CPSC 340 Assignment 5 (due March 31 at 11:59pm)

Latent-Factor Models

Instructions

Rubric: {mechanics:3}

The above points are allocated for following the general homework instructions.

As usual, if you're using Python 2:

- Add `from __future__ import division` to the top of each Python file.
- Grab the Python 2 compatible data files from the "home" repo on GitHub.

Attention Python 3 users: this time you need to grab the data files from the "home" repo **even if you're using Python 3**. This has to do with one of the files being over 1 MB, which makes it more difficult to distribute to your individual repos in the usual way.

1 Principal Component Analysis

1.1 PCA by Hand

Rubric: {reasoning:3}

Consider the following dataset, containing 5 examples with 2 features each:

x_1	x_2
-2	-1
-1	0
0	1
1	2
2	3

Recall that with PCA we usually assume that the PCs are normalized ($\|w\| = 1$), we need to center the data before we apply PCA, and that the direction of the first PC is the one that minimizes the orthogonal distance to all data points.

1. What is the first principal component?
2. What is the (L2-norm) reconstruction error of the point (3,3)? (Show your work.)
3. What is the (L2-norm) reconstruction error of the point (3,4)? (Show your work.)

Answers:
1. The first column is equal to the second column if we take the mean of the second column and subtract it. Therefore, the columns are:

x_1	x_2
-2	-2
-1	-1
0	0
1	1
2	2

With PCA we usually assume that the PCs are normalized ($\|w\| = 1$). We can see x1=x2 here and $\sqrt{w_1^2 + w_2^2} = 1 => w = \frac{1}{\sqrt{2}}$.
$\sqrt{w_1} = 1$

2. To calculate reconstruction error we need to break it down and form it again.

To compress the data :Subtracting the mean and multiply by $\sqrt{w}$
$3 - 0$*w $+ 3 - 1$*w$= \frac{5}{\sqrt{2}}$

To form it again,multiply and adding the means: $\frac{5}{\sqrt{2}}$*w + 0=2.5 $\frac{5}{\sqrt{2}}$*w + 1=3.5

Finally Reconstruction error, $\sqrt{(3.5-3)^2 + (2.5-3)^2} = \frac{\sqrt{2}}{2}$

3. Same procedure as above:
$3 - 0$*w $+ 4 - 1$*w$= \frac{6}{\sqrt{2}}$
Form it again: $\frac{6}{\sqrt{2}}$*w + 0=3 $\frac{6}{\sqrt{2}}$*w + 1=4

Reconstruction error:
$\sqrt{(3-3)^2 + (4-4)^2} = \frac{\sqrt{2}}{2} = 0$

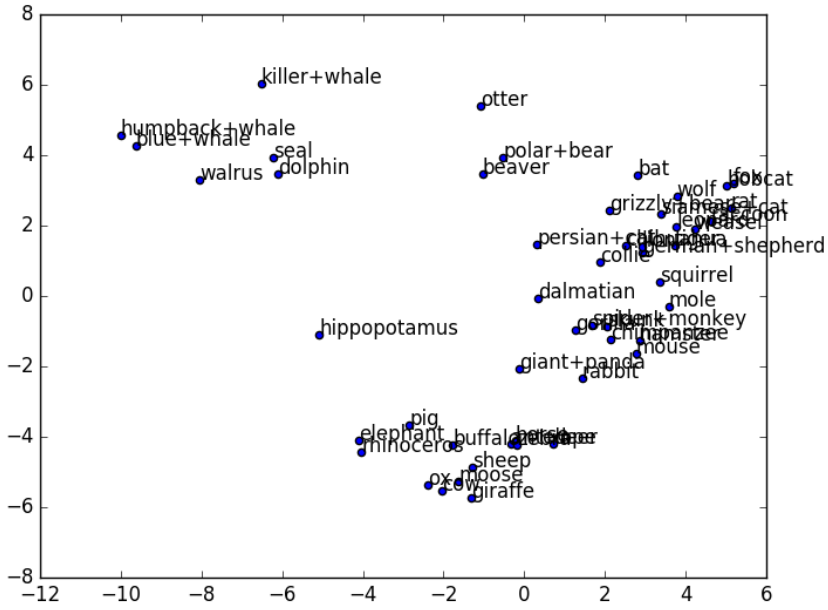
1.2 Data Visualization

Rubric: {reasoning:2}

The command *main -q 1.2* will load the animals dataset from a previous assignment, standardize the features, and then give two unsatisfying visualizations of it. First it shows a plot of the matrix entries, which has too much information and thus gives little insight into the relationships between the animals. Next it shows a scatterplot based on two random features. We label some random points, but because of the binary features even a scatterplot matrix will show us almost nothing about the data.

The class *pca.PCA* applies the classic PCA method (orthogonal bases via SVD) for a given k. Using this class, modify the demo so that the scatterplot uses the latent features z_i from the PCA model. Make a scatterplot of the two columns in Z, and label a bunch of the points in the scatterplot. Hand in your modified demo and the scatterplot.

The code can be found at `https://github.ubc.ca/cpsc340/jeanlam_nafis1_hw5/blob/master/code/main.py`
Figures generated:



1.3 Data Compression

Rubric: {reasoning:2}

It is important to know how much of the information in our dataset is captured by the low-dimensional PCA representation. In class we discussed the "analysis" view that PCA maximizes the variance that is explained by the PCs, and the connection between the Frobenius norm and the variance of a centered data matrix X. Use this connection to answer the following:

1. How much of the variance is explained by our two-dimensional representation from the previous question?
2. How many PCs are required to explain 50% of the variance in the data?

Answer:
1. 0.3019 is the variance explained by our two-dimensional representation from the previous question.

2.k = 5 can explain 50% of the variance in the data.
See the variance code in the comment in 1.2.

2 PCA Generalizations

2.1 Robust PCA

Rubric: {code:4}

The command *main -q 2.1* loads a dataset X where each row contains the pixels from a single frame of a video of a highway. The demo applies PCA to this dataset and then uses this to reconstruct the original

image. It then shows the following 3 images for each frame (pausing and waiting for input between each frame):

1. The original frame.
2. The reconstruction based on PCA.
3. A binary image showing locations where the reconstruction error is non-trivial.

Recently, latent-factor models have been proposed as a strategy for "background subtraction": trying to separate objects from their background. In this case, the background is the highway and the objects are the cars on the highway. In this demo, we see that PCA does an ok job of identifying the cars on the highway in that it does tend to identify the locations of cars. However, the results aren't great as it identifies quite a few irrelevant parts of the image as objects.

Robust PCA is a variation on PCA where we replace the L2-norm with the L1-norm,

$$f(Z, W) = \sum_{i=1}^{n} \sum_{j=1}^{d} |w_j^T z_i - x_{ij}|,$$

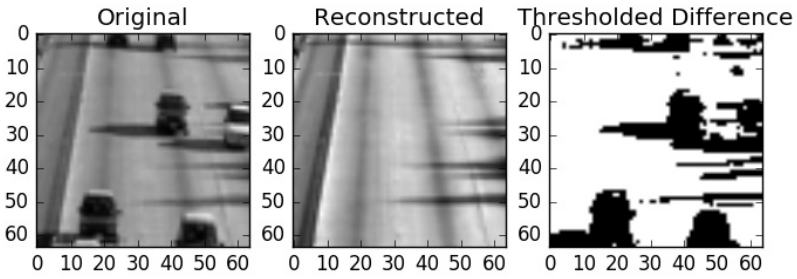
and it has recently been proposed as a more effective model for background subtraction. Complete the class *pca.RobustPCA*, that uses a smooth approximation to the absolute value to implement robust PCA.

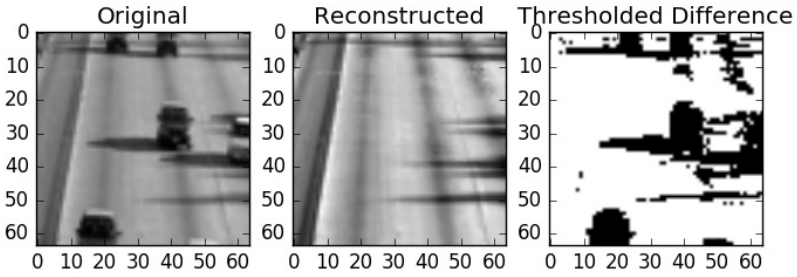
Hint: most of the work has been done for you in the class *pca.AlternativePCA*. This work implements an alternating minimization approach to minimizing the (squared) PCA objective (without enforcing orthogonality). This gradient-based approach to PCA can be modified to use a smooth approximation of the L1-norm. Note that the log-sum-exp approximation to the absolute value may be hard to get working due to numerical issues, and a numerically-nicer approach is to use the "multi-quadric" approximation:

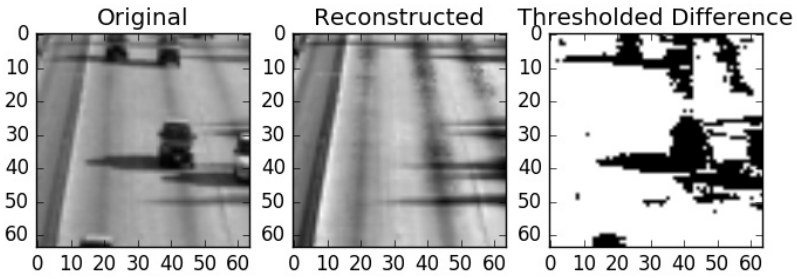
$$|\alpha| \approx \sqrt{\alpha^2 + \epsilon},$$

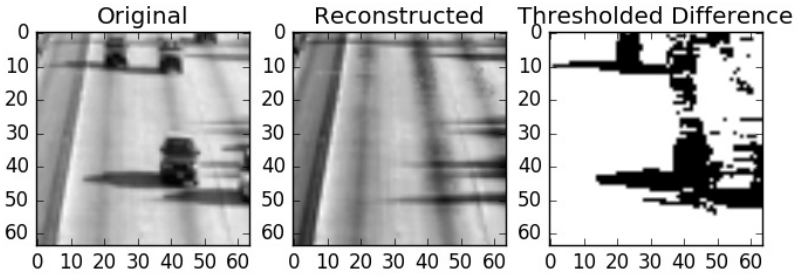
where ϵ controls the accuracy of the approximation (a typical value of ϵ is 0.0001).

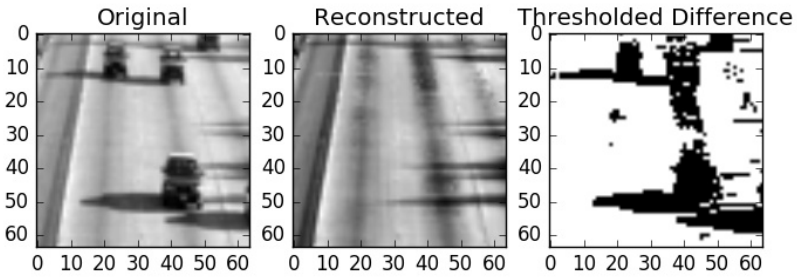
The code can be found at `https://github.ubc.ca/cpsc340/jeanlam_nafis1_hw5/blob/master/code/pca.py`
Figures generated:

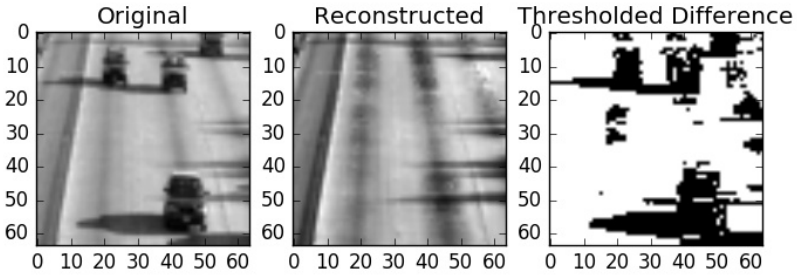

Original
Reconstructed
Thresholded Difference
0 10 20 30 40 50 60

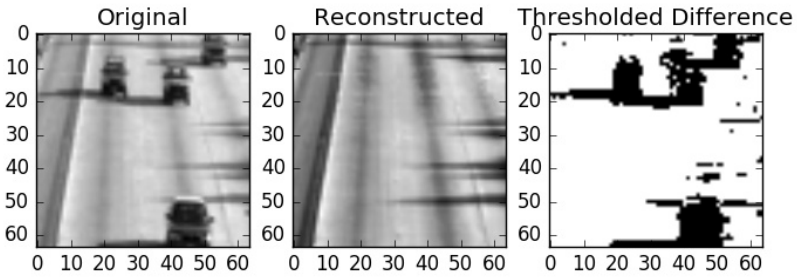

Original
Reconstructed
Thresholded Difference
0 10 20 30 40 50 60

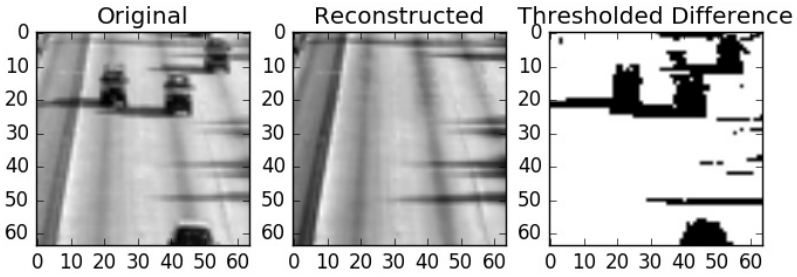

Original
Reconstructed
Thresholded Difference
0
10
20
30
40
50
60

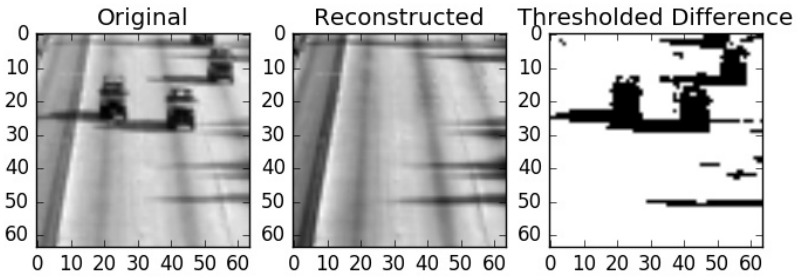

Original
Reconstructed
Thresholded Difference
0
10
20
30
40
50
60

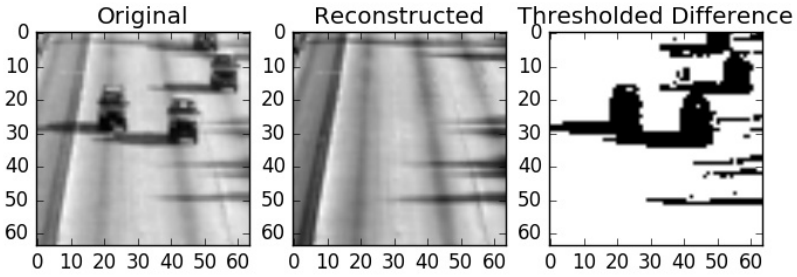

Original
Reconstructed
Thresholded Difference
0 10 20 30 40 50 60


Original
Reconstructed
Thresholded Difference
0 10 20 30 40 50 60


Original
Reconstructed
Thresholded Difference
0 10 20 30 40 50 60


Original
Reconstructed
Thresholded Difference
0 10 20 30 40 50 60





2.2 L1-Regularized and Binary Latent-Factor Models

Rubric: {reasoning:5}

We have a matrix X, where we have observed a subset of its individual elements. Let $\mathcal{R}$ be the set of indices

(i, j) where we have observed the element x_{ij}. We want to build a model that predicts the missing entries, so we use a latent-factor model with an L1-regularizer on the coefficients W and a separate L2-regularizer on the coefficients Z,

$$f(Z, W) = \frac{1}{2} \sum_{(i,j) \in \mathcal{R}} \left[(w_j^T z_i - x_{ij})^2 \right] + \lambda_W \sum_{j=1}^{d} \left[\|w_j\|_1 \right] + \frac{\lambda_Z}{2} \sum_{i=1}^{n} \left[\|z_i\|^2 \right],$$

where the regularization parameters satisfy $\lambda_W > 0$ and $\lambda_Z > 0$.

1. What is the effect of λ_W on the sparsity of the parameters W and Z? What is the effect of λ_Z on the sparsity of W and Z?
2. What is the effect of λ_Z on the two parts of the fundamental trade-off in machine learning? What is the effect of k on the two parts?
3. Would the answers to (2) change if $\lambda_W = 0$?
4. Suppose each element of the matrix X is either $+1$ or -1 and our goal is to build a model that makes the sign of $w_j^T z_i$ match the sign of x_{ij}. Write down a (continuous) objective function that would be more suitable.

Answer:
1. W is directly related to λ_W. So as λ_W increase so W will become more sparse.
λ_W does not affect Z.
There is no effect of λ_Z on sparsity of Z and W.

2. If λ_Z goes up training error will go up too but lead to less overfitting.The training error will approximate the test error better.
If k is increased the training error will decrease but will lead to overfitting.

3. λ_Z will not be effective anymore if $\lambda_W = 0$ because then we can cancel the effect of Z by changing W as there is no regularizer on W anymore. So λ_Z will not be effective anymore.

K is not affected by λ_W, so it's effect stays the same.

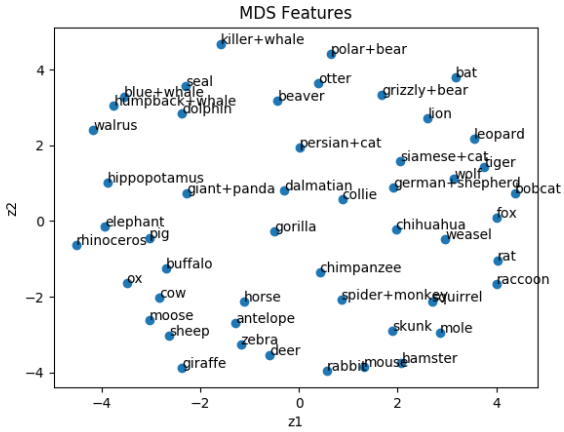
4. We can use the logistic regression since we are taking the sign of ${w_j}^T$ to match the sign of x_{ij}. Therefore the new objective function is : $\sum_{i,j}^{n}[\log(1 + \exp(-x_{ij} w_j^T z_i))] + \lambda_W \sum_{j=1}^{k} \left[\|w_j\|_1 \right] + \frac{\lambda_Z}{2} \sum_{i=1}^{n} \left[\|z_i\|^2 \right]$

3 Multi-Dimensional Scaling

The command *main -q 3* loads the animals dataset and then applies gradient dsecent to minimize the following multi-dimensional scaling (MDS) objective (starting from the PCA solution):

$$f(Z) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} (\|z_i - z_j\| - \|x_i - x_j\|)^2. \tag{1}$$

The result of applying MDS is shown below.



Although this visualization isn't perfect (with "gorilla" being placed close to the dogs and "otter" being placed close to two types of bears), this visualization does organize the animals in a mostly-logical way.

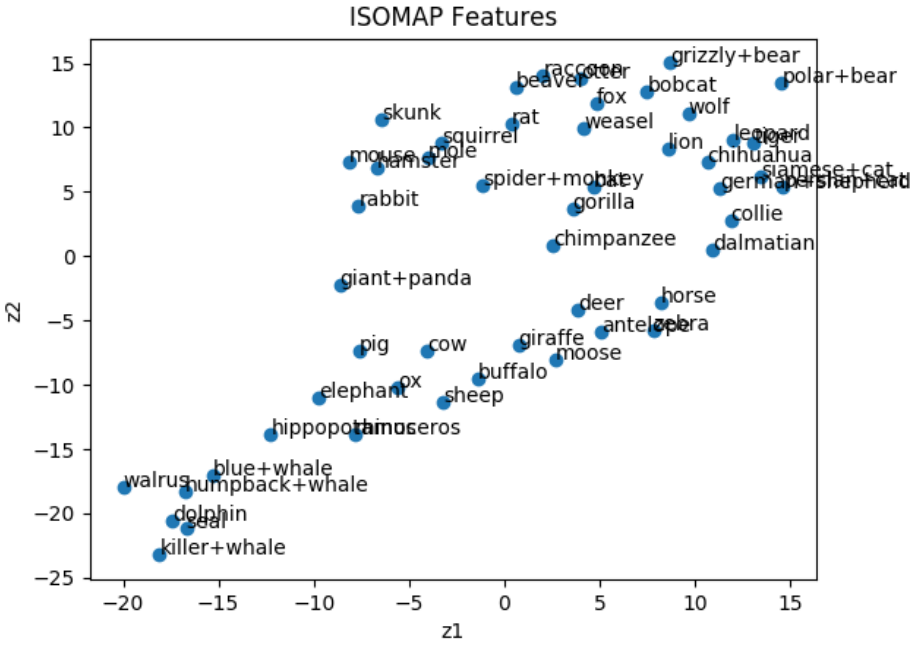
3.1 ISOMAP

Rubric: {code:4}

Euclidean distances between very different animals are unlikely to be particularly meaningful. However, since related animals tend to share similar traits we might expect the animals to live on a low-dimensional manifold. This suggests that ISOMAP may give a better visualization. Make a new class *ISOMAP* that computes the approximate geodesic distance (shortest path through a graph where the edges are only between nodes that are k-nearest neighbour) between each pair of points, and then fits a standard MDS model (**??**) using gradient descent. Hand in your code and the plot of the result when using the 3-nearest neighbours.

Hint: the function *utils.dijskstra* can be used to compute the shortest (weighted) distance between two points in a weighted graph. This function requires an n by n matrix giving the weights on each edge (use 0 as the weight for absent edges). Note that ISOMAP uses an undirected graph, while the k-nearest neighbour graph might be asymmetric. One of the usual heuristics to turn this into a undirected graph is to include an edge i to j if i is a KNN of j or if j is a KNN of i. (Another possibility is to include an edge only if i and j are mutually KNNs.)

The code can be found at `https://github.ubc.ca/cpsc340/jeanlam_nafis1_hw5/blob/master/code/manifold.py` in a class called ISOMAP
Figure generated:

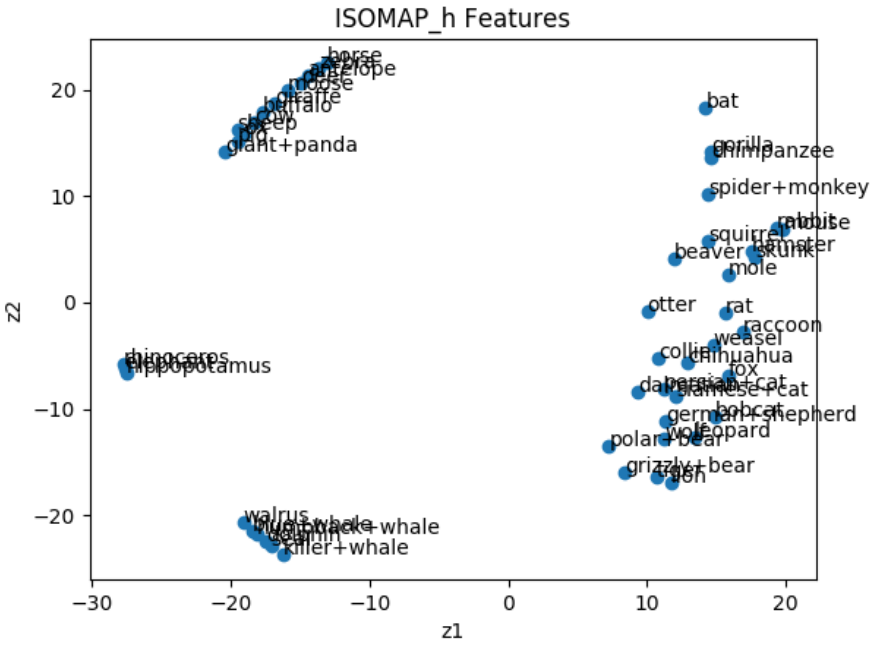


3.2 ISOMAP with Disconnected Graph

Rubric: {code:2}

An issue with measuring distances on graphs is that the graph may not be connected. For example, if you run your ISOMAP code with 2-nearest neighbours then some of the distances are infinite. One heuristic to address this is to set these infinite distances to the maximum distance in the graph (i.e., the maximum geodesic distance between any two points that are connected), which will encourage non-connected points to be far apart. Modify your ISOMAP function to implement this heuristic. Hand in your code and the plot of the result when using the 2-nearest neighbours.

The code can be found at `https://github.ubc.ca/cpsc340/jeanlam_nafis1_hw5/blob/master/code/manifold.py` in a class called $ISOMAP_h$(ISOMAP underscore h)
Figure generated:


ISOMAP_h Features
z1
z2
bat
gorilla
chimpanzee
spider+monkey
squirrel
beaver
skunk
mole
otter
rat
raccoon
weasel
chihuahua
fox
persian+cat
siamese+cat
bobcat
german+shepherd
leopard
polar+bear
grizzly+bear
tiger
giant+panda
hippopotamus
walrus
killer+whale