Lecture 4: Image classification

Image classification, k-NN, linear classifiers, loss functions, regularization

With slides from A. Karpathy, J. Johnson, F. Fleuret, …

Recap

- Introduction to Machine Learning
- Running and analyzing a pre-trained CNN
- Fine-tuning a CNN on a new task
- Installing and using pytorch (one of the myriad of deep learning libraires out there.

Recap

Recap

Today

- Image classification task
- How to address image classification
- Loss functions
- Regularization

The problem with Computer Vision

This is what you see

The true output of a digital camera

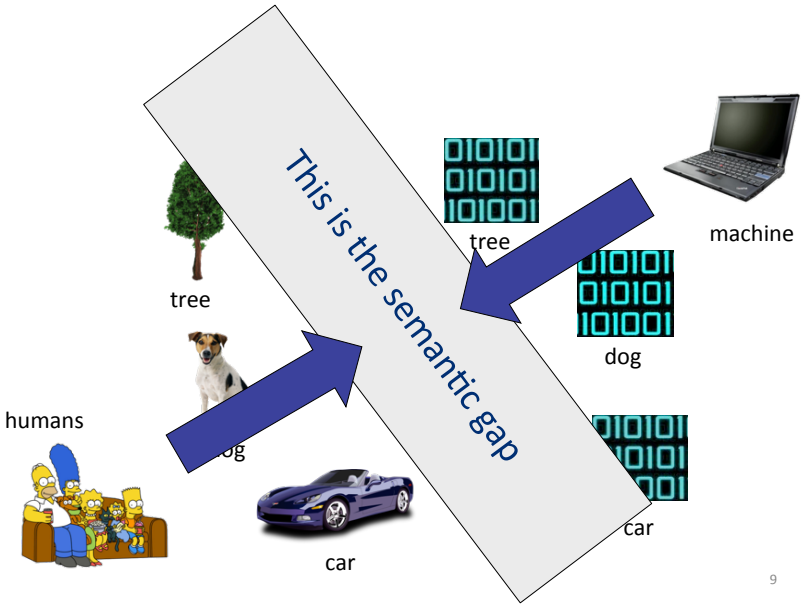
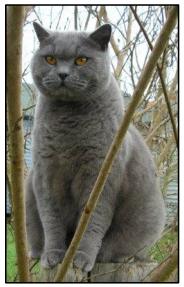


Image classification

- Core task in Computer Vision
- Easy to extrapolate / build other tasks



(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

Image classification

Challenges: robustness

Illumination | Object pose | Clutter

Occlusions | Intra-class appearance | Viewpoint

Image classification

Challenges: intra-class variation

Image classification

Challenges: complexity

- Millions of pixels in an image
- 30k human recognizable categories
- 30+ degrees of freedom in the pose of articulated objects (humans)
- Billions of images online

Image classification

Challenges: learning with minimal supervision

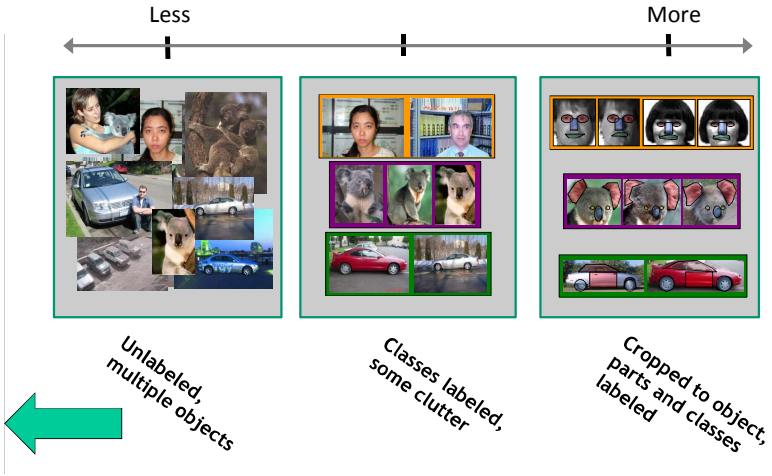


Image classification

- Given a set collection of annotated images, what's the simplest classification we can propose?

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

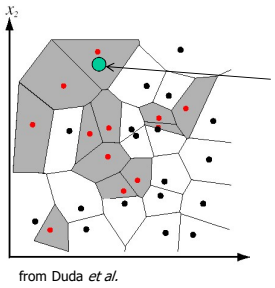
Nearest Neigbor classification

- Given a set collection of annotated images, what's the simplest classification we can propose?

Nearest Neigbor classification

- Assign label of nearest training data point to each test data point

Black = negative
Red = positive



Novel test example

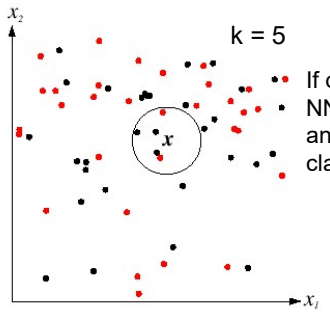
Closest to a positive example from the training set, so classify it as positive.

Voronoi partitioning of feature space for 2-category 2D data

K-Nearest Neighbor classification

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

Black = negative
Red = positive



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

K-Nearest Neighbor classification

- Memorize all labels

```
def train(images, labels):
  # Machine learning!
  return model
```

- Predict the label of the most similar training image

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```


K-Nearest Neighbor classification

Examples

- Model of choice when huge amounts of data are available

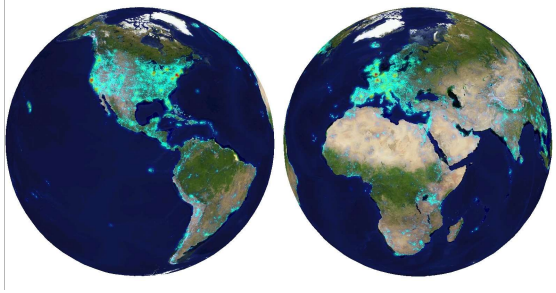
80M Tiny images [Torralba 2008]

K-Nearest Neighbor classification

Examples

- Where in the world?

6M geotagged photos by 109k photographers



im2gps [Hays 2008]

K-Nearest Neighbor classification

Examples

- Typical features:
  - Gist scene descriptor: 80d
  - Color histogram: L*a*b* 4x14x14 histograms
  - Texton histogram: 512d
  - Line features: histogram of straight line stats

K-Nearest Neighbor classification

Examples

- Distance metric to compare images

L1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

test image					training image					pixel-wise absolute value differences				
56	32	10	18	-	10	20	24	17	=	46	12	14	1	add → 456
90	23	128	133		8	10	89	100		82	13	39	33	
24	26	178	200		12	16	178	170		12	10	0	30	
2	0	255	220		4	32	233	112		2	32	22	108	

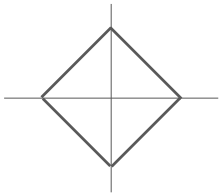
K-Nearest Neighbor classification

Examples

- Distance metric to compare images

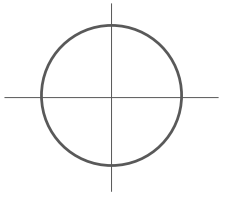
L1 (Manhattan) distance

$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$



L2 (Euclidean) distance

$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$



K-Nearest Neighbor classification

Examples

- Where in the world?

im2gps [Hays 2008]

K-Nearest Neighbor classification

Examples

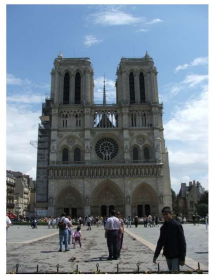
- Where in the world?

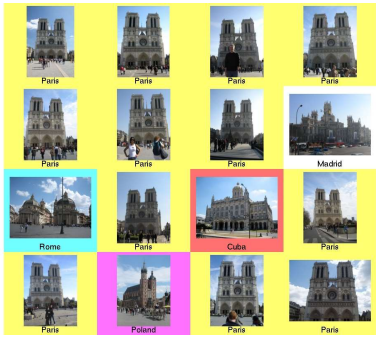
im2gps [Hays 2008]

K-Nearest Neighbor classification

Examples

- Where in the world?





im2gps [Hays 2008]

K-Nearest Neighbor classification

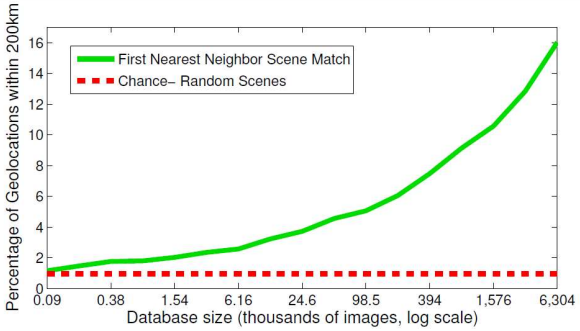
Examples

- Where in the world?

im2gps [Hays 2008]

K-Nearest Neighbor classification

- The importance of data



im2gps [Hays 2008]

K-Nearest Neighbor classification

- Question:
  - With N examples how fast are training and prediction?

K-Nearest Neighbor classification

- Question:
  - With N examples how fast are training and prediction?

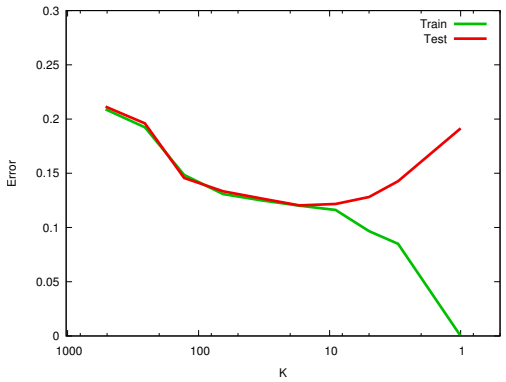
Train O(1); Predict O(N)

We want classifiers that are fast at prediction; it's ok if they're slow for training.

K-Nearest Neighbor classification

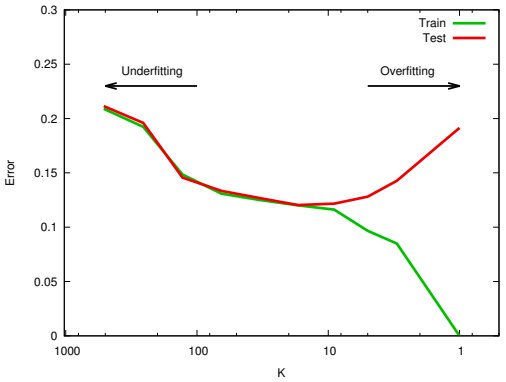
- Question:
  - With N examples how fast are training and prediction?
  - What are you hyper-parameters for k-NN?

K-Nearest Neighbor classification



Slide credit: F. Fleuret

K-Nearest Neighbor classification



Slide credit: F. Fleuret

K-Nearest Neighbor classification

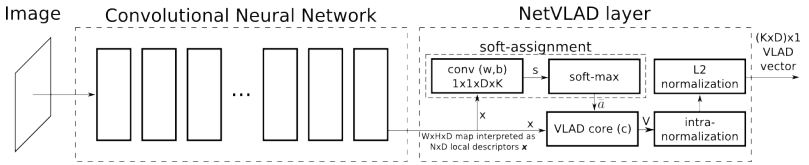
Pros & Cons

- Pros:
  - Simple to implement
  - Flexible to feature/distance choices
  - Naturally handles multi-class cases
  - Can do well in practice with enough representative data
- Cons:
  - Large search problem to find nearest neighbors
  - Storage of data
  - Must know we have a meaningful distance function
  - Curse of dimensionality

K-Nearest Neighbor classification

Pros & Cons

(a) Mobile phone query (b) Retrieved image of same place



Linear classification

Linear classification

- CIFAR-10 dataset

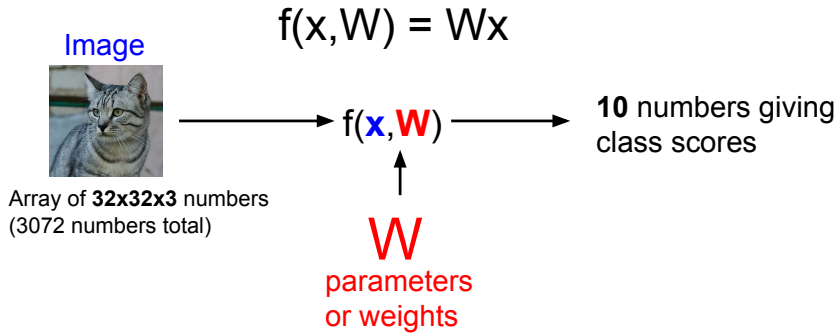
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

50,000 training images
each image is **32x32x3**

10,000 test images.

Linear classification

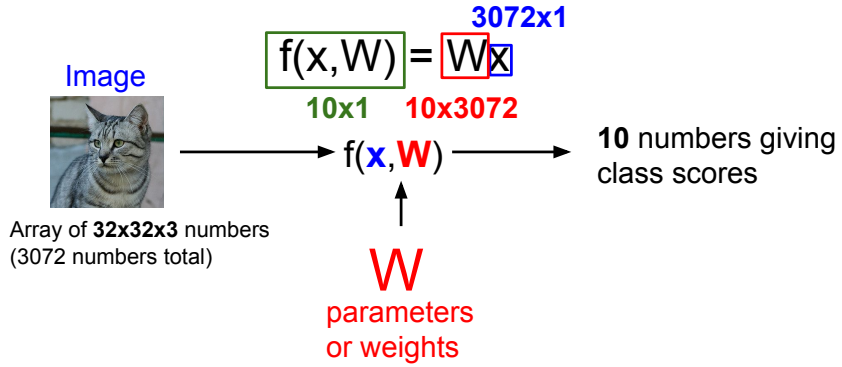
Parametric approach



Slide credit: J. Johnson

Linear classification

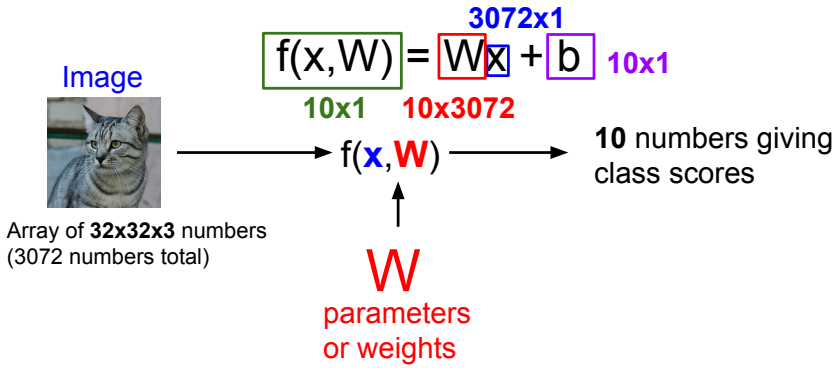
Parametric approach



Slide credit: J. Johnson

Linear classification

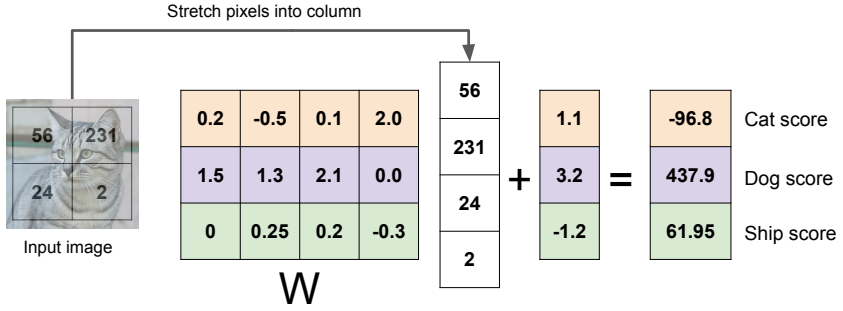
Parametric approach



Linear classification

Parametric approach

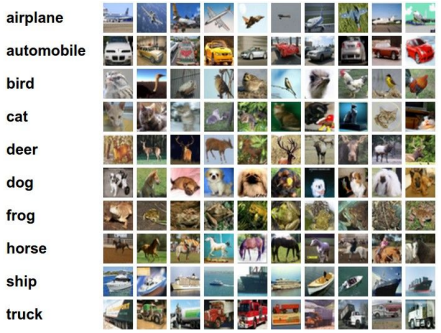
- Example with an image with 4 pixels and 3 classes (cat/dog/ship)
- **X:** 4-dimensional descriptor/feature



Linear classification

Interpreting a linear classifier

- What does actually the linear classifier do?

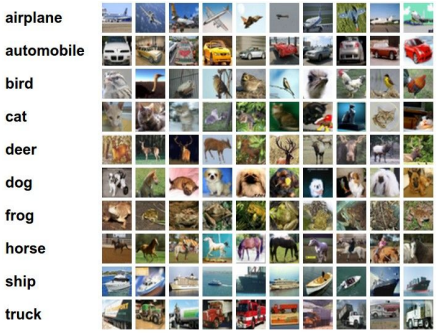


$$f(x_i, W, b) = Wx_i + b$$

Linear classification

Interpreting a linear classifier

- What does actually the linear classifier do?



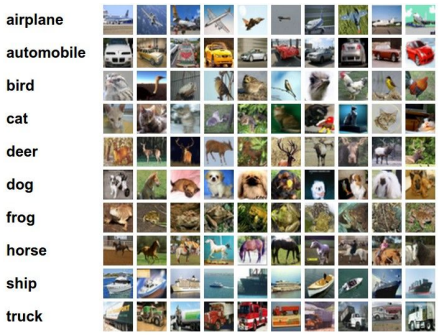
$$f(x_i, W, b) = Wx_i + b$$

- Practically learning a template representation for its corresponding class

Linear classification

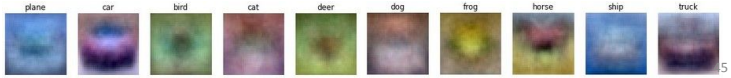
Interpreting a linear classifier

- What does actually the linear classifier do?



$$f(x_i, W, b) = Wx_i + b$$

- Example trained weights of a linear classifier trained on CIFAR-10

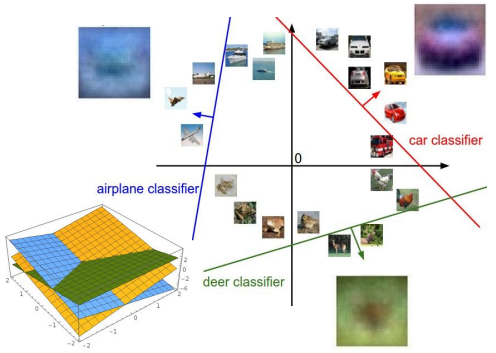


Linear classification

Interpreting a linear classifier

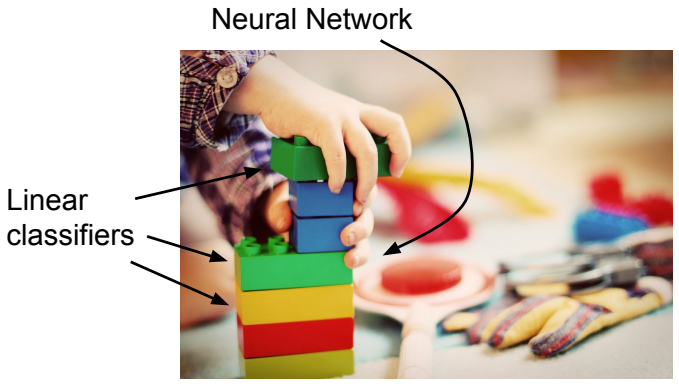
- What does actually the linear classifier do?

$$f(x_i, W, b) = Wx_i + b$$



Linear classification

Interpreting a linear classifier



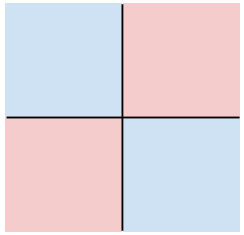
Linear classification

Interpreting a linear classifier

- Hard cases for a linear classifier?

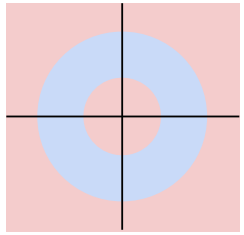
Class 1:
number of pixels > 0 odd

Class 2:
number of pixels > 0 even



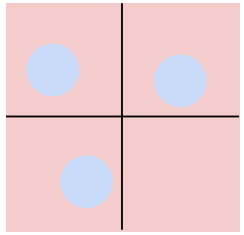
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else



Classification

- How to measure how "*happy*" are we with the weights the classifier has learned?
- Define a **loss function** that quantifies our unhappiness with the scores across the training data
- Let's see how to do it on a few popular classifiers: **multi-class SVM**, **softmax classifier**

Classification

- Suppose 3 training examples and their scores $f(x_i, W, b) = Wx_i + b$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Slide credit: J. Johnson

Classification

- Suppose 3 training examples and their scores $f(x_i, W, b) = Wx_i + b$
- A loss function shows how good our current classifier is
- Given a dataset of examples $\{(x_i, y_i)\}_{i=1}^{N}$
- Loss over the dataset is a sum of loss over examples

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

x_i image
y_i label (integer)

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Slide credit: J. Johnson

Multi-class SVM loss

- Suppose 3 training examples and their scores $f(x_i, W, b) = Wx_i + b$
- Given a dataset of examples $\{(x_i, y_i)\}_{i=1}^N$
- The SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$
$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

$s = f(x_i, W)$

x_i image
y_i label (integer)

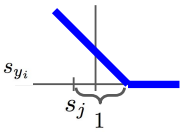
Multi-class SVM loss

- Suppose 3 training examples and their scores $f(x_i, W, b) = Wx_i + b$
- Given a dataset of examples $\{(x_i, y_i)\}_{i=1}^N$
- The SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$
$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

"Hinge loss"



Multi-class SVM loss

- Suppose 3 training examples and their scores $f(x_i, W, b) = Wx_i + b$
- Given a dataset of examples $\{(x_i, y_i)\}_{i=1}^N$
- The SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$
$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

alternative formulation

$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

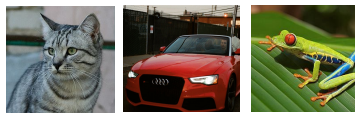
Multi-class SVM loss

- Suppose 3 training examples and their scores $f(x_i, W, b) = Wx_i + b$
- Given a dataset of examples $\{(x_i, y_i)\}_{i=1}^N$
- The SVM loss has the form:

$s = f(x_i, W)$

$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

= max(0, 5.1 - 3.2 + 1)
+max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9		

Multi-class SVM loss

- Suppose 3 training examples and their scores $f(x_i, W, b) = Wx_i + b$
- Given a dataset of examples $\{(x_i, y_i)\}_{i=1}^N$
- The SVM loss has the form:

$$s = f(x_i, W)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	

Slide credit: J. Johnson

Multi-class SVM loss

- Suppose 3 training examples and their scores $f(x_i, W, b) = Wx_i + b$
- Given a dataset of examples $\{(x_i, y_i)\}_{i=1}^N$
- The SVM loss has the form:

$$s = f(x_i, W)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 6.3) + max(0, 6.6)
= 6.3 + 6.6
= 12.9

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Slide credit: J. Johnson

Multi-class SVM loss

- Suppose 3 training examples and their scores $f(x_i, W, b) = Wx_i + b$
- Given a dataset of examples $\{(x_i, y_i)\}_{i=1}^N$
- The SVM loss has the form:

$$s = f(x_i, W)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Loss over full dataset

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

L = (2.9 + 0 + 12.9)/3
= **5.27**

Slide credit: J. Johnson

Multi-class SVM loss

Questions

- What happens to the loss if *car* scores change a bit?

$$s = f(x_i, W)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Slide credit: J. Johnson

Multi-class SVM loss

Questions

- What is the possible min/max loss?

$s = f(x_i, W)$

$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Slide credit: J. Johnson

Multi-class SVM loss

Questions

- At initialization W is so small all s ~= 0. What is the loss?

$s = f(x_i, W)$

$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Slide credit: J. Johnson

Multi-class SVM loss

Questions

- What if we used? $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2$

$s = f(x_i, W)$

$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Slide credit: J. Johnson

Multi-class SVM loss

Example code

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

```
def L_i_vectorized(x, y, W):
  scores = W.dot(x)
  margins = np.maximum(0, scores - scores[y] + 1)
  margins[y] = 0
  loss_i = np.sum(margins)
  return loss_i
```

Slide credit: J. Johnson

Softmax Classifier (Multinomial Logistic Regression)

scores = unnormalized log probabilities of the classes.

$$s = f(x_i; W)$$

cat	**3.2**
car	5.1
frog	-1.7

Slide credit: A. Karpathy

Softmax Classifier (Multinomial Logistic Regression)

scores = unnormalized log probabilities of the classes.

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \quad \text{where} \quad s = f(x_i; W)$$

cat	**3.2**
car	5.1
frog	-1.7

Slide credit: A. Karpathy

Softmax Classifier (Multinomial Logistic Regression)

scores = unnormalized log probabilities of the classes.

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \quad \text{where} \quad s = f(x_i; W)$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

$$L_i = -\log P(Y = y_i|X = x_i)$$

cat	**3.2**
car	5.1
frog	-1.7

Slide credit: A. Karpathy

Softmax Classifier (Multinomial Logistic Regression)

scores = unnormalized log probabilities of the classes.

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \quad \text{where} \quad s = f(x_i; W)$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

$$L_i = -\log P(Y = y_i|X = x_i)$$

in summary: $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})$

cat	**3.2**
car	5.1
frog	-1.7

Slide credit: A. Karpathy

Softmax Classifier (Multinomial Logistic Regression)

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

unnormalized probabilities

	unnormalized log probabilities		exp		normalize		probabilities	
cat	**3.2**	→	**24.5**	→	**0.13**	→	L_i = -log(0.13) = **0.89**
car	5.1		164.0		0.87		
frog	-1.7		0.18		0.00		

Slide credit: A. Karpathy

Softmax Classifier (Multinomial Logistic Regression)

Questions

- What is the min/max possible for L_i?

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

unnormalized probabilities

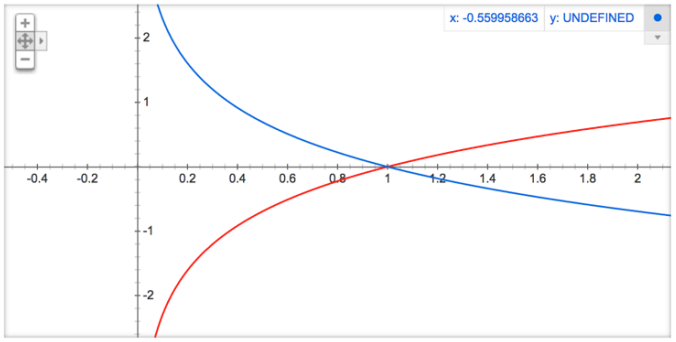
	unnormalized log probabilities		exp		normalize		probabilities	
cat	**3.2**	→	**24.5**	→	**0.13**	→	L_i = -log(0.13) = **0.89**
car	5.1		164.0		0.87		
frog	-1.7		0.18		0.00		

Slide credit: A. Karpathy

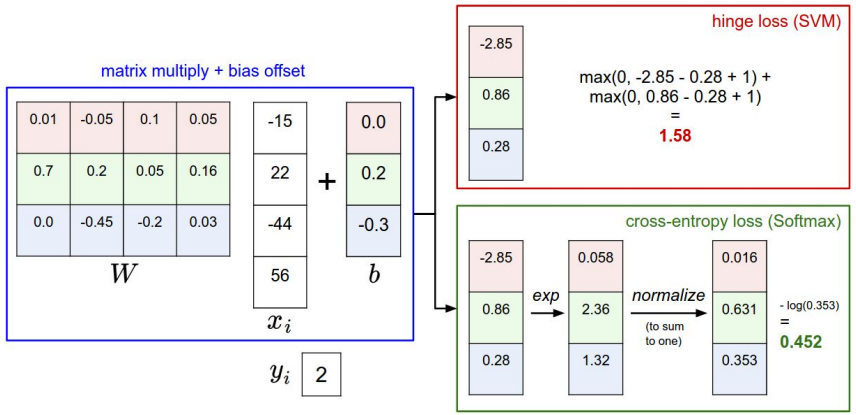
Softmax Classifier (Multinomial Logistic Regression)

- ln(x)

Graph for -ln(x), ln(x)



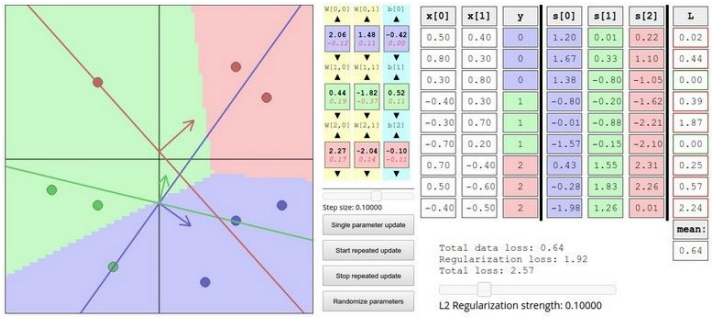
Softmax Classifier (Multinomial Logistic Regression)



Slide credit: A. Karpathy

Softmax Classifier (Multinomial Logistic Regression)

- Seeing the classifier in action: http://vision.stanford.edu/teaching/cs231n/linear-classify-demo/



Softmax vs SVM

- SVM stops optimizing when margin is ok
- Softmax is never reaching zero
- In practice both do fairly well
- Softmax is more popular in the last years

Multi-class SVM loss

Questions

- Suppose that we found a W such that L=0. Is this W unique?

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

Slide credit: J. Johnson

Multi-class SVM loss

Questions

- Suppose that we found a W such that L=0. Is this W unique?
- No! 2W also has L = 0!

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

Slide credit: J. Johnson

Multi-class SVM loss

Questions

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	

Slide credit: J. Johnson

Before:

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

With W twice as large:

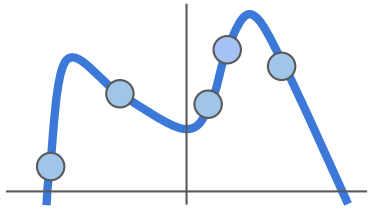
= max(0, 2.6 - 9.8 + 1)
+max(0, 4.0 - 9.8 + 1)
= max(0, -6.2) + max(0, -4.8)
= 0 + 0
= 0

Regularization

- Why regularization?

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

- **Data loss:** model predictions should match training data

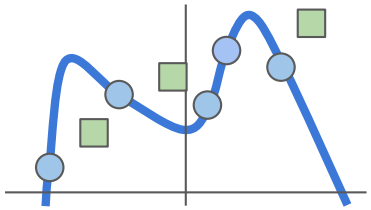


Regularization

- Why regularization?

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

- **Data loss:** model predictions should match training data

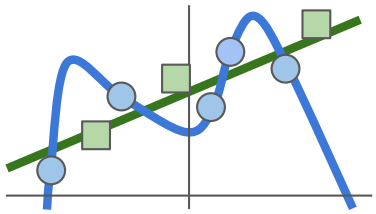


Regularization

- Why regularization?

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

- **Data loss:** model predictions should match training data

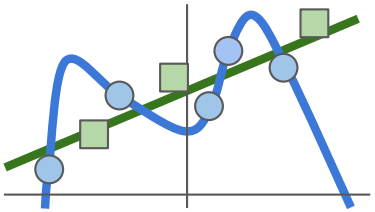


Regularization

- Why regularization?

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)\lambda$$

- **Data loss:** model predictions should match training data
- **Regularization**: Model should be "simple", to work on test data



Regularization

- Full-loss

Lambda = regularization strength (hyperparameter)

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)\lambda$$

- Weight regularization (popular approaches):
  - L2 regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$
  - L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$
  - Dropout (for deep)
  - Batch normalization, stochastic depth (for deep)

Regularization

- **L2 regularization** (weight decay)

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

$$x = [1, 1, 1, 1]$$

$$w_1 = [1, 0, 0, 0]$$

$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

$$w_1^T x = w_2^T x = 1$$

Regularization

- Consider a regression problem
- This is a standard quadratic problem, for which we have efficient algorithms

$$f(x;\alpha) = \sum_{d=0}^{D} \alpha_d x^d.$$

Given $(x_n, y_n) \in \mathbb{R}^2, n = 1, \ldots, N$, minimize

$$\begin{aligned}\mathcal{L}(\alpha) &= \sum_n (f(x_n;\alpha) - y_n)^2 \\ &= \sum_n \left(\sum_{d=0}^{D} \alpha_d x_n^d - y_n\right)^2 \\ &= \left\| \begin{pmatrix} x_1^0 & \ldots & x_1^D \\ \vdots & & \vdots \\ x_N^0 & \ldots & x_N^D \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_D \end{pmatrix} - \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \right\|^2.\end{aligned}$$

Slide credit: F. Fleuret

Regularization

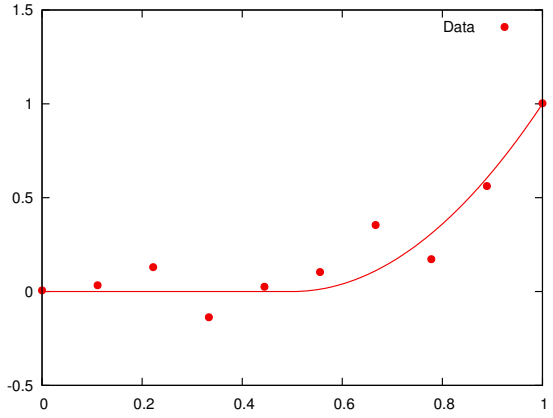
```
X = Tensor(N, D+1)
Y = Tensor(N, 1)

for n in range(N):
    for d in range(D+1):
        X[n, d] = x[n]**d
    Y[n, 0] = y[n]

# LAPACK's GEneralized Least-Square
alpha, _ = torch.gels(Y, X)
```

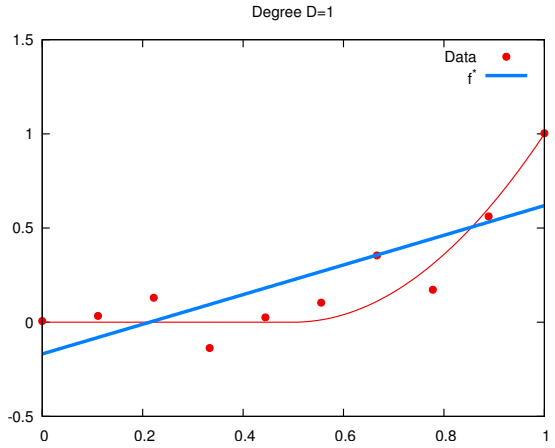
Slide credit: F. Fleuret

Regularization



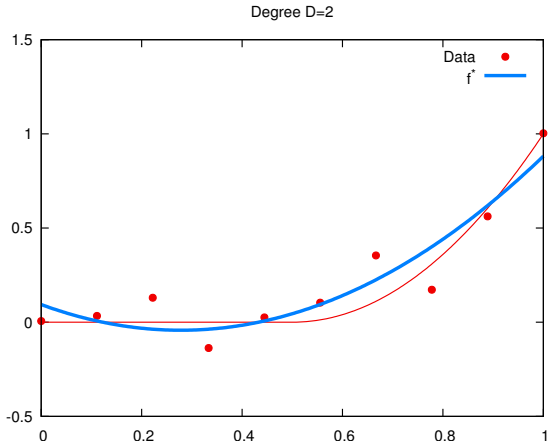
Slide credit: F. Fleuret

Regularization



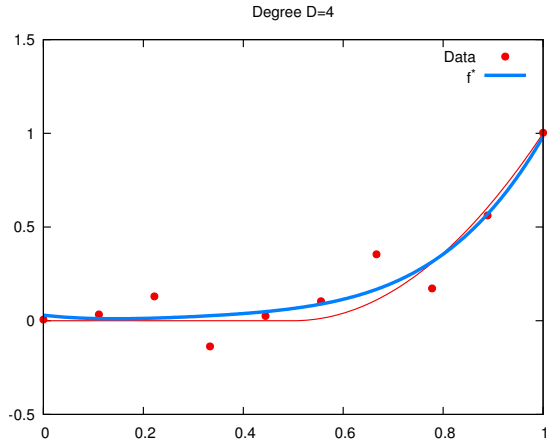
Slide credit: F. Fleuret

Regularization



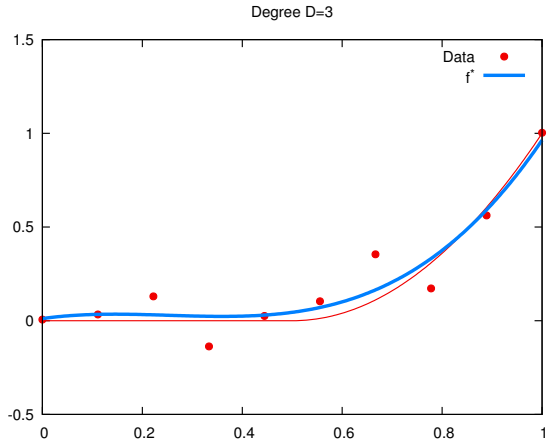
Slide credit: F. Fleuret

Regularization



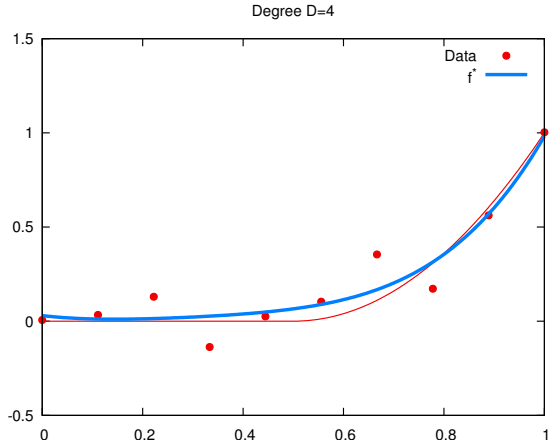
Slide credit: F. Fleuret

Regularization



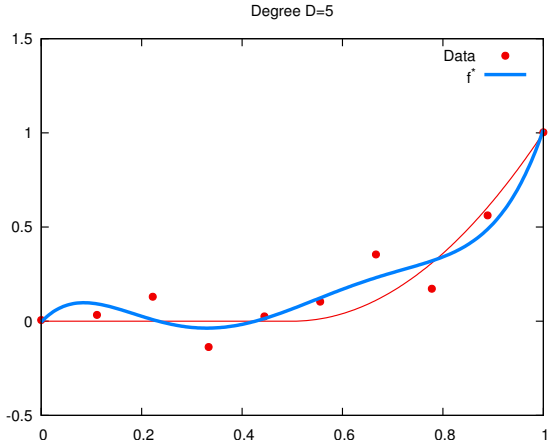
Slide credit: F. Fleuret

Regularization



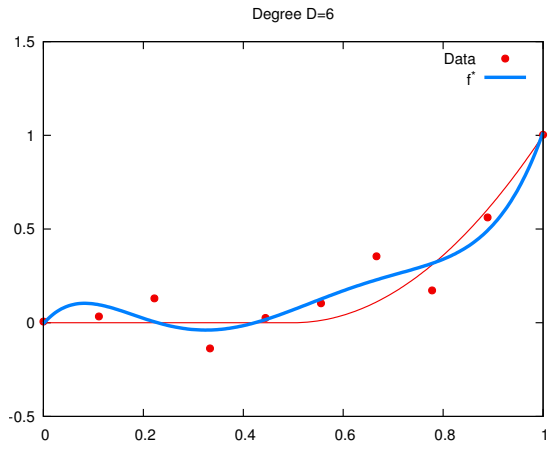
Slide credit: F. Fleuret

Regularization



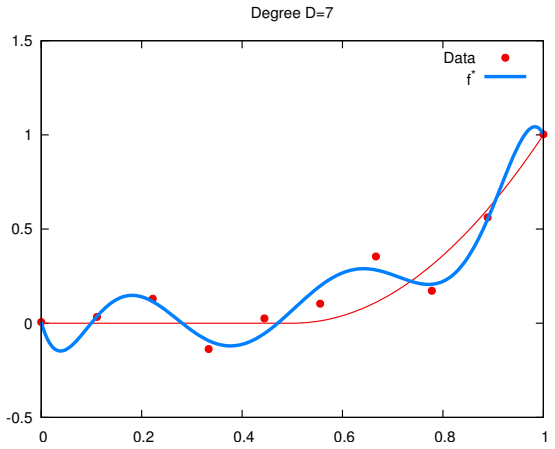
Slide credit: F. Fleuret

Regularization



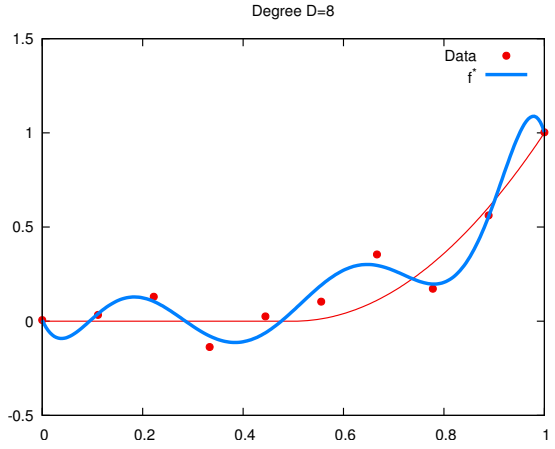
Slide credit: F. Fleuret

Regularization



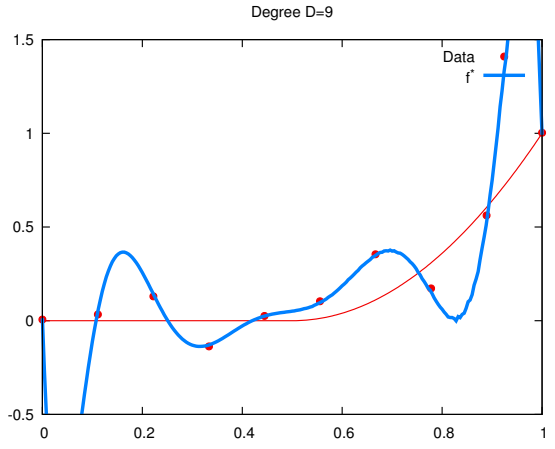
Slide credit: F. Fleuret

Regularization



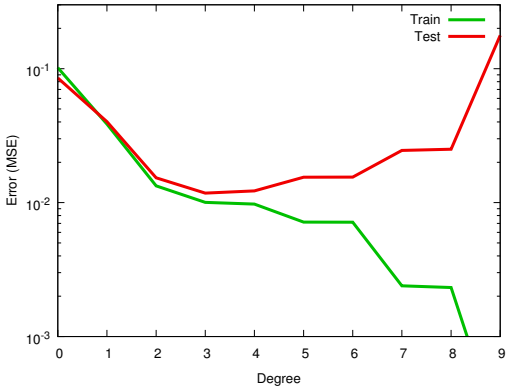
Slide credit: F. Fleuret

Regularization



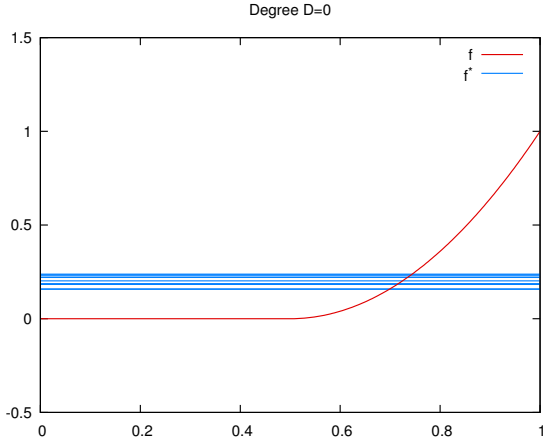
Slide credit: F. Fleuret

Regularization



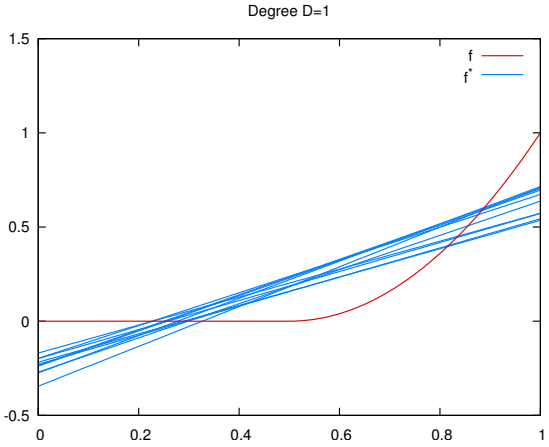
Slide credit: F. Fleuret

Regularization



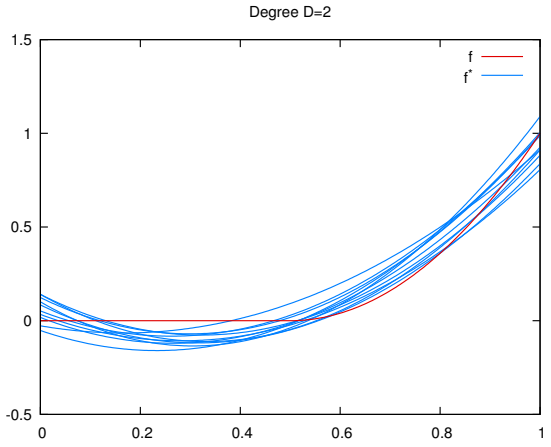
Slide credit: F. Fleuret

Regularization



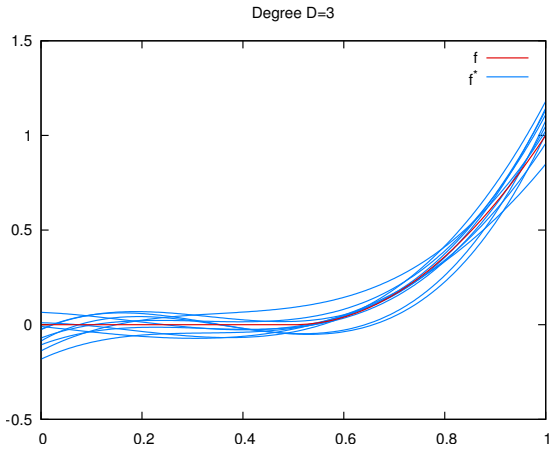
Slide credit: F. Fleuret

Regularization



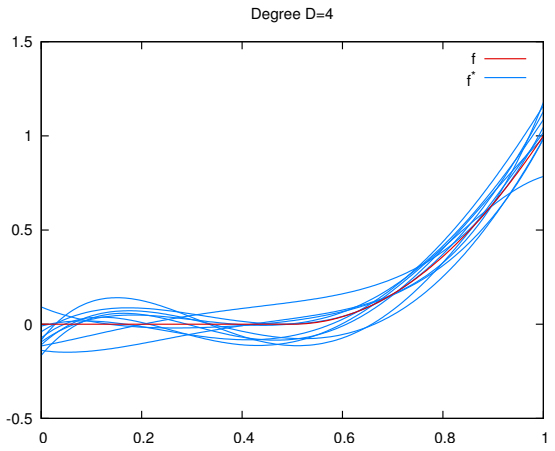
Slide credit: F. Fleuret

Regularization



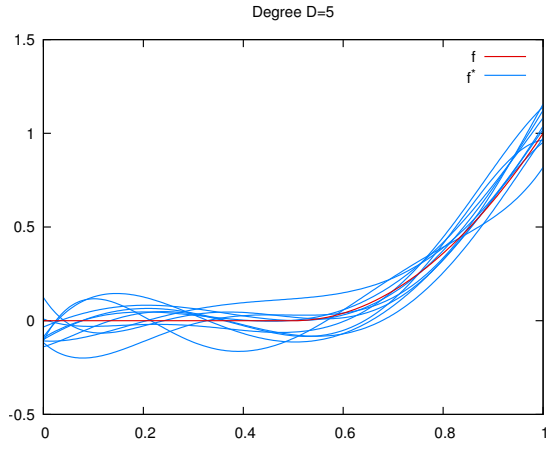
Slide credit: F. Fleuret

Regularization



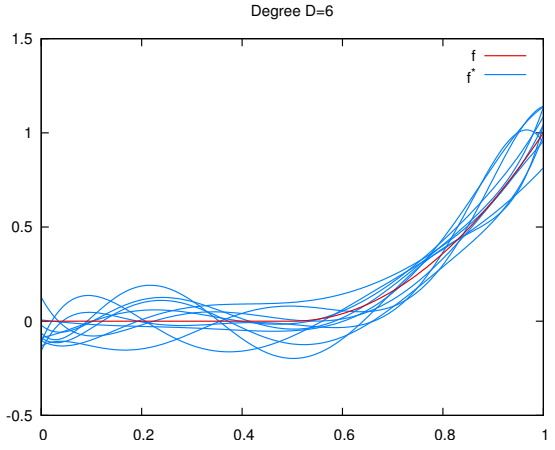
Slide credit: F. Fleuret

Regularization



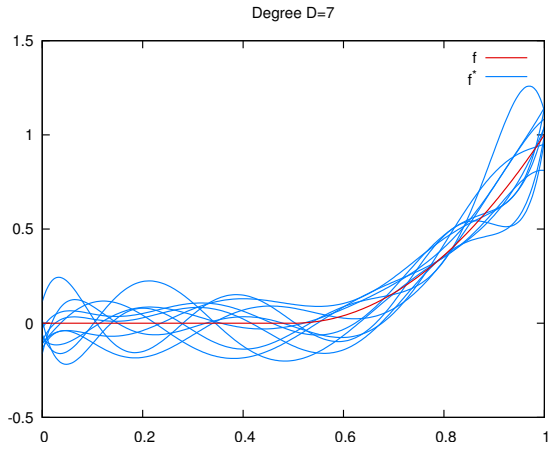
Slide credit: F. Fleuret

Regularization



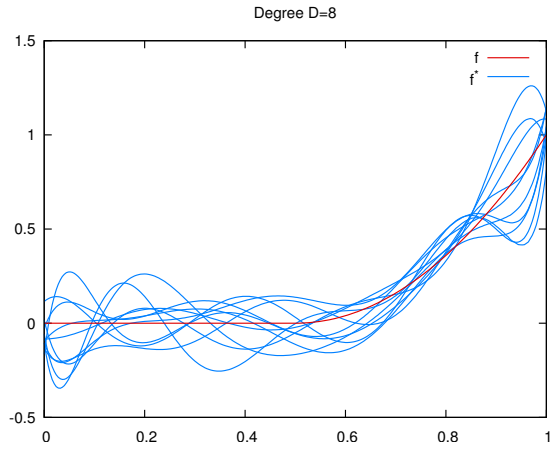
Slide credit: F. Fleuret

Regularization



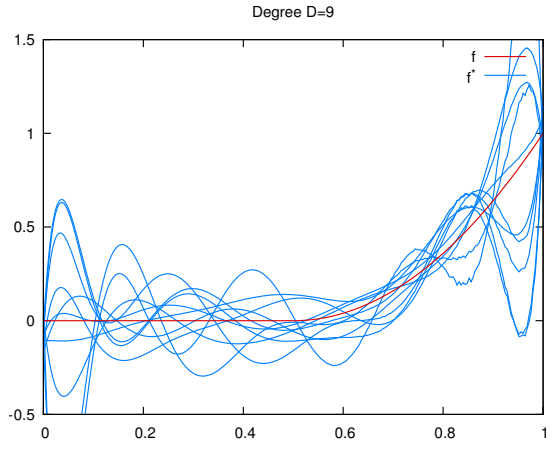
Slide credit: F. Fleuret

Regularization



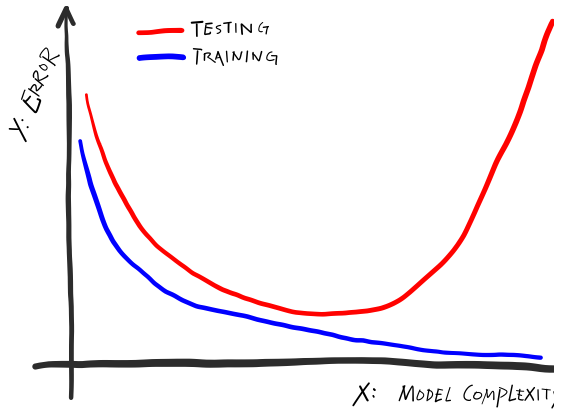
Slide credit: F. Fleuret

Regularization



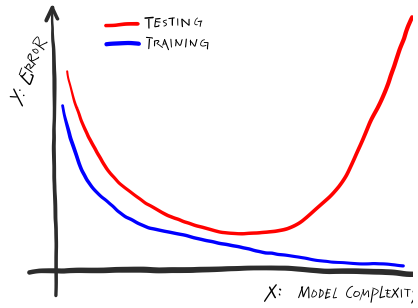
Slide credit: F. Fleuret

Regularization



Slide credit: C. Zhang

Regularization

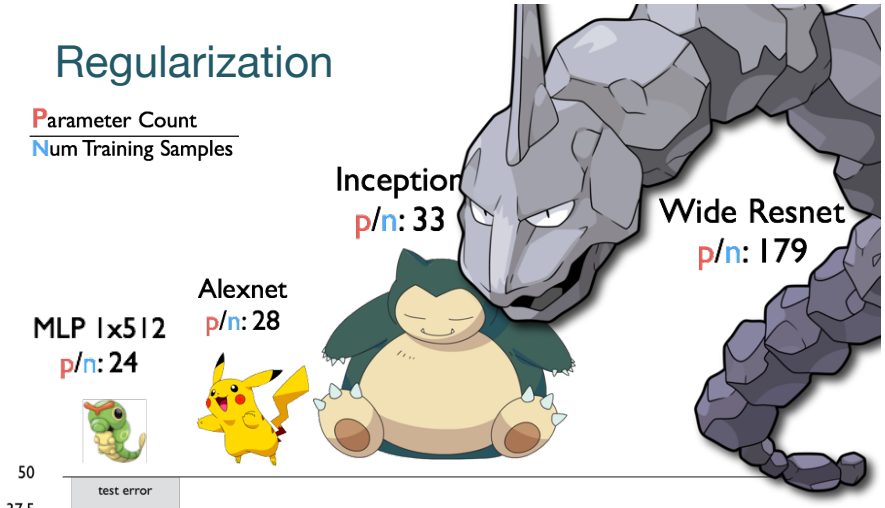


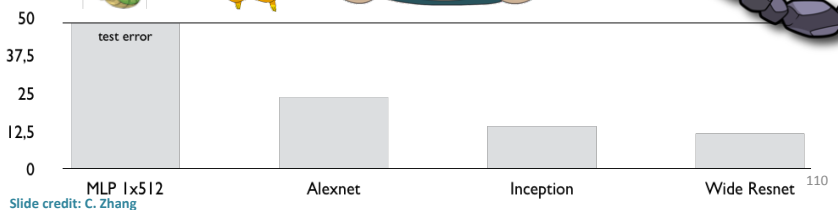
Deep Learning ★

Slide credit: C. Zhang

Regularization

Parameter Count / Num Training Samples





Slide credit: C. Zhang

Regularization

A REGULARIZER is ~~a mechanism~~ anything that ~~constrain the model or empower the data.~~ hurt the training process.

Slide credit: C. Zhang

Regularization

We can reformulate this control of the degree with a penalty

$$\mathscr{L}(\alpha) = \sum_n (f(x_n;\alpha) - y_n)^2 + \sum_d l_d(\alpha_d)$$

where

$$l_d(\alpha) = \begin{cases} 0 & \text{if } d \leq D \text{ or } \alpha = 0 \\ +\infty & \text{otherwise.} \end{cases}$$

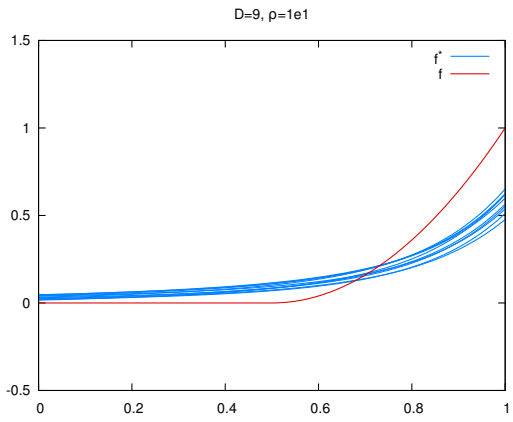
Such a penalty kills any term of degree $> D$.

This motivates the use of more subtle variants. For instance, to keep all this quadratic

$$\mathscr{L}(\alpha) = \sum_n (f(x_n;\alpha) - y_n)^2 + \rho \sum_d \alpha_d^2.$$

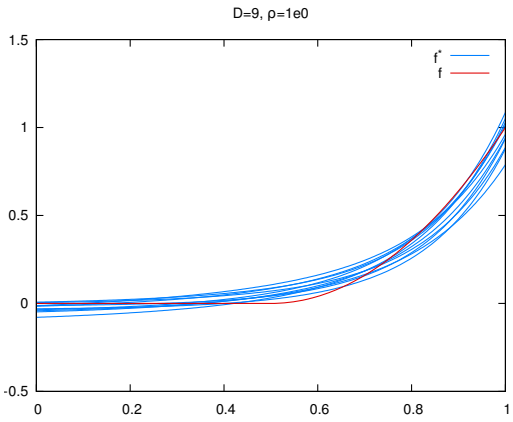
Slide credit: F. Fleuret

Regularization



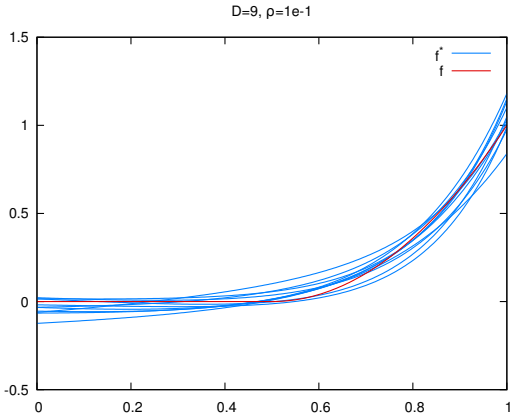
Slide credit: F. Fleuret

Regularization



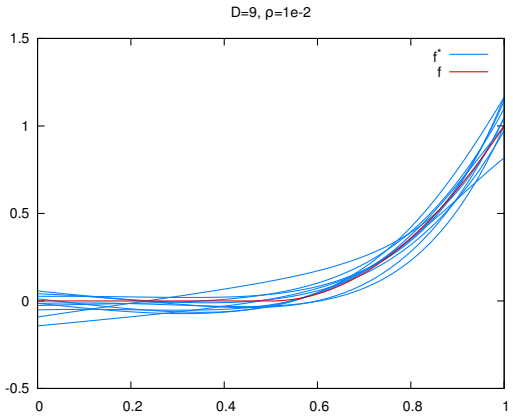
Slide credit: F. Fleuret

Regularization



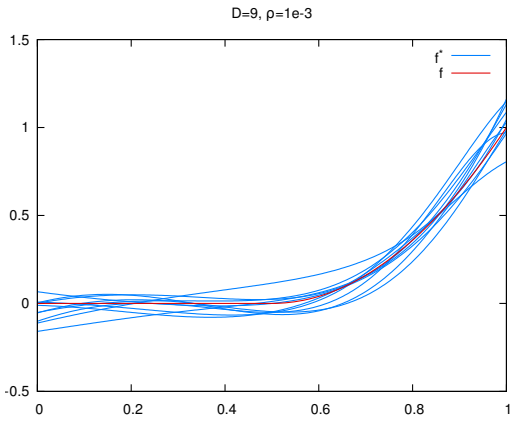
Slide credit: F. Fleuret

Regularization



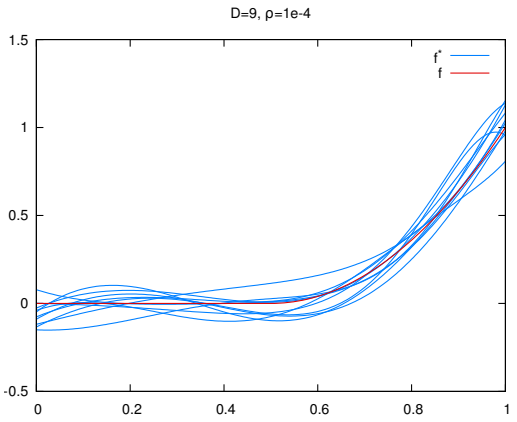
Slide credit: F. Fleuret

Regularization



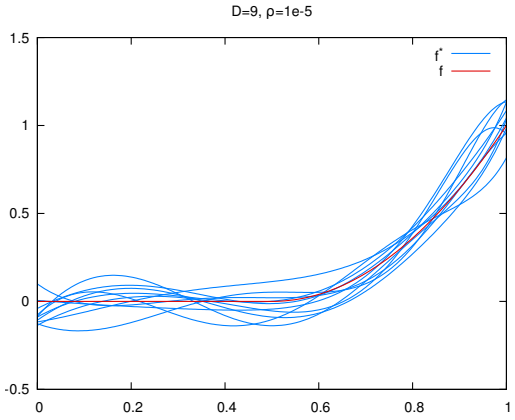
Slide credit: F. Fleuret

Regularization



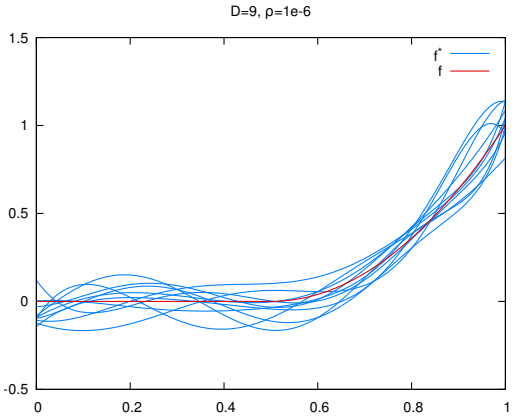
Slide credit: F. Fleuret

Regularization



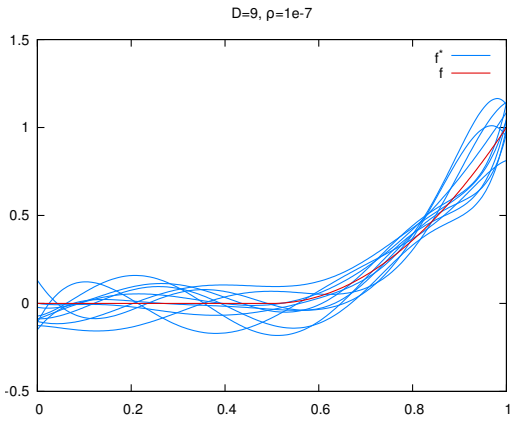
Slide credit: F. Fleuret

Regularization



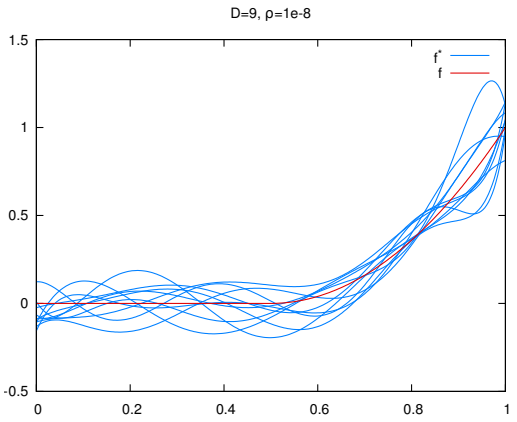
Slide credit: F. Fleuret

Regularization



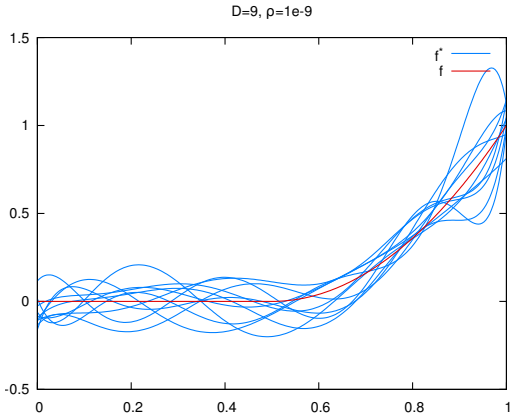
Slide credit: F. Fleuret

Regularization



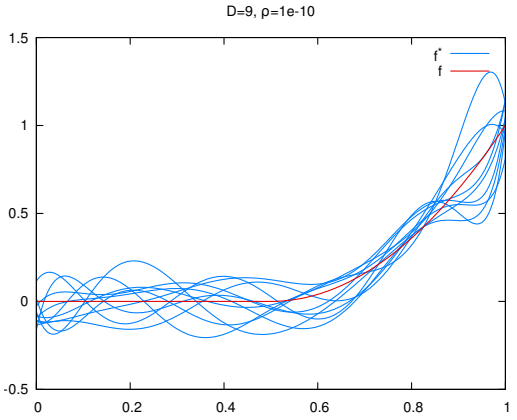
Slide credit: F. Fleuret

Regularization



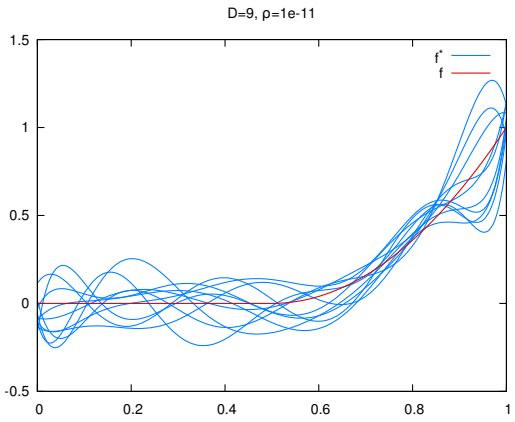
Slide credit: F. Fleuret

Regularization



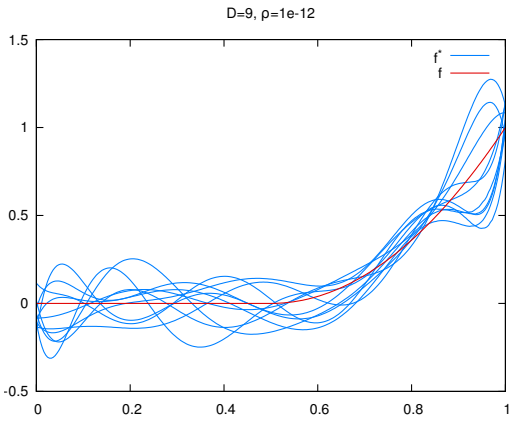
Slide credit: F. Fleuret

Regularization



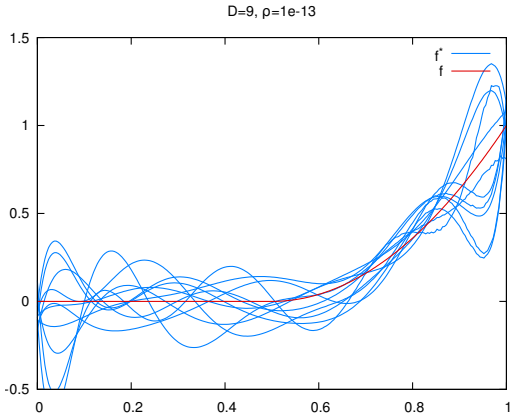
Slide credit: F. Fleuret

Regularization



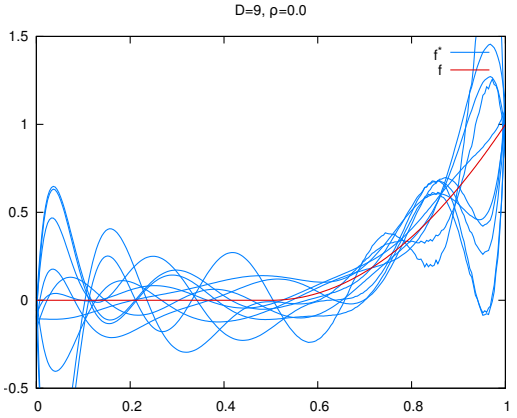
Slide credit: F. Fleuret

Regularization



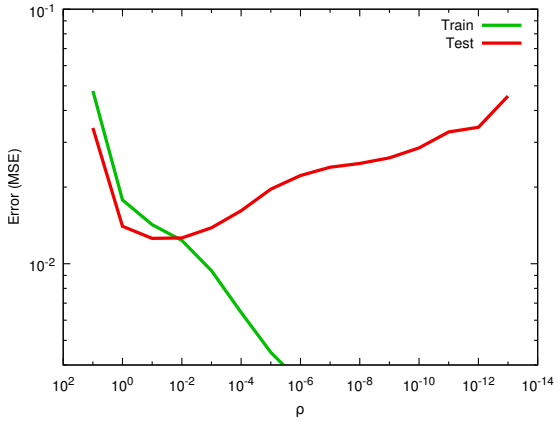
Slide credit: F. Fleuret

Regularization



Slide credit: F. Fleuret

Regularization



Slide credit: F. Fleuret

Regularization

- We define the capacity of a set of predictors as its ability to model an arbitrary functional
- Although it is difficult to define precisely, it is quite clear in practice how to increase or decrease it for a given class of models

Regularization

- Overfitting can be controlled by:
  - Impoverishing the space of functionals (less functionals, early stopping)
  - Make the choice of f* less dependent on data (penalty on coefficients, margin maximization, ensemble methods)

Loss versus Error

- Loss:
  - Measure how happy are we with the performance of the network on a given sample
  - Is differentiable and allows our model to correct itself towards better performance
  - Typically dependent on the problem and can be combined with other losses
- Error:
  - Measures the overall performance for the task hand, i.e. statistics over the entire dataset
  - Different flavors: accuracy (top1), top5, precision, recall, F1-score, Mean Average Precision, SNR, etc.
  - Typically non-differentiable and difficult to insert in training pipeline

Recap

- Image classification
- K-Nearest Neighbors
- Linear classification
- Multi-class SVM and Softmax
- Regularization

Next time

- Gradient descent
- Hand-crafted features
- Neural Networks
- Backpropagation