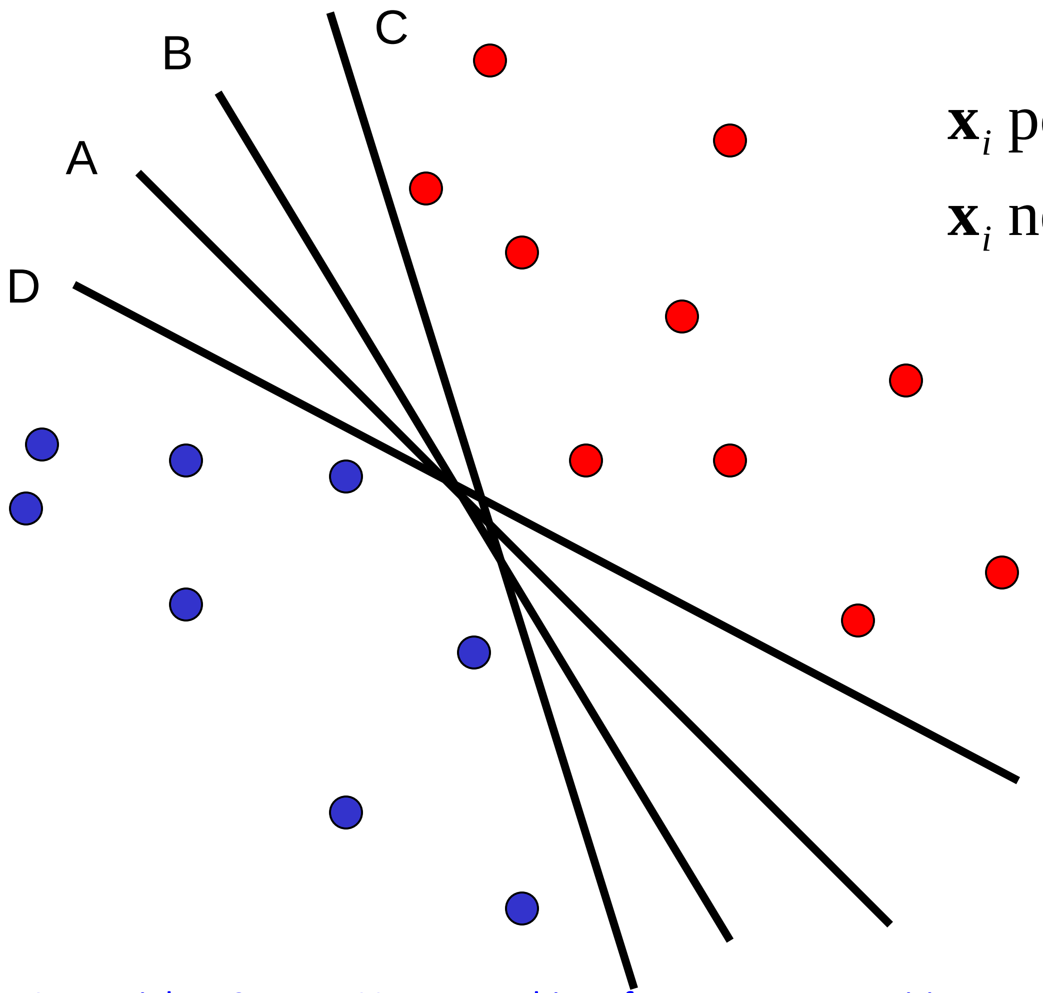
$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} = \frac{|\mathbf{w}^{\mathrm{T}}\mathbf{x} + b|}{\|\mathbf{w}\|}$$

distance from point to line

Adapted from Kristen Grauman

Linear classifiers

- Find linear function to separate positive and negative examples



$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

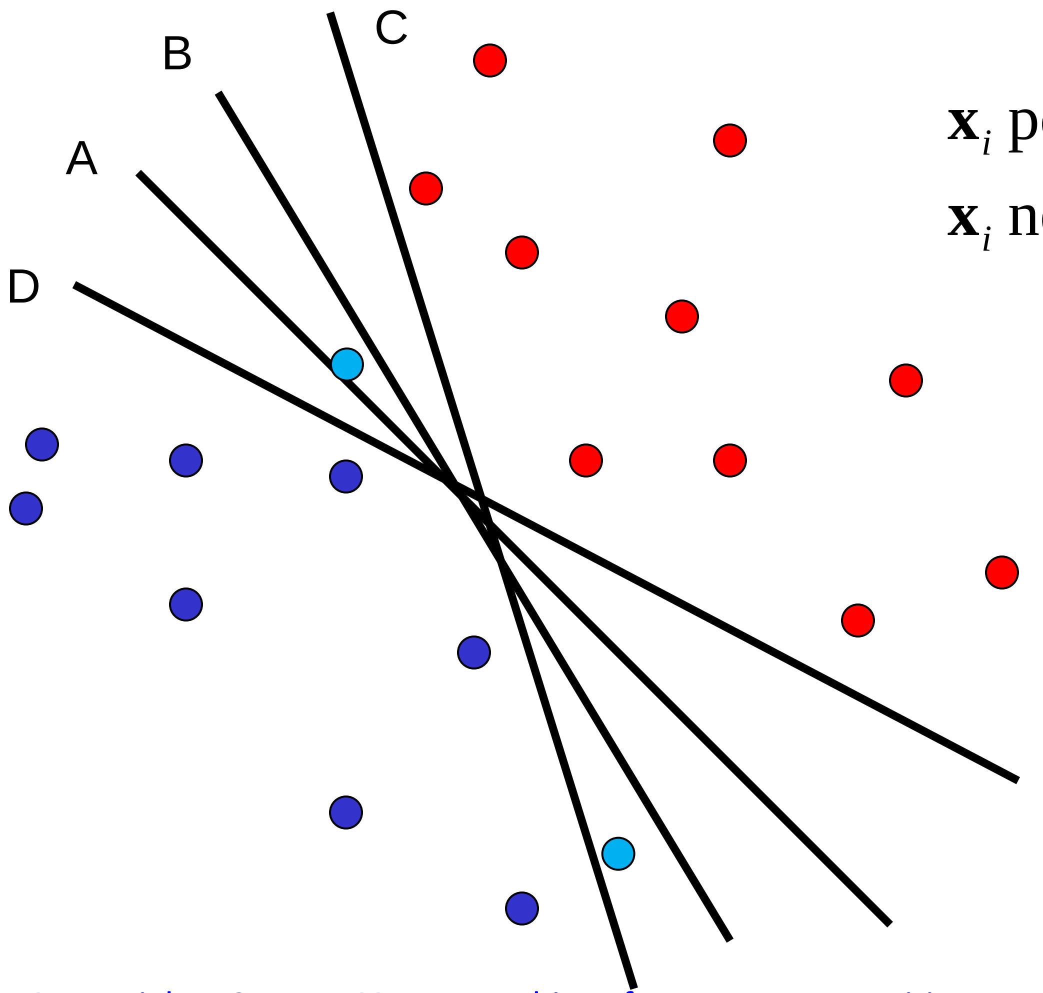
$$\mathbf{x}_i \text{ negative}: \quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

Which line is best?

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Linear classifiers

- Find linear function to separate positive and negative examples



$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

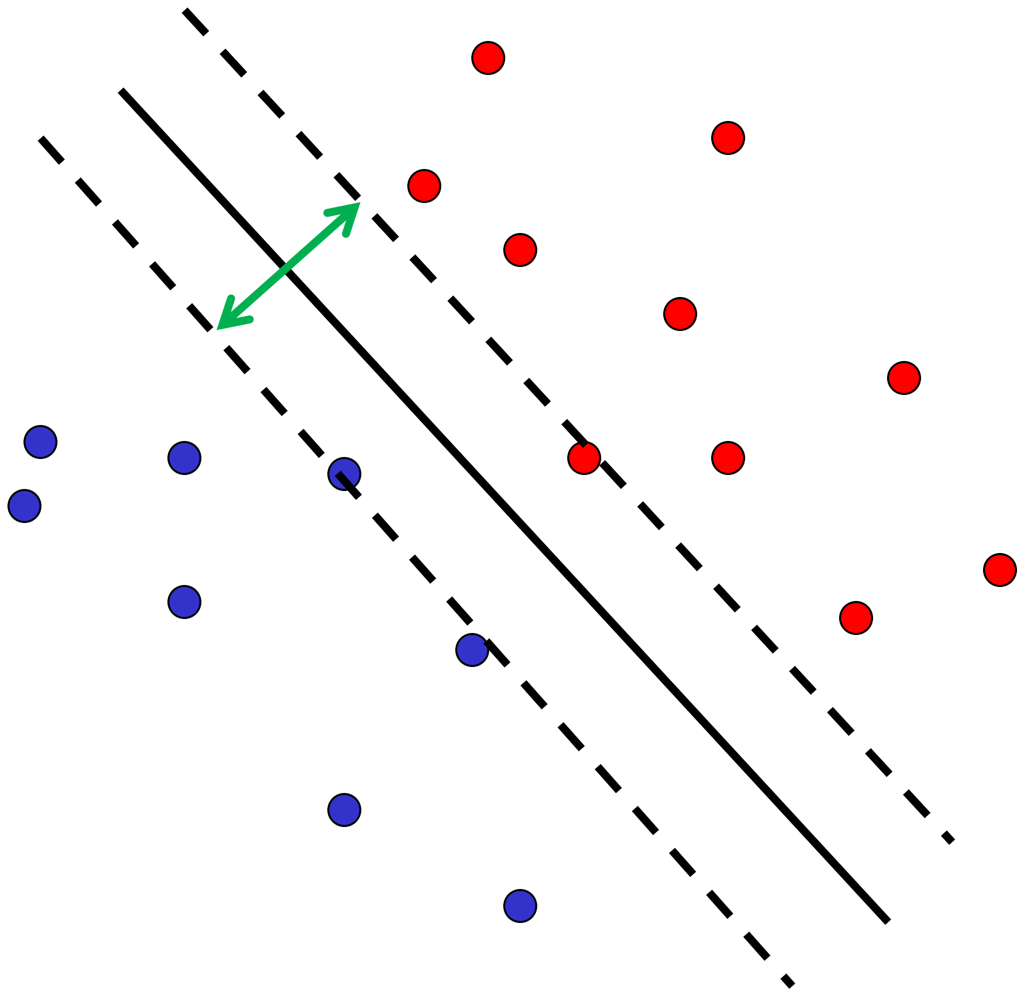
$$\mathbf{x}_i \text{ negative}: \quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

Not seen until test time, of class blue

Which line is best?

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support vector machines

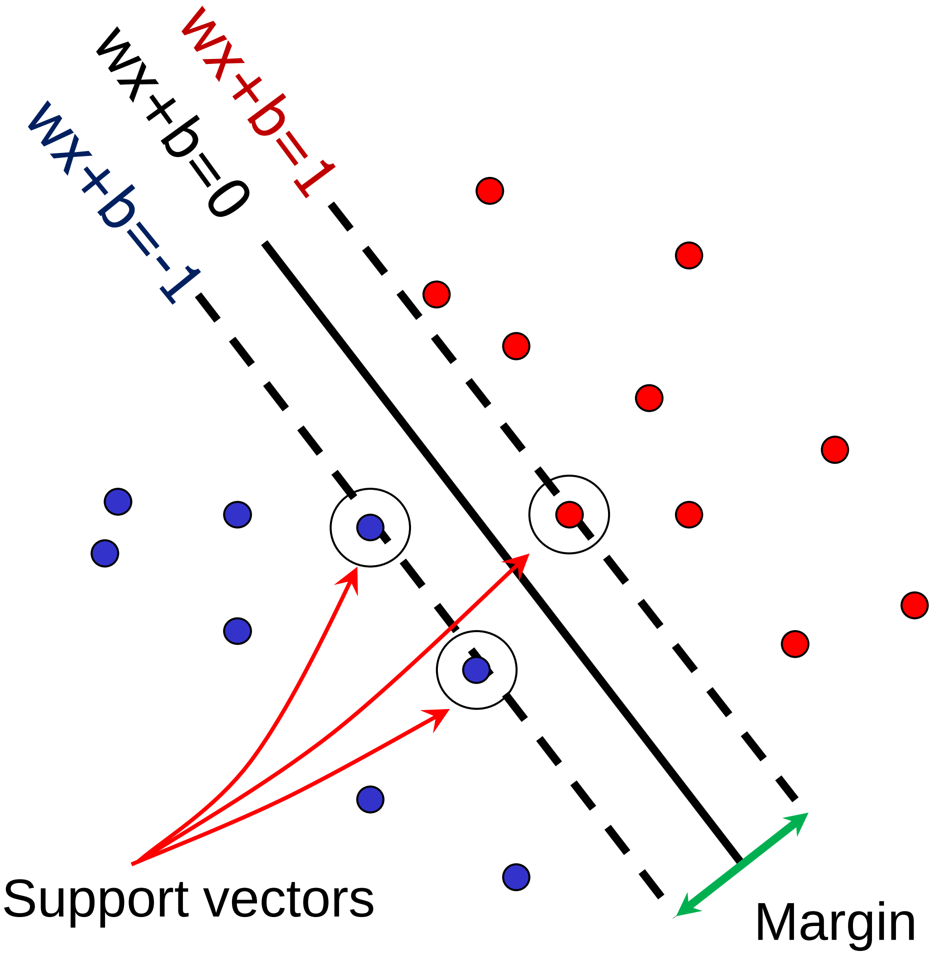


- Discriminative classifier based on *optimal separating line (for 2d case)*
- Maximize the *margin* between the positive and negative training examples

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support vector machines

- Want line that maximizes the margin.



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

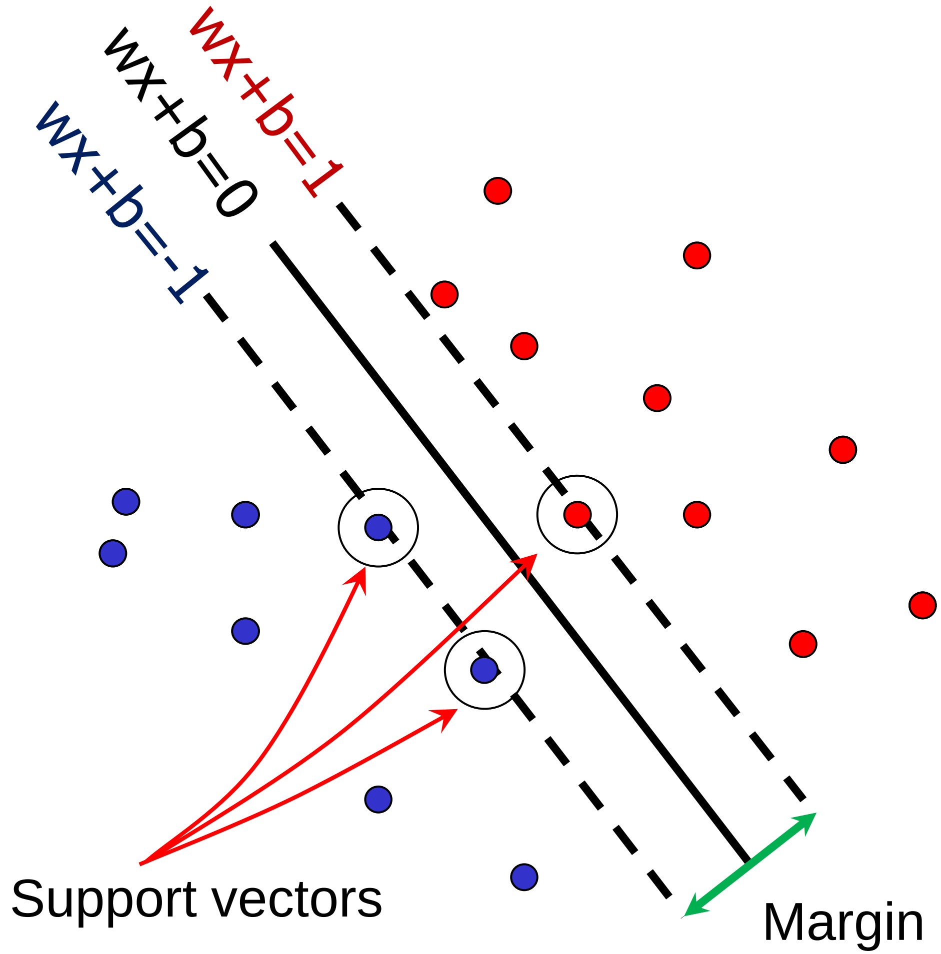
$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support, vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support vector machines

- Want line that maximizes the margin.



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support, vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and line: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{||\mathbf{w}||}$

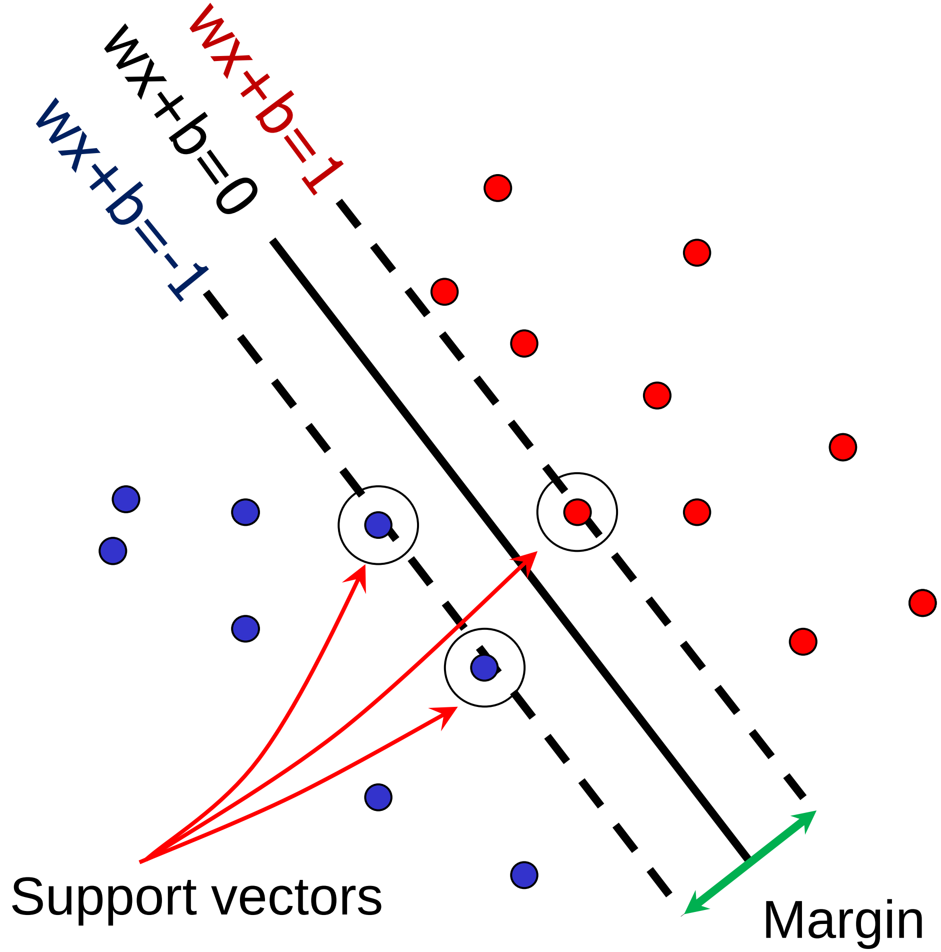
For support vectors:

$$\frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|} = \frac{\pm 1}{\|\mathbf{w}\|} \qquad M = \left| \frac{1}{\|\mathbf{w}\|} - \frac{-1}{\|\mathbf{w}\|} \right| = \frac{2}{\|\mathbf{w}\|}$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support vector machines

- Want line that maximizes the margin.



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support, vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and line: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{||\mathbf{w}||}$

Therefore, the margin is $2 / ||\mathbf{w}||$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin line

1. Maximize margin $2/||\mathbf{w}||$
2. Correctly classify all training data points:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

Quadratic optimization problem:

Minimize $\frac{1}{2}\mathbf{w}^T\mathbf{w}$

Subject to $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$

One constraint per training point.

Note sign trick:
$w \cdot x_i + b >= 1$ (if $y_i = 1$)

$w \cdot x_i + b <= -1$ (if $y_i = -1$)
$(-1)\ w \cdot x_i + b >= 1$

Adapted from C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition

Finding the maximum margin line

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

Learned weight

Support vector

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin line

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

  $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)

- Classification function:

$$f(x) = \text{sign}\,(\mathbf{w} \cdot \mathbf{x} + \mathrm{b})$$
$$= \text{sign}\left(\sum_{\mathrm{i}} \alpha_{\mathrm{i}} y_i \mathbf{x}_i \cdot \mathbf{x} + b\right)$$

If f(x) < 0, classify as negative, otherwise classify as positive.

- Notice that it relies on an *inner product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x}_i$
- (Solving the optimization problem also involves computing the inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ between all pairs of training points)

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Inner product

- **The decision boundary for the SVM and its optimization depend on the inner product of two data points (vectors):**

$$\mathbf{x}_i^T \mathbf{x}_j$$

$$f(x) = \text{sign}\,(\mathbf{w}\cdot\mathbf{x} + \text{b})$$
$$= \text{sign}\left(\sum_{\text{i}} \alpha_{\text{i}} y_i \mathbf{x}_i \cdot \mathbf{x} + b\right)$$

- **The inner product is equal**

$$(\mathbf{x}_i^T \mathbf{x}) = \|\mathbf{x}_i\| * \|\mathbf{x}_i\| \cos\theta$$

If the angle in between them is 0 then: $(\mathbf{x}_i^T \mathbf{x}) = \|\mathbf{x}_i\| * \|\mathbf{x}_i\|$

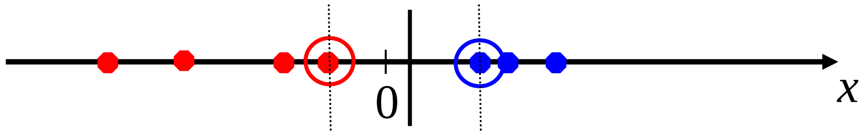
If the angle between them is 90 then: $(\mathbf{x}_i^T \mathbf{x}) = 0$

The inner product measures how similar the two vectors are

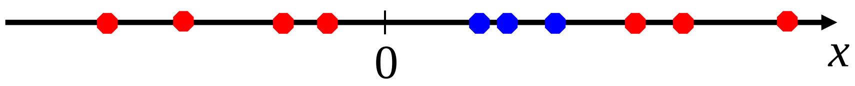
Adapted from Milos Hauskrecht

Nonlinear SVMs

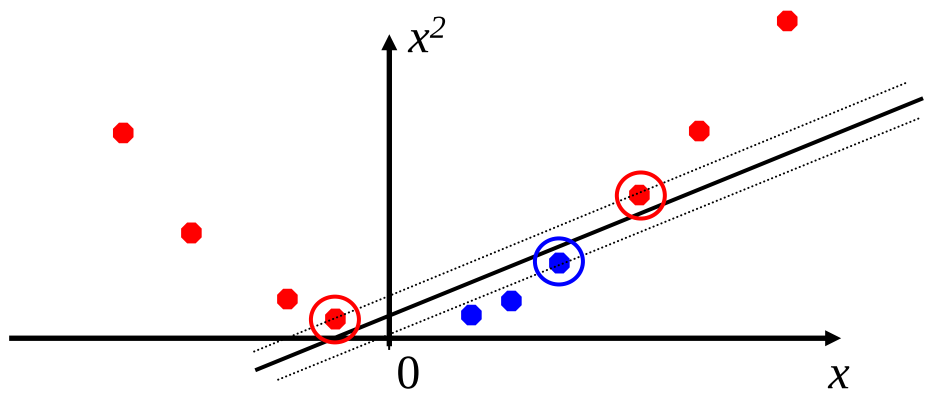
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



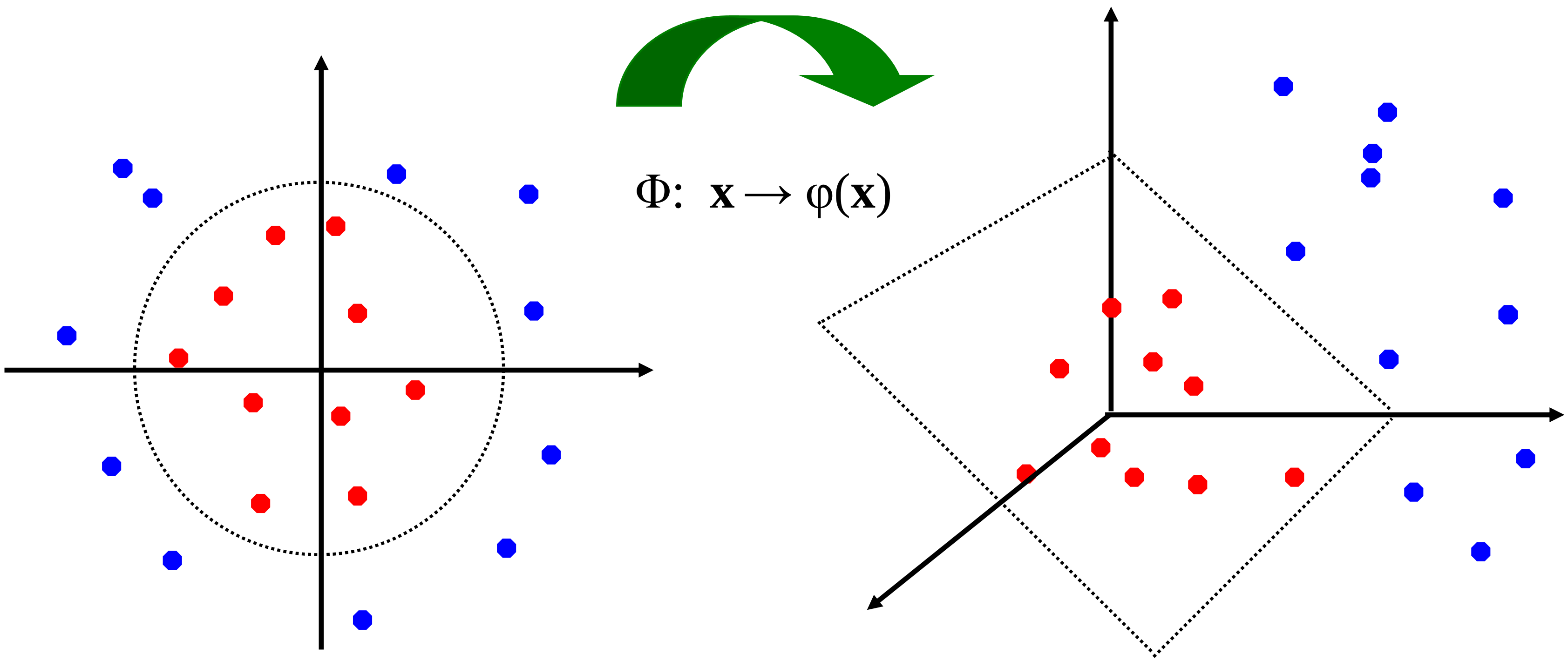
- We can map it to a higher-dimensional space:



Andrew Moore

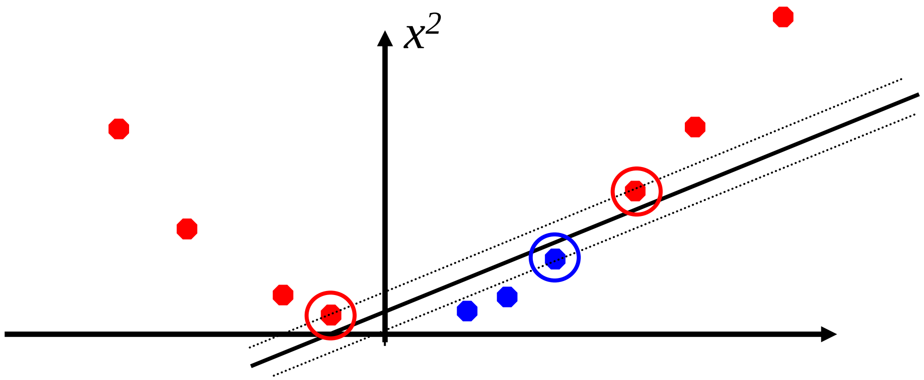
Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Nonlinear kernel: Example

- Consider the mapping $\varphi(x) = (x, x^2)$



$$\varphi(x) \cdot \varphi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2 y^2$$

$$K(x, y) = xy + x^2 y^2$$

Svetlana Lazebnik

The "Kernel Trick"

- The linear classifier relies on dot product between vectors $K(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
- If every data point is mapped into high-dimensional space via some transformation Φ: $\mathbf{x}_i \rightarrow \varphi(\mathbf{x}_i)$, the dot product becomes: $K(\mathbf{x}_i,\mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$
- A *kernel function* is similarity function that corresponds to an inner product in some expanded feature space
- *The kernel trick*: instead of explicitly computing the lifting transformation $\varphi(\mathbf{x})$, define a kernel function K such that: $K(\mathbf{x}_i,\mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$

Examples of kernel functions

- Linear: $K(x_i, x_j) = x_i^T x_j$
- Polynomials of degree up to *d*:

$$K(x_i, x_j) = (x_i^T x_j + 1)^d$$

- Gaussian RBF:

$$K(x_i, x_j) = \exp(-\frac{\left\| x_i - x_j \right\|^2}{2\sigma^2})$$

- Histogram intersection:

$$K(x_i, x_j) = \sum_k \min(x_i(k), x_j(k))$$

The benefit of the “kernel trick”

- Example: Polynomial kernel for 2-dim features

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= \left(1 + \mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2 = (1 + x_1 z_1 + x_2 z_2)^2 \\ &= 1 + 2x_1 z_1 + 2x_2 z_2 + x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2)(1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, \sqrt{2}z_1 z_2, z_2^2)^{\mathrm{T}} \\ &= \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{z}). \end{aligned} \tag{7.42}$$

- … lives in 6 dimensions
- With the kernel trick, we directly compute an inner product in 2-dim space, obtaining a scalar that we add 1 to and exponentiate

Hard-margin SVMs

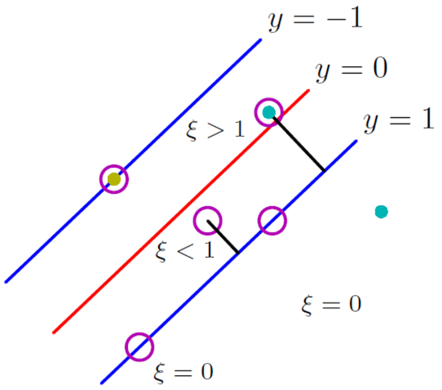
$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2$$

The *w* that minimizes…

Maximize margin

$$\text{subject to} \quad y_i \boldsymbol{w}^T \boldsymbol{x}_i \geq 1 \qquad , \quad \forall i = 1, \ldots, N$$

Soft-margin SVMs



Misclassification cost

data samples

Slack variable

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

The *w* that minimizes…

Maximize margin

Minimize misclassification

$$\text{subject to} \quad y_i \boldsymbol{w}^T \boldsymbol{x}_i \geq 1 - \xi_i,$$

$$\xi_i \geq 0, \quad \forall i = 1, \ldots, N$$

Figure from Chris Bishop

Example of choosing optimal **w**

- Suppose **w** is just a scalar, w, that can only take values 1, 2, 3
- Let L(w) denote the sum of ξ_i values incurred when using a particular w
- Suppose L(w=1) = 4, L(w=2) = 5, L(w=3) = 0.5
- Find the optimal w as argmin_w ||w||
    - What is the optimal w?
- Now find the optimal w for L^(w) = ||w|| + L(w)
    - L^(1) = 1 + 4 = 5
    - L^(2) = ? L^(3) = ?
    - Now what is the optimal w?

What about multi-class SVMs?

- Unfortunately, there is no “definitive” multi-class SVM formulation
- In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs
- One vs. others/all
  - Training: learn an SVM for each class vs. the others
  - Testing: apply each SVM to the test example, and assign it to the class of the SVM that returns the highest decision value
- One vs. one
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM “votes” for a class to assign to the test example

Svetlana Lazebnik

Multi-class problems

One-vs-all (a.k.a. one-vs-others)

- Train K classifiers
- In each, pos = data from class *i*, neg = data from classes other than *i*
- The class with the most confident prediction wins
- Example:
  - You have 4 classes, train 4 classifiers
  - 1 vs others: score 3.5
  - 2 vs others: score 6.2
  - 3 vs others: score 1.4
  - 4 vs other: score 5.5
  - Final prediction: class 2

Multi-class problems

One-vs-one (a.k.a. all-vs-all)

- Train K(K-1)/2 binary classifiers (all pairs of classes)
- They all vote for the label
- Example:
  - You have 4 classes, then train 6 classifiers
  - 1 vs 2, 1 vs 3, 1 vs 4, 2 vs 3, 2 vs 4, 3 vs 4
  - Votes: 1, 1, 4, 2, 4, 4
  - Final prediction is class 4

Using SVMs

1. Select a kernel function.
2. Compute pairwise kernel values between labeled examples.
3. Use this “kernel matrix” to solve for SVM support vectors & alpha weights.
4. To classify a new example: compute kernel values between new input and support vectors, apply alpha weights, check sign of output.

Adapted from Kristen Grauman

Some SVM packages

- LIBSVM http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- LIBLINEAR https://www.csie.ntu.edu.tw/~cjlin/liblinear/
- SVM Light http://svmlight.joachims.org/

Linear classifiers vs nearest neighbors

- Linear pros:
  - + Low-dimensional *parametric* representation
  - + Very fast at test time
- Linear cons:
  - – Can be tricky to select best kernel function for a problem
  - – Learning can take a very long time for large-scale problem
- NN pros:
  - + Works for any number of classes
  - + Decision boundaries not necessarily linear
  - + *Nonparametric* method
  - + Simple to implement
- NN cons:
  - – Slow at test time (large search problem to find neighbors)
  - – Storage of data
  - – Especially need good distance function (but true for all classifiers)

Adapted from L. Lazebnik

Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories

CVPR 2006

Winner of 2016 Longuet-Higgins Prize

Svetlana Lazebnik (slazebni@uiuc.edu)
Beckman Institute, University of Illinois at Urbana-Champaign

Cordelia Schmid (cordelia.schmid@inrialpes.fr)
INRIA Rhône-Alpes, France

Jean Ponce (ponce@di.ens.fr)
Ecole Normale Supérieure, France

Scene category dataset

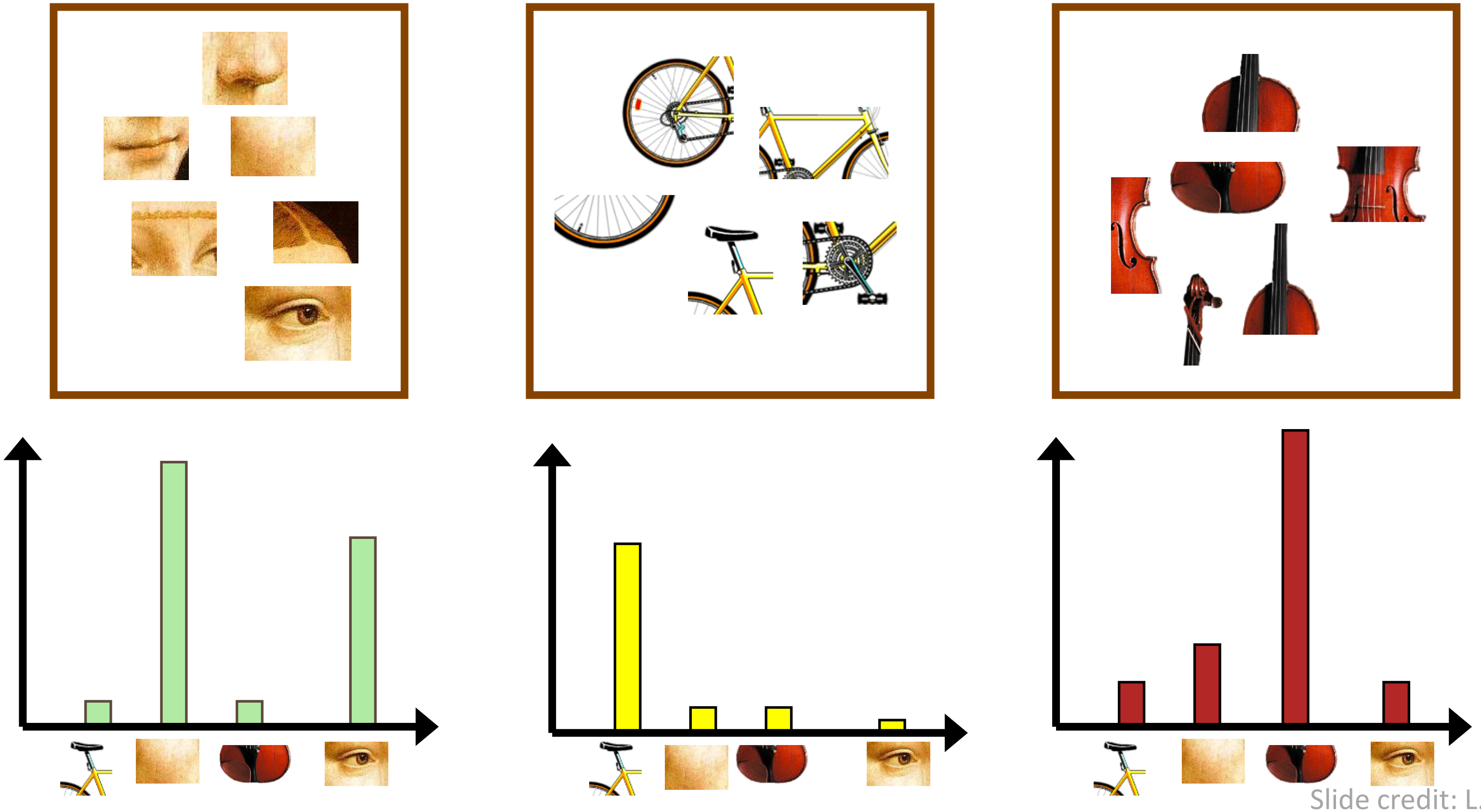
Fei-Fei & Perona (2005), Oliva & Torralba (2001)

http://www-cvr.ai.uiuc.edu/ponce_grp/data

Slide credit: L. Lazebnik

Bag-of-words representation

1. Extract local features
2. Learn “visual vocabulary” using clustering
3. Quantize local features using visual vocabulary
4. Represent images by frequencies of “visual words”



Slide credit: L. Lazebnik

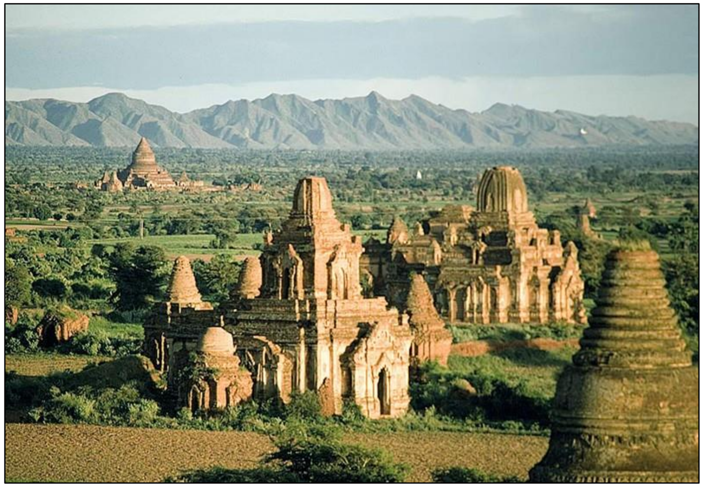
Image categorization with bag of words

Training
1. Compute bag-of-words representation for training images
2. Train classifier on labeled examples using histogram values as features
3. Labels are the scene types (e.g. mountain vs field)

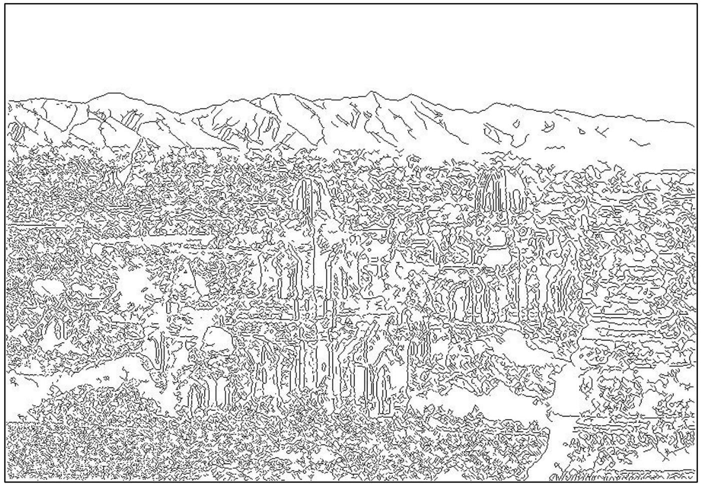
Testing
1. Extract keypoints/descriptors for test images
2. Quantize into visual words using the clusters computed at training time
3. Compute visual word histogram for test images
4. Compute labels on test images using classifier obtained at training time
5. **Evaluation only, do only once:** Measure accuracy of test predictions by comparing them to ground-truth test labels (obtained from humans)

Adapted from D. Hoiem

Feature extraction (on which BOW is based)

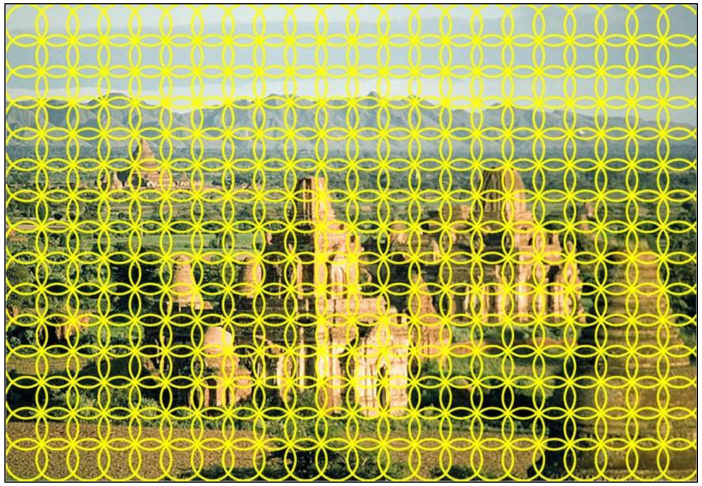


Weak features



Edge points at 2 scales and 8 orientations (vocabulary size 16)

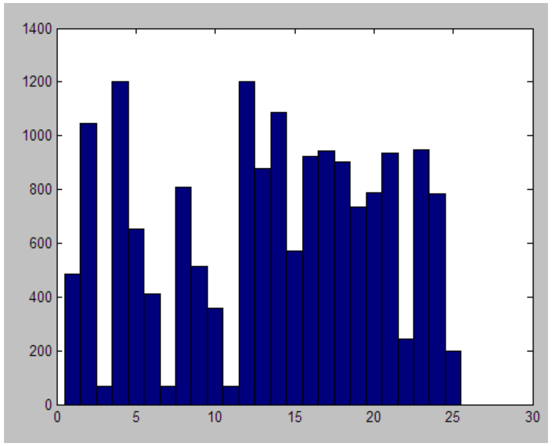
Strong features

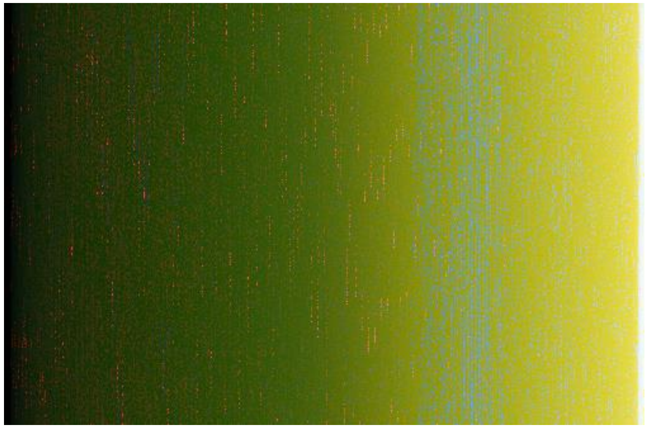


SIFT descriptors of 16x16 patches sampled on a regular grid, quantized to form visual vocabulary (size 200, 400)

Slide credit: L. Lazebnik

What about spatial layout?

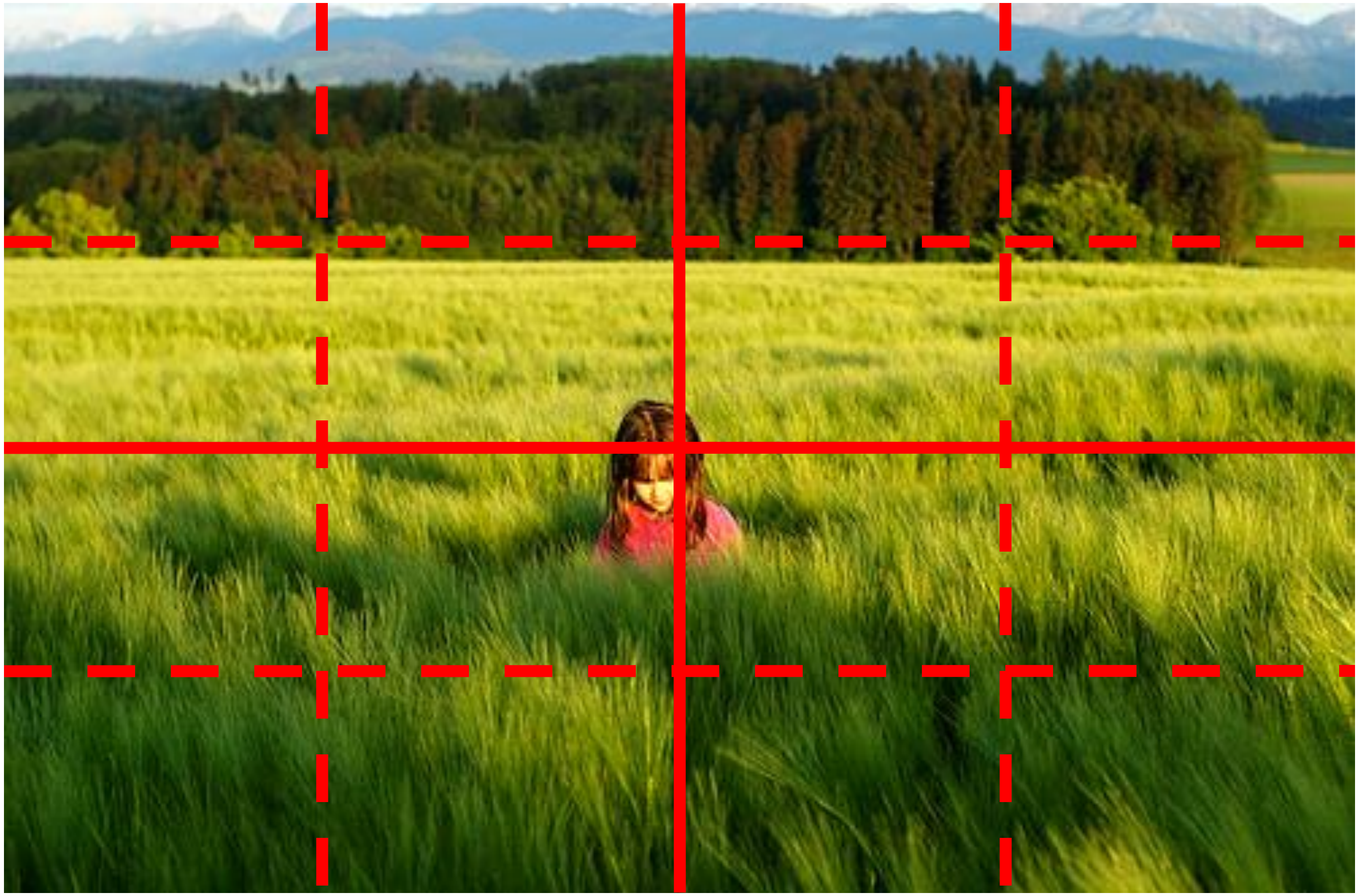




All of these images have the same color histogram

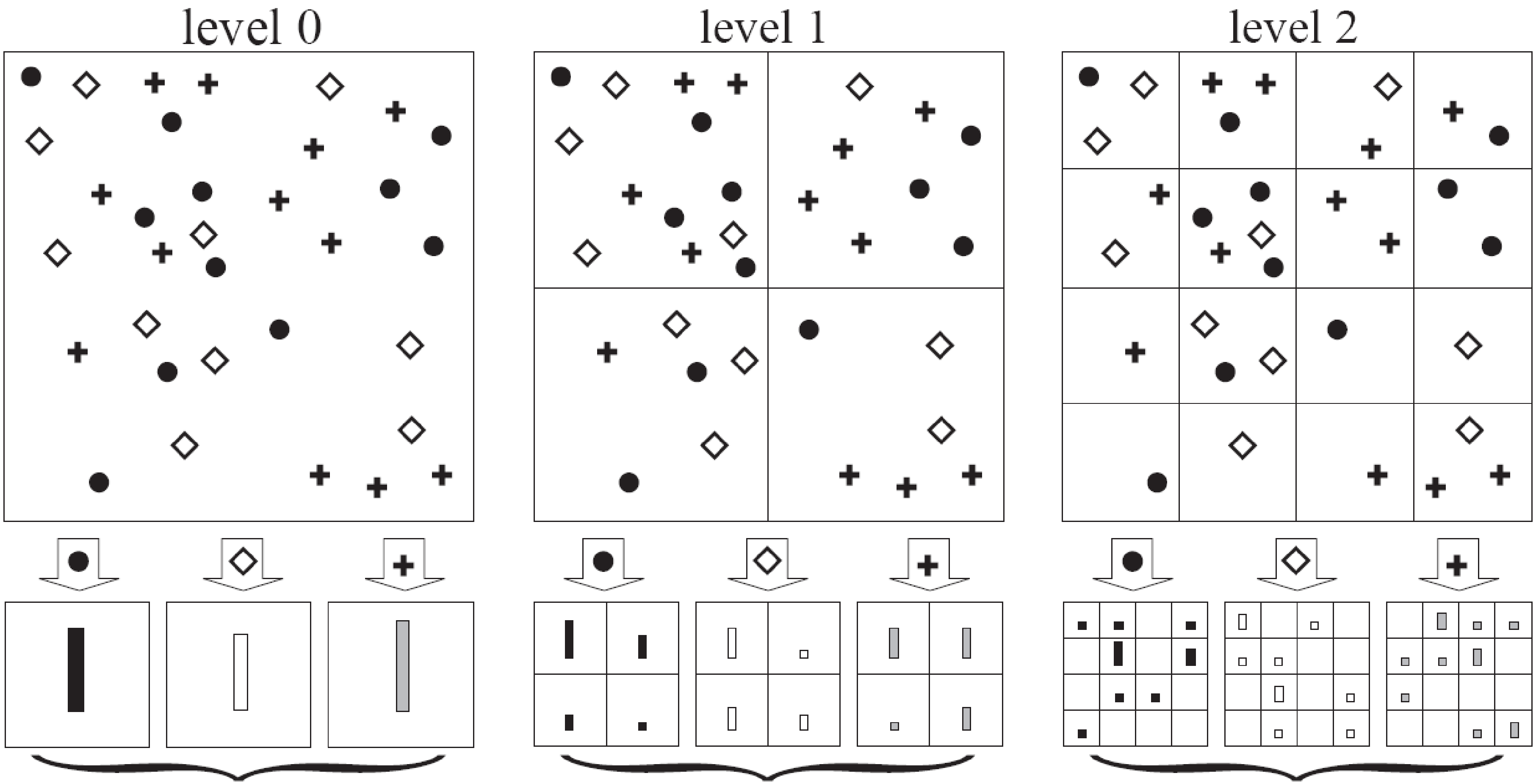
Slide credit: D. Hoiem

Spatial pyramid



Compute histogram in each spatial bin

Spatial pyramid



[Lazebnik et al. CVPR 2006]

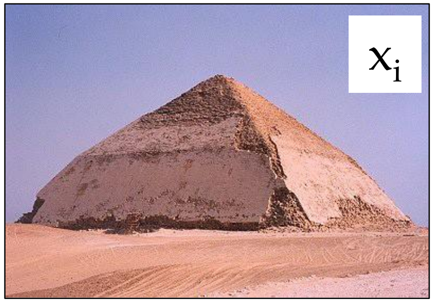
Slide credit: D. Hoiem

Pyramid matching

Indyk & Thaper (2003), Grauman & Darrell (2005)

Matching using pyramid and histogram intersection for some particular visual word:

Original images



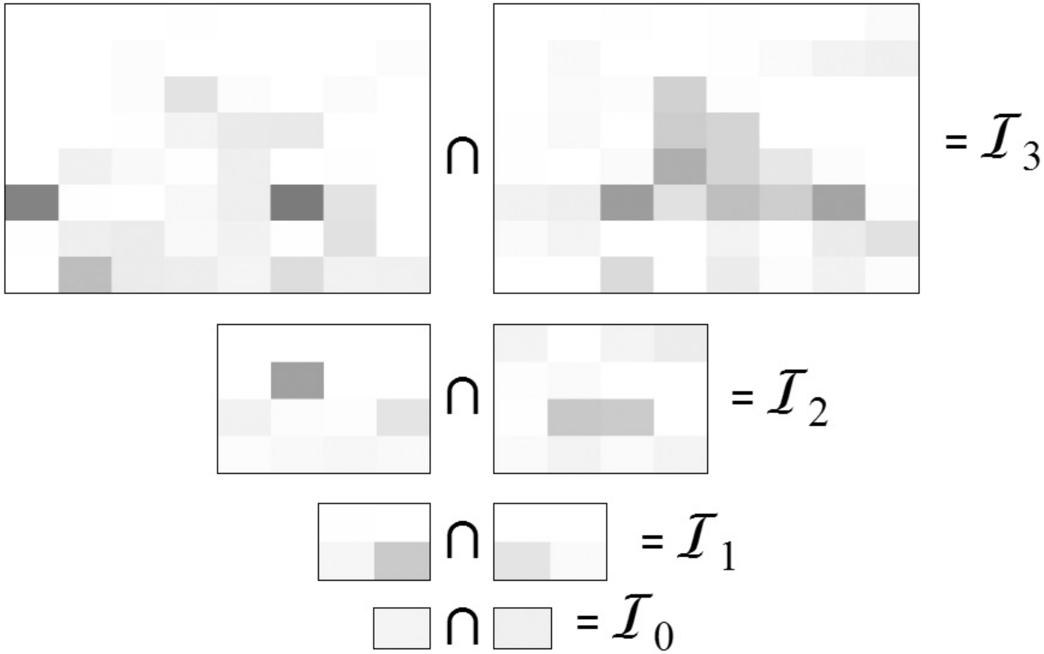
Feature histograms:

Level 3

Level 2

Level 1

Level 0



$K(x_i, x_j)$ (value of *pyramid match kernel*): $\mathcal{I}_3 + \frac{1}{2}(\mathcal{I}_2 - \mathcal{I}_3) + \frac{1}{4}(\mathcal{I}_1 - \mathcal{I}_2) + \frac{1}{8}(\mathcal{I}_0 - \mathcal{I}_1)$

Adapted from L. Lazebnik

Scene category dataset

Fei-Fei & Perona (2005), Oliva & Torralba (2001)

http://www-cvr.ai.uiuc.edu/ponce_grp/data

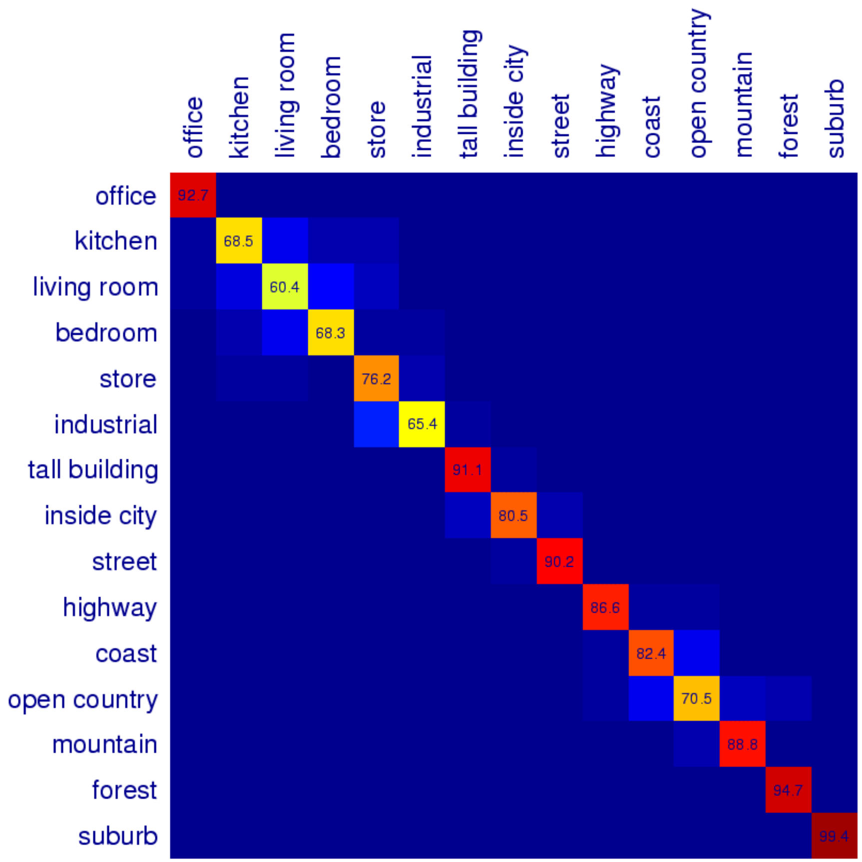
Multi-class classification results (100 training images per class)

	Weak features (vocabulary size: 16)		Strong features (vocabulary size: 200)	
Level	Single-level	Pyramid	Single-level	Pyramid
0 (1×1)	45.3 ± 0.5		72.2 ± 0.6	
1 (2×2)	53.6 ± 0.3	56.2 ± 0.6	77.9 ± 0.6	79.0 ± 0.5
2 (4×4)	61.7 ± 0.6	64.7 ± 0.7	79.4 ± 0.3	**81.1** ± 0.3
3 (8×8)	63.3 ± 0.8	**66.8** ± 0.6	77.2 ± 0.4	80.7 ± 0.3

Fei-Fei & Perona: 65.2%

Slide credit: L. Lazebnik

Scene category confusions



Difficult indoor images

kitchen

living room

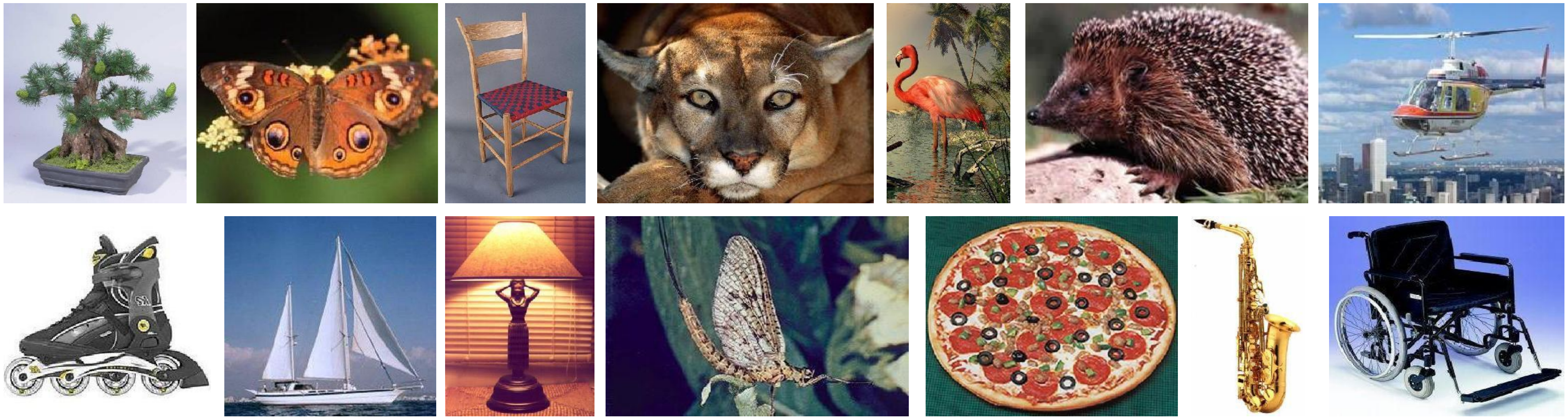
bedroom

Slide credit: L. Lazebnik

Caltech101 dataset

Fei-Fei et al. (2004)

http://www.vision.caltech.edu/Image_Datasets/Caltech101/Caltech101.html



Multi-class classification results (30 training images per class)

	Weak features (16)		Strong features (200)	
Level	Single-level	Pyramid	Single-level	Pyramid
0	15.5 ±0.9		41.2 ±1.2	
1	31.4 ±1.2	32.8 ±1.3	55.9 ±0.9	57.0 ±0.8
2	47.2 ±1.1	49.3 ±1.4	63.6 ±0.9	**64.6** ±0.8
3	52.2 ±0.8	**54.0** ±1.1	60.3 ±0.9	64.6 ±0.7

Slide credit: L. Lazebnik

Training vs Testing

- What do we want?
  - High accuracy on training data?
  - No, high accuracy on *unseen/new/test data!*
  - Why is this tricky?
- Training data
  - Features (x) and labels (y) used to learn mapping f
- Test data
  - Features (x) used to make a prediction
  - Labels (y) only used to see how well we've learned f!!!
- Validation data
  - Held-out set of the *training data*
  - Can use both features (x) and labels (y) to tune parameters of the model we're learning

Generalization

Training set (labels known)

Test set (labels unknown)

- How well does a learned model generalize from the data it was trained on to a new test set?

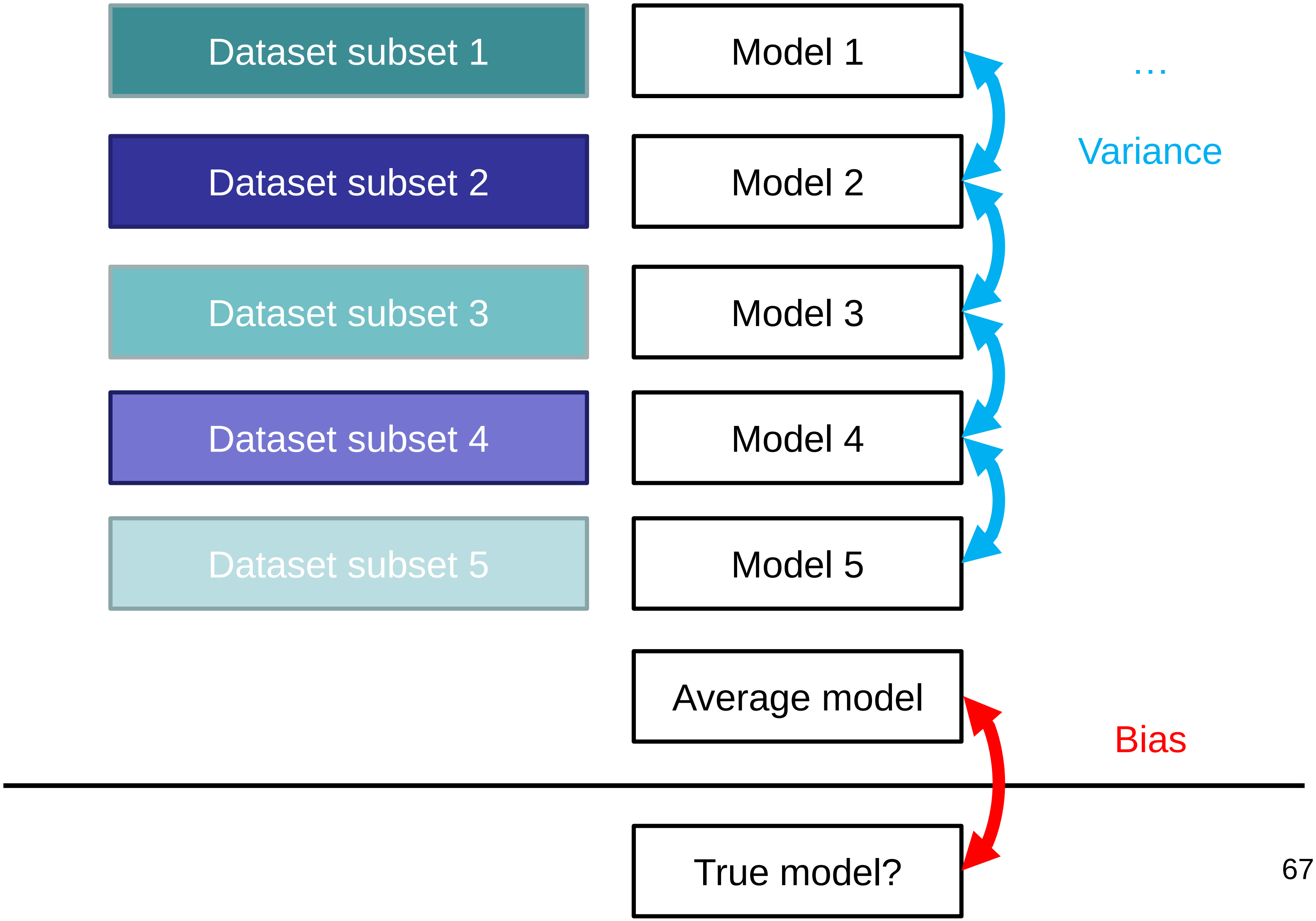
Slide credit: L. Lazebnik

Generalization

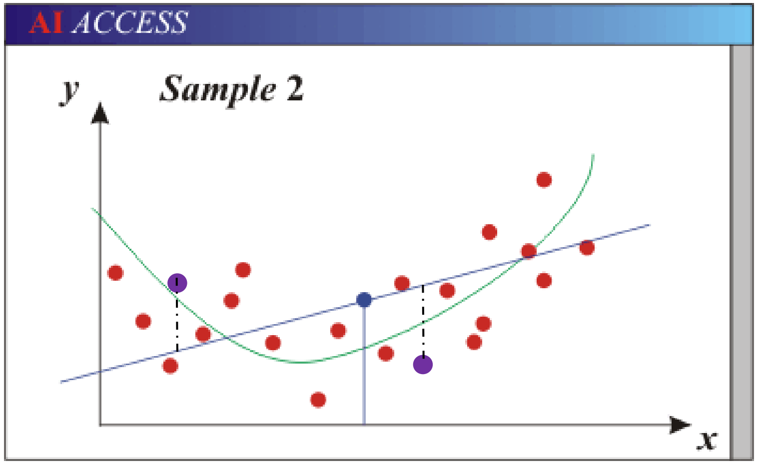
- Components of generalization error
  - **Noise** in our observations: unavoidable
  - **Bias:** due to inaccurate assumptions/simplifications by model
  - **Variance:** models estimated from different training sets differ greatly from each other
- **Underfitting:** model is too “simple” to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting:** model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

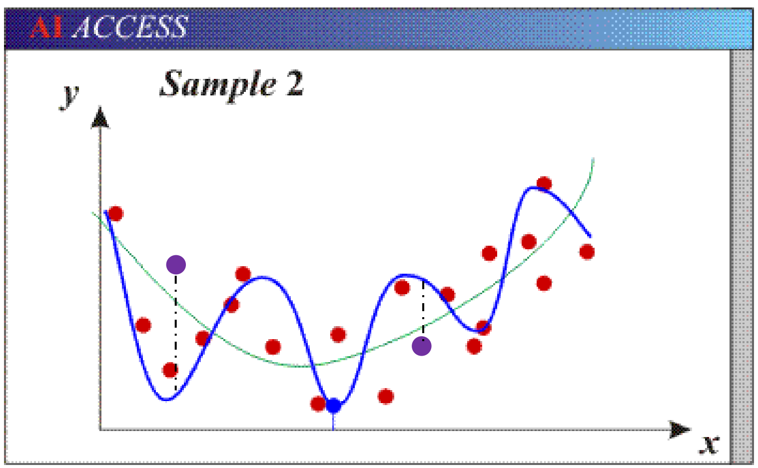
Slide credit: L. Lazebnik

Generalization



Generalization





- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).

- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Purple dots = possible test points

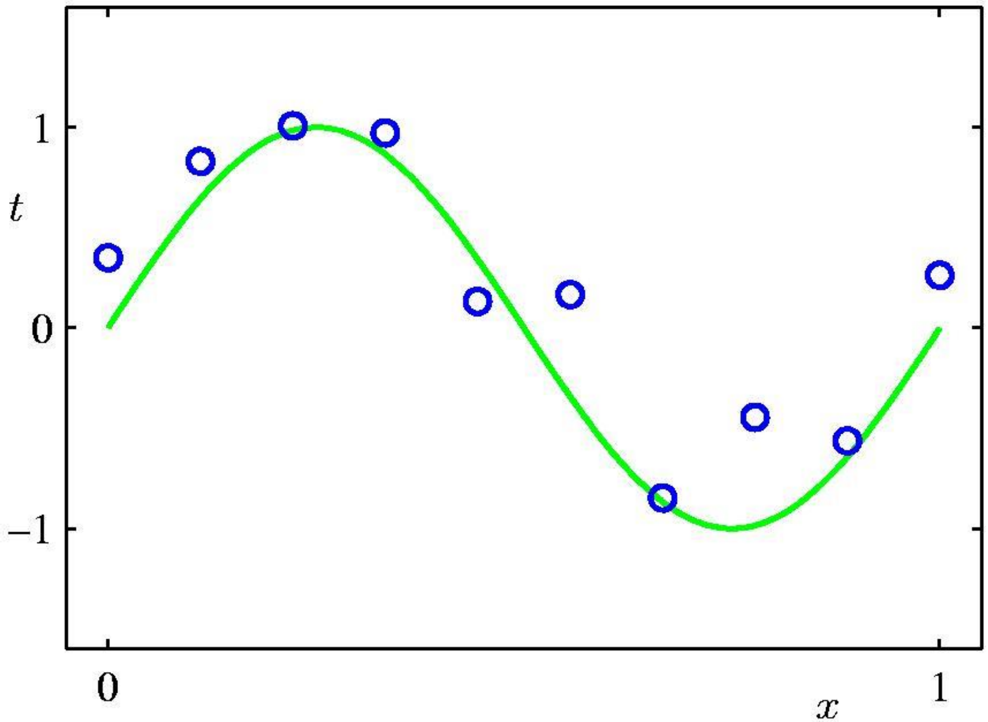
Red dots = training data (all that we see before we ship off our model!)

Green curve = true underlying model

Blue curve = our predicted model/fit

Adapted from D. Hoiem

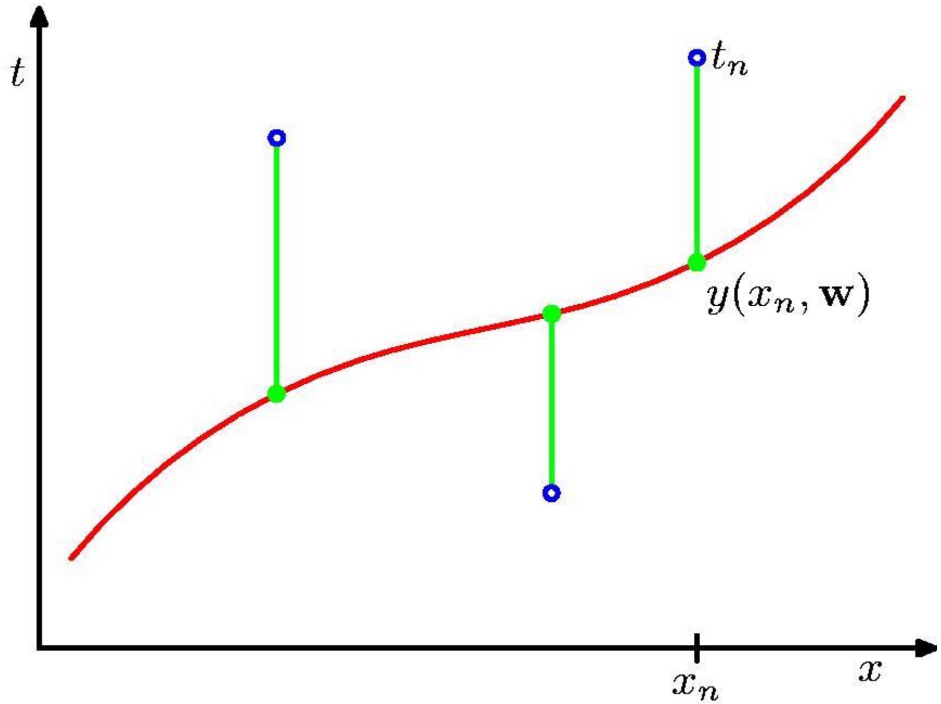
Polynomial Curve Fitting



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Slide credit: Chris Bishop

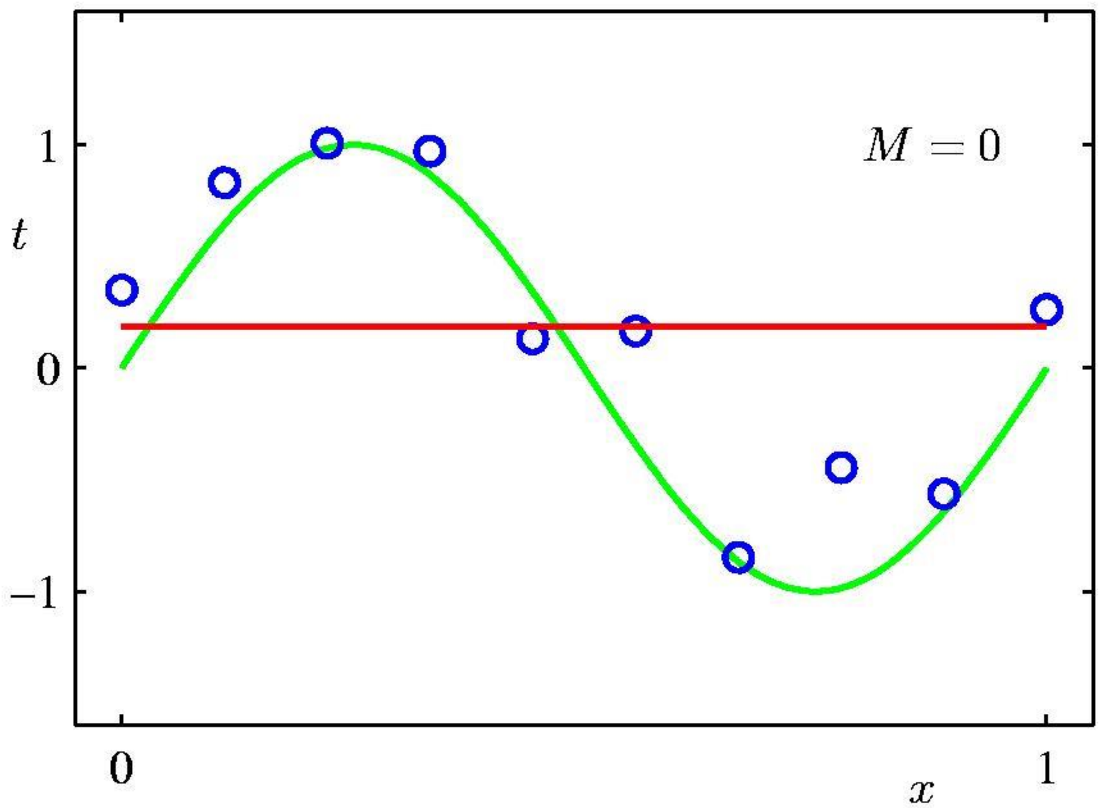
Sum-of-Squares Error Function



$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

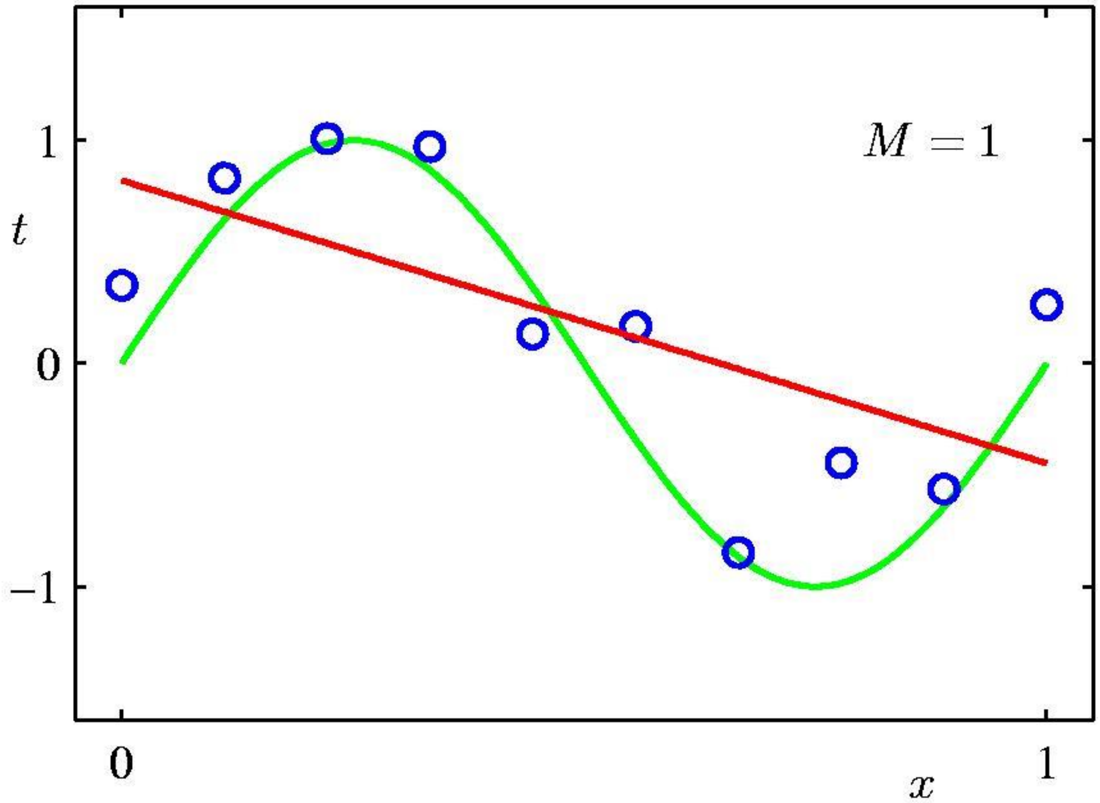
Slide credit: Chris Bishop

0^{th} Order Polynomial



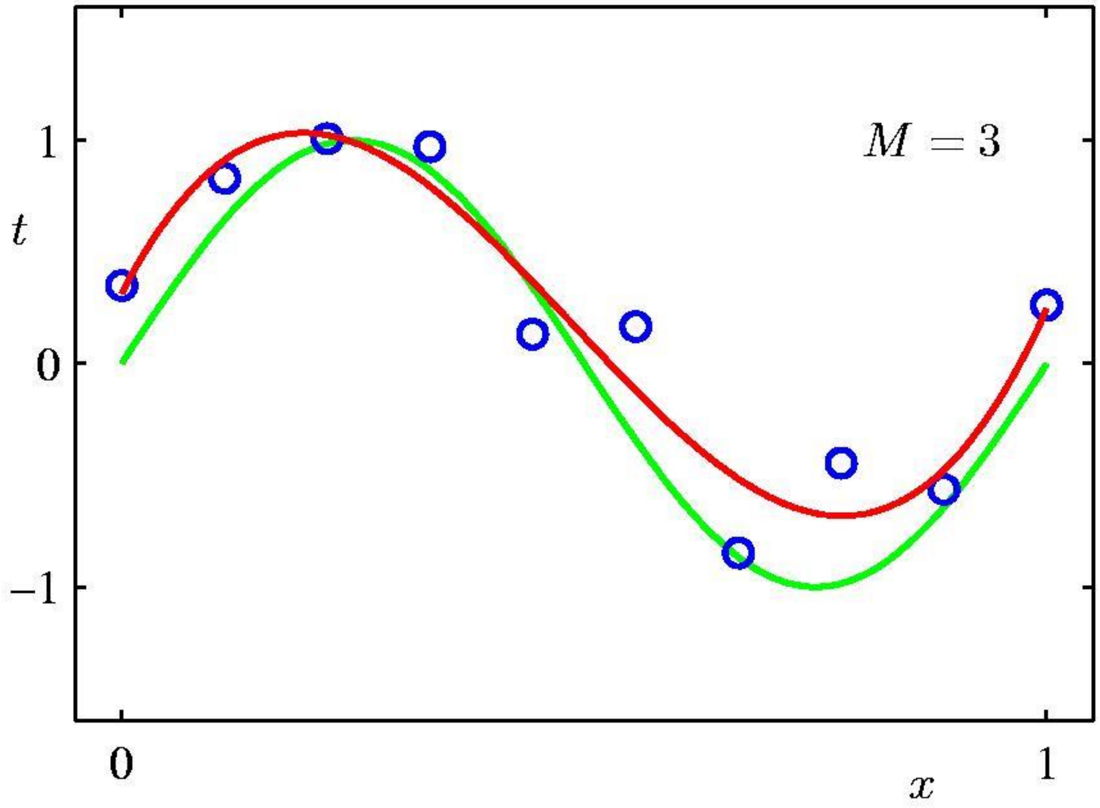
Slide credit: Chris Bishop

1^{st} Order Polynomial



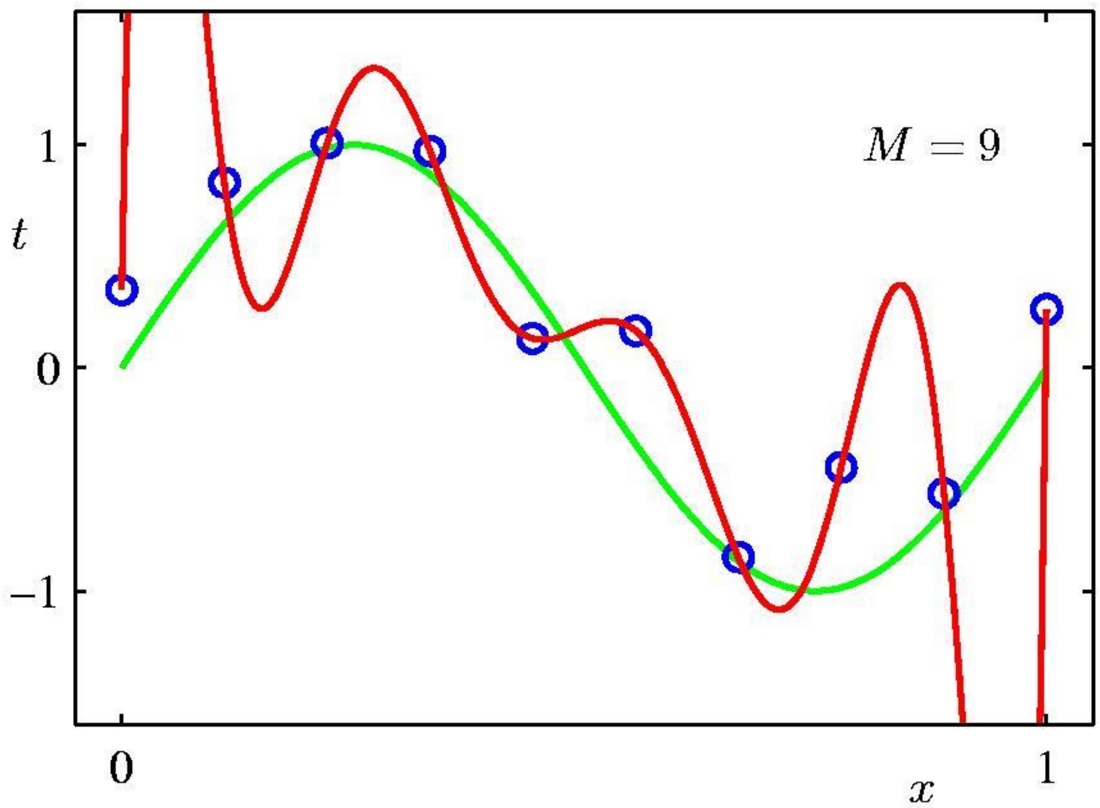
Slide credit: Chris Bishop

3rd Order Polynomial



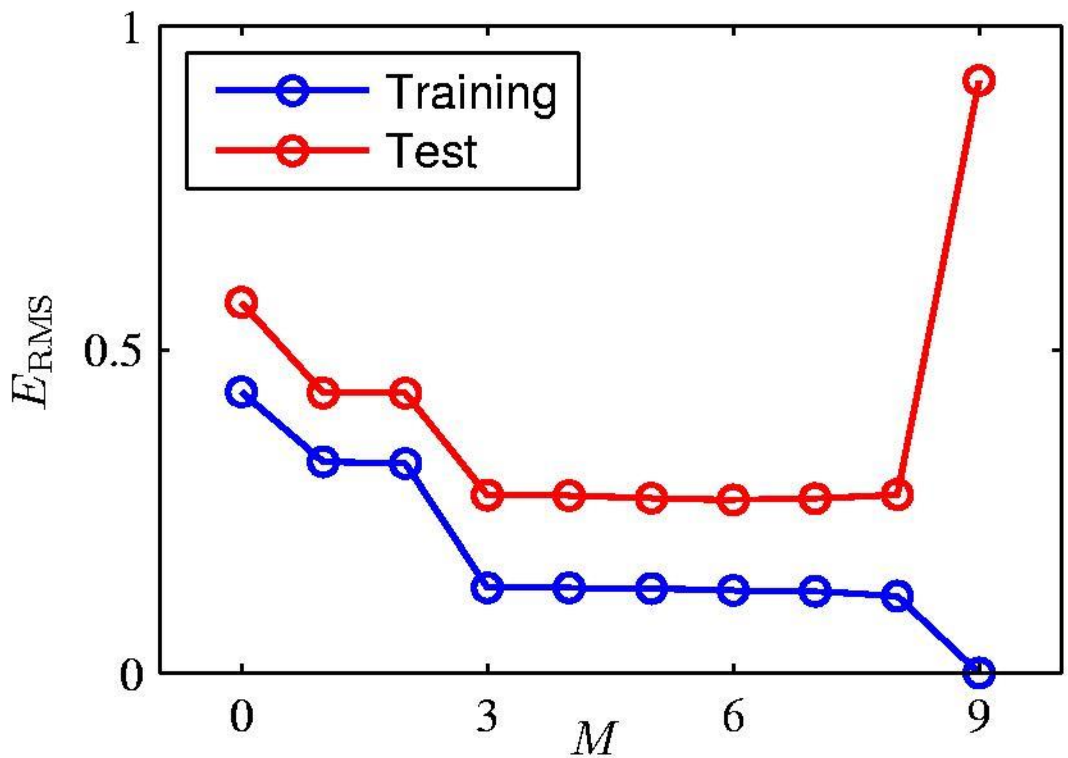
Slide credit: Chris Bishop

9th Order Polynomial



Slide credit: Chris Bishop

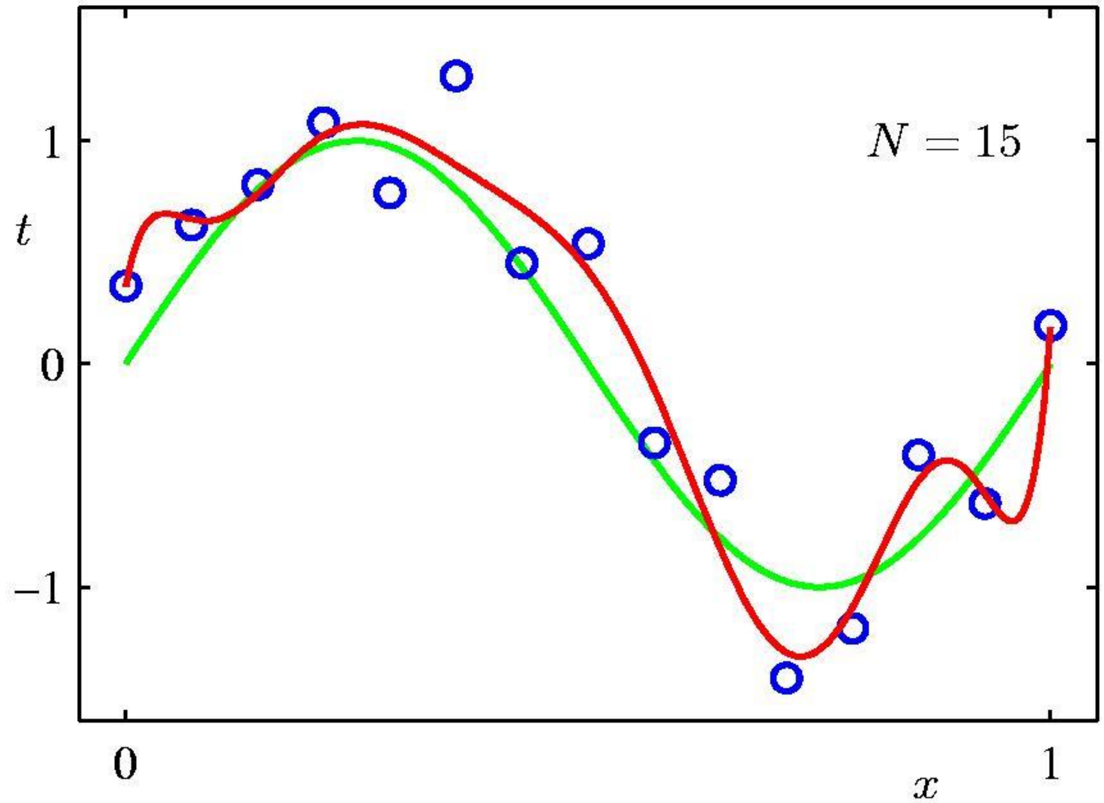
Over-fitting



Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

Data Set Size: $N = 15$

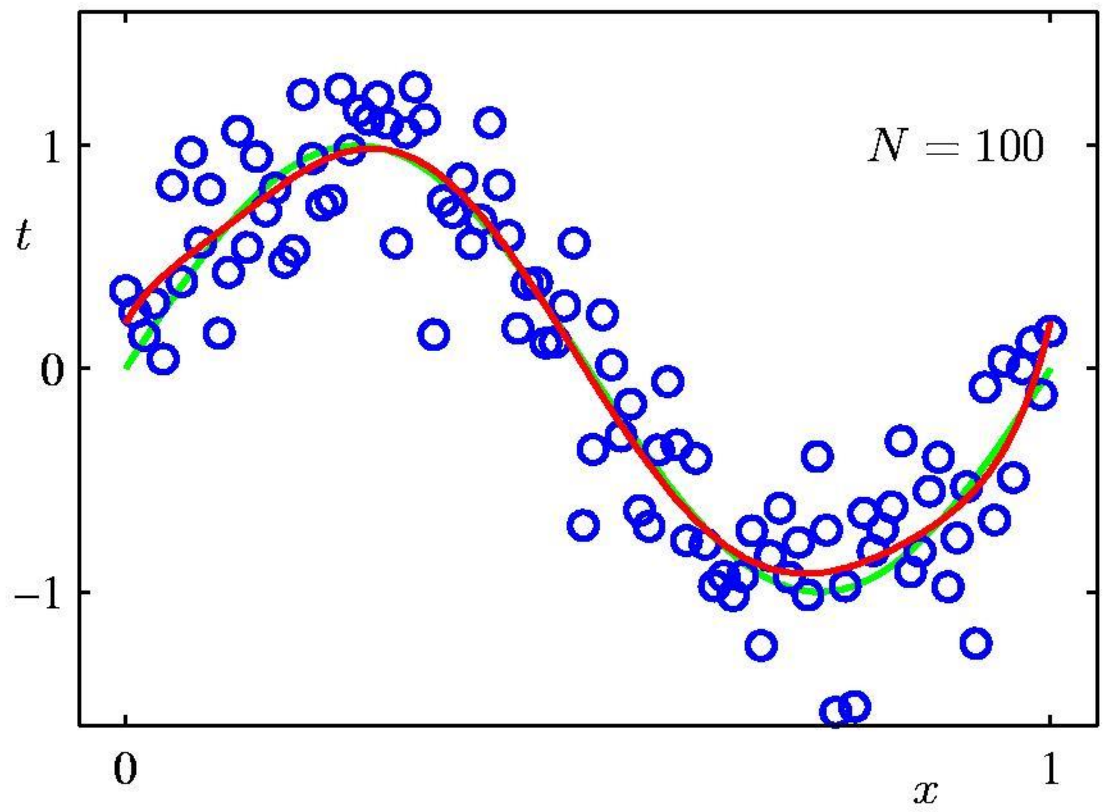
9^{th} Order Polynomial



Slide credit: Chris Bishop

Data Set Size: $N = 100$

9th Order Polynomial



Slide credit: Chris Bishop

Regularization

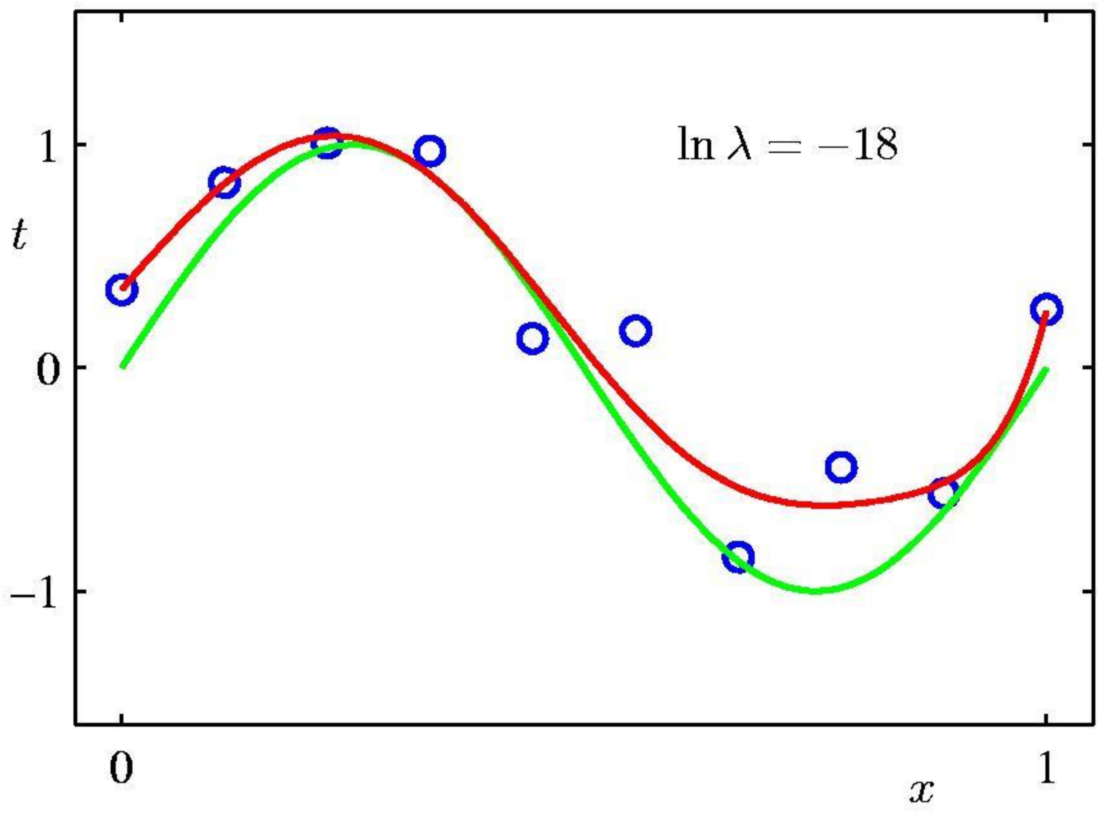
Penalize large coefficient values

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

(Remember: We want to minimize this expression.)

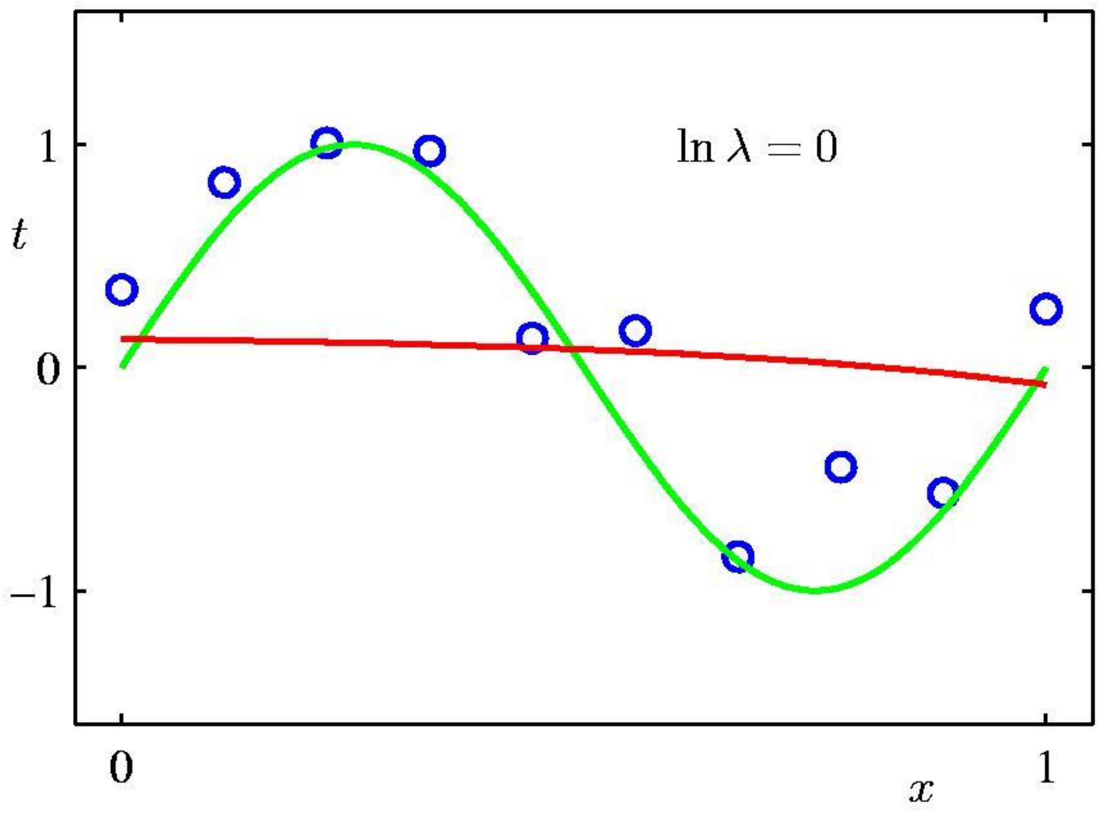
Adapted from Chris Bishop

Regularization: $\ln \lambda = -18$



Slide credit: Chris Bishop

Regularization: $\ln \lambda = 0$



Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

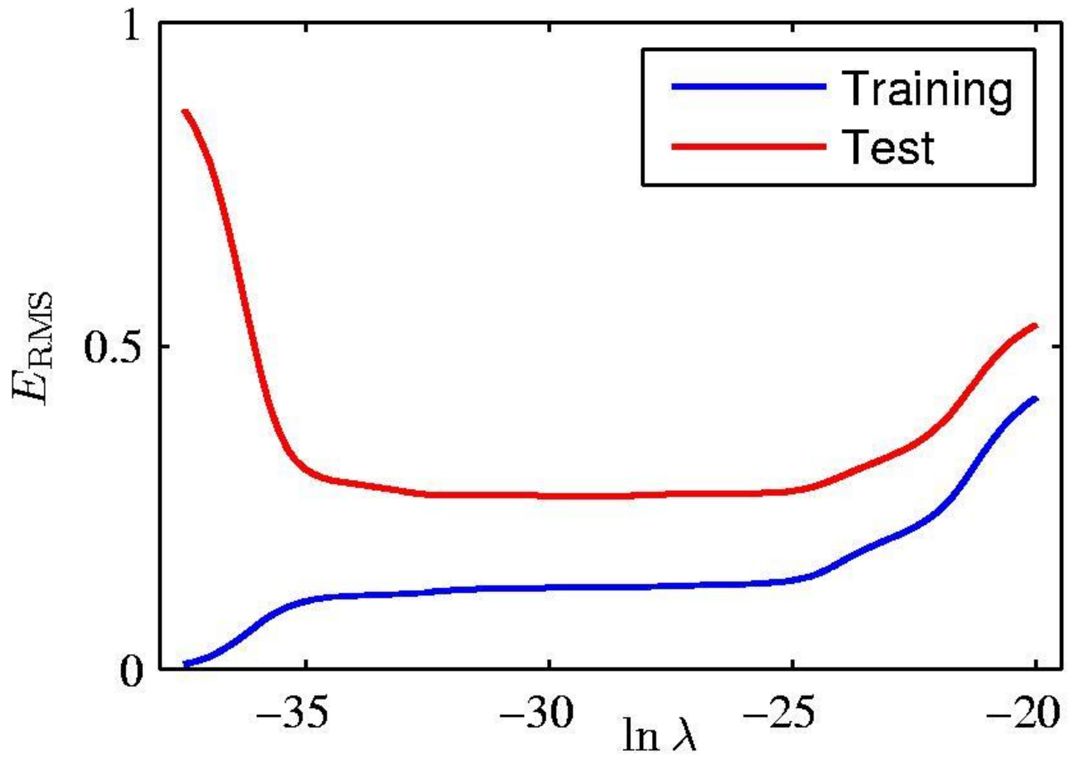
Slide credit: Chris Bishop

Polynomial Coefficients

	No regularization		Huge regularization
	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Adapted from Chris Bishop

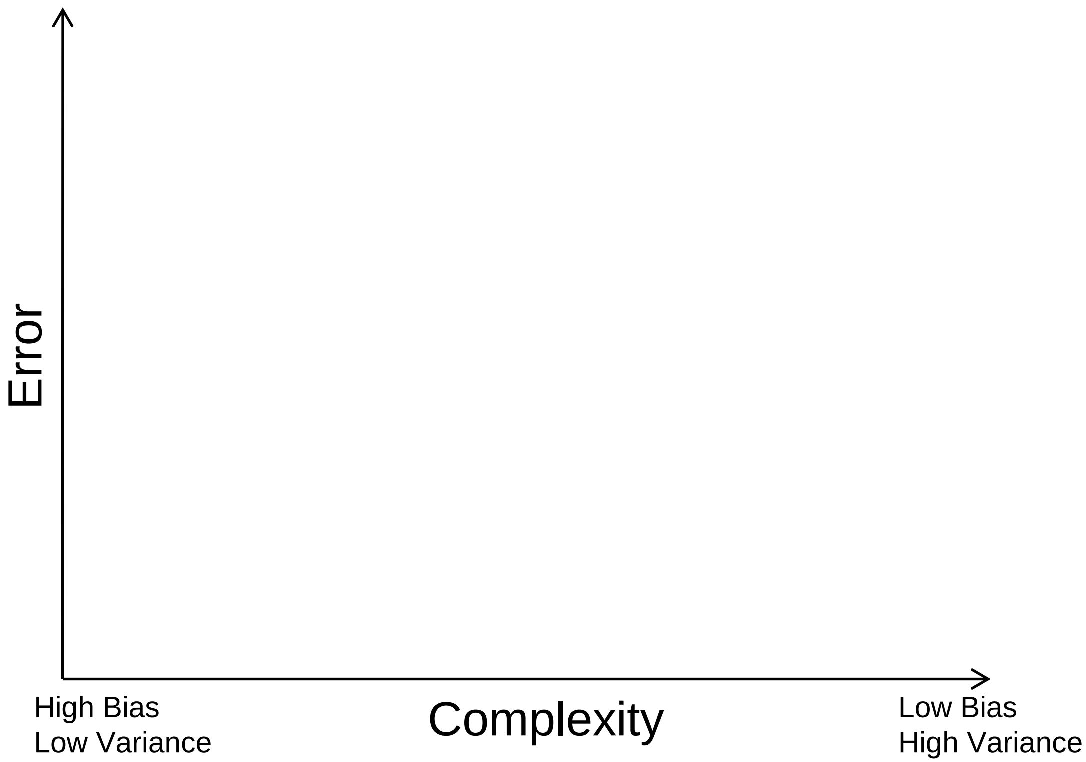
Regularization: E_{RMS} vs. $\ln \lambda$



Slide credit: Chris Bishop

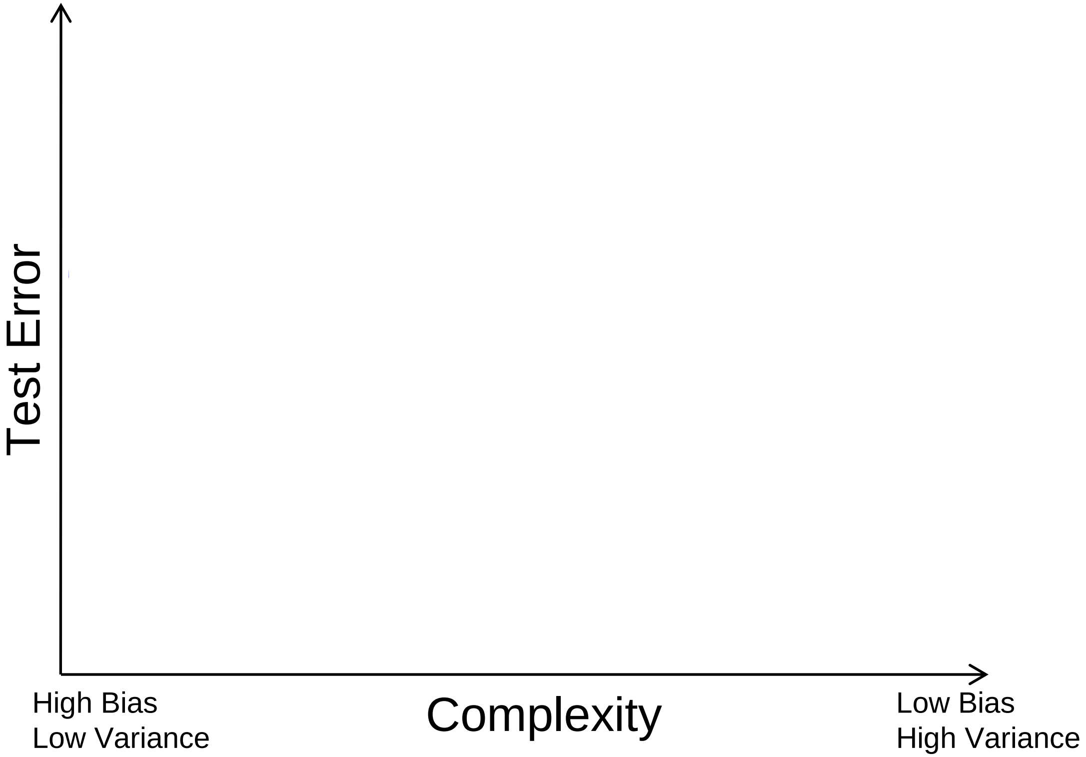
Training vs test error

Underfitting **Overfitting**



Slide credit: D. Hoiem

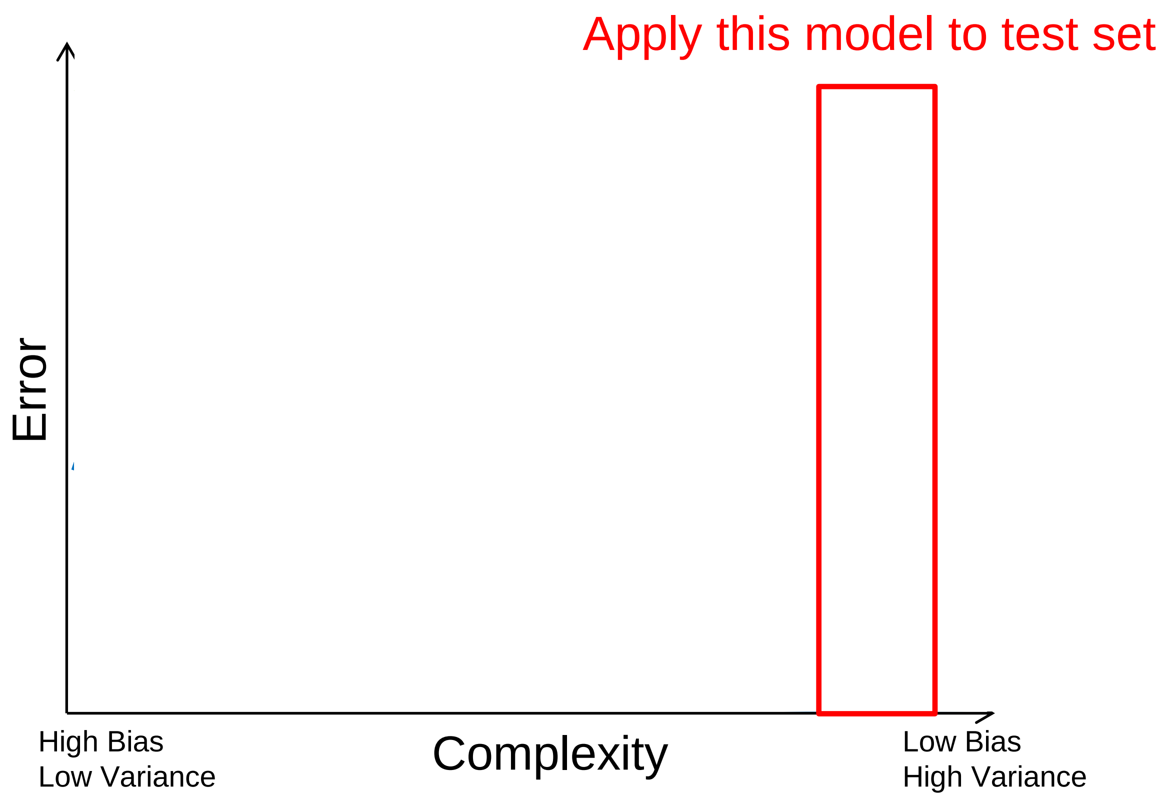
The effect of training set size



Slide credit: D. Hoiem

Choosing the trade-off between bias and variance

- Need validation set (separate from the test set)



Generalization tips

- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
- Use increasingly powerful classifiers with more training data
- As an additional technique for reducing variance, try *regularizing* the parameters (penalize high magnitude weights)

Slide credit: D. Hoiem