# CS540 Machine learning
# Lecture 5

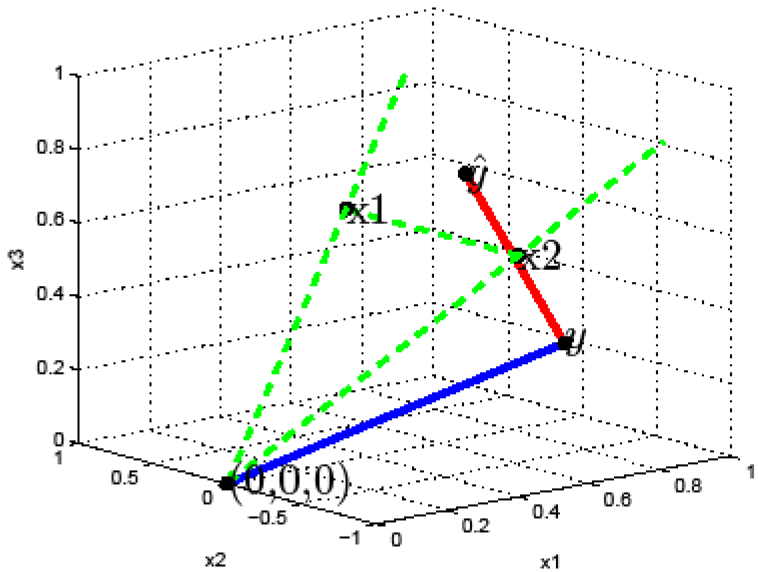
# Last time

- Basis functions for linear regression
- Normal equations
- QR
- SVD - briefly

# This time

- Geometry of least squares (again)
- SVD – more slowly
- LMS
- Ridge regression

# Geometry of least squares




Columns of X define a d-dimensional linear subspace in n-dimensions.
Yhat is projection of y into that subspace.
Here n=3, d=2.

$$\mathbf{X} = \begin{pmatrix} 1 & 2 \\ 1 & -2 \\ 1 & 2 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} 8.8957 \\ 0.6130 \\ 1.7761 \end{pmatrix}, \quad \hat{\mathbf{y}} = X\hat{\mathbf{w}} = \begin{pmatrix} 5.3359 \\ 0.6130 \\ 5.3359 \end{pmatrix}$$

$$\mathbf{X} = \begin{pmatrix} 0.5774 & 0.5774 \\ 0.5774 & -0.5774 \\ 0.5774 & 0.5774 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} 0.9784 \\ 0.0674 \\ 0.1954 \end{pmatrix}, \quad \hat{\mathbf{y}} = \begin{pmatrix} 0.7048 \\ 0.0810 \\ 0.7048 \end{pmatrix}$$

Unit norm

# Orthogonal projection

- Projection of y onto X

$$\mathrm{Proj}(\mathbf{y};\mathbf{X}) = \mathrm{argmin}_{\hat{\mathbf{y}} \in \mathrm{span}(\{\mathbf{x}_1,\ldots,\mathbf{x}_n\})} \|\mathbf{y} - \hat{\mathbf{y}}\|_2.$$

- Let r = y - \hat{y}. Residual must be orthogonal to X. Hence

$$\mathbf{x}_j^T(\mathbf{y} - \hat{\mathbf{y}}) = 0 \Rightarrow \mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0} \Rightarrow \mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- Prediction on training set

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \stackrel{\text{def}}{=} \mathbf{H}\mathbf{y}$$

Hat matrix

- Residual is orthogonal

$$\mathbf{X}^T(\mathbf{y} - \mathbf{H}\mathbf{y}) = \mathbf{X}^T(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}) = \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{0}$$

# This time

- Geometry of least squares (again)
- SVD – more slowly
- LMS
- Ridge regression

# Eigenvector decomposition (EVD)

- For any square matrix A, we say λ is an eval and u is its evec if

$$\mathbf{A}\mathbf{u} = \lambda \mathbf{u}, \quad \mathbf{u} \neq 0 \ .$$

- Stacking up all evecs/vals gives

$$\mathbf{A}\mathbf{U} = \mathbf{U}\boldsymbol{\Lambda} = \begin{pmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \\ | & | & & | \end{pmatrix} \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}$$

- If evecs linearly independent

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}.$$ diagonalization

# EVD of symmetric matrices

- If A is symmetric, all its evals are real, and all its evecs are orthonormal, $u_i^T u_j = \delta_{ij}$
- Hence $\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}, \ \ |\mathbf{U}| = 1.$
- and

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = \sum_{i=1}^{n} \lambda_i \mathbf{u}_i \mathbf{u}_i^T$$

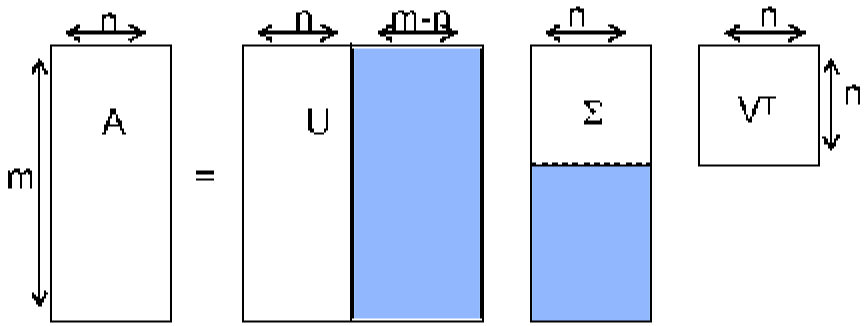
$$\begin{aligned}
\mathbf{A} &= \ = \begin{pmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \\ | & | & & | \end{pmatrix} \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} \begin{pmatrix} - & \mathbf{u}_1^T & - \\ - & \mathbf{u}_2^T & - \\ & \vdots & \\ - & \mathbf{u}_n^T & - \end{pmatrix} \\
&= \lambda_1 \begin{pmatrix} | \\ \mathbf{u}_1 \\ | \end{pmatrix} \begin{pmatrix} - & \mathbf{u}_1^T & - \end{pmatrix} + \cdots + \lambda_n \begin{pmatrix} | \\ \mathbf{u}_n \\ | \end{pmatrix} \begin{pmatrix} - & \mathbf{u}_n^T & - \end{pmatrix}
\end{aligned}$$

# SVD

For any real matrix

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sigma_1 \begin{pmatrix} | \\ \mathbf{u}_1 \\ | \end{pmatrix} \begin{pmatrix} - & \mathbf{v}_1^T & - \end{pmatrix} + \cdots + \sigma_r \begin{pmatrix} | \\ \mathbf{u}_r \\ | \end{pmatrix} \begin{pmatrix} - & \mathbf{v}_r^T & - \end{pmatrix}$$

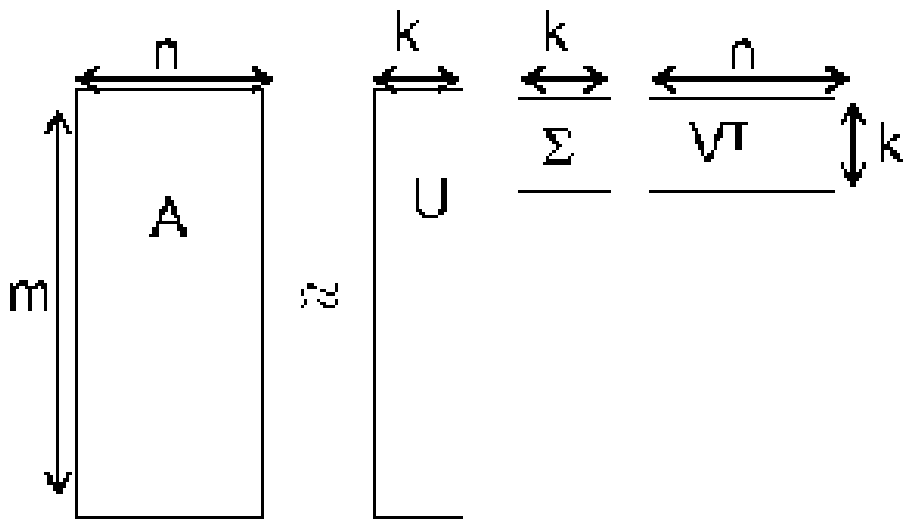
$$\begin{aligned} \mathbf{U}^T\mathbf{U} &= \mathbf{I} \\ \mathbf{V}^T\mathbf{V} &= \mathbf{V}\mathbf{V}^T = \mathbf{I} \end{aligned}$$



# Truncated SVD

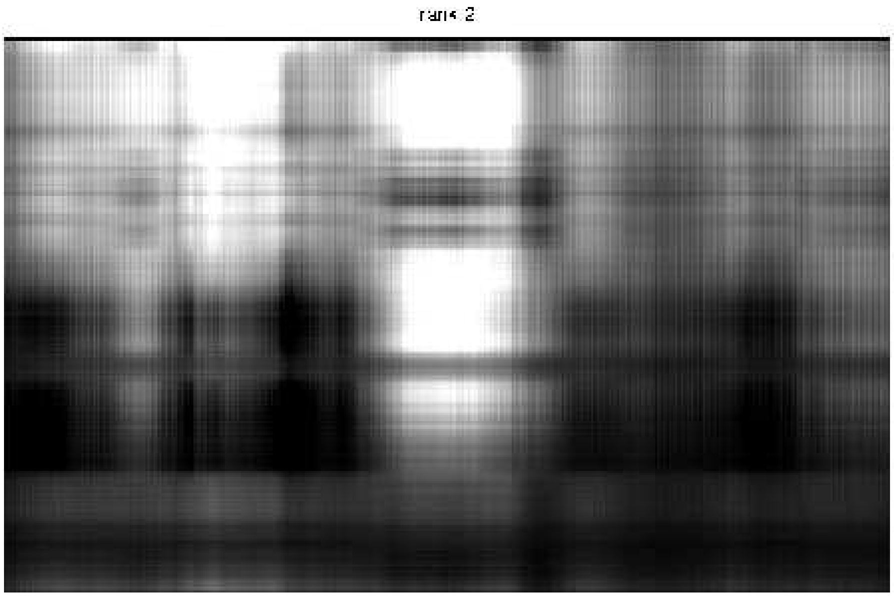
- Rank k approximation to a matrix

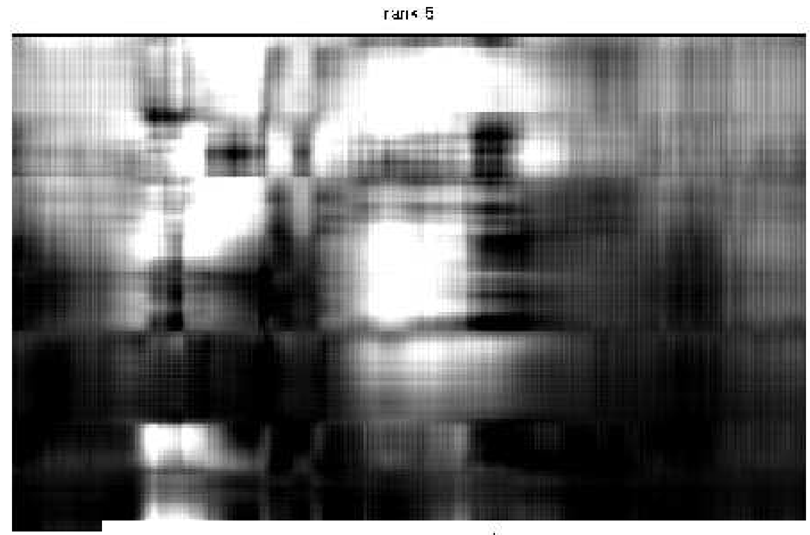
$$\mathbf{A}_k = \sum_{j=1}^{k} \sigma_j \mathbf{u}_j \mathbf{v}_k^T = \mathbf{U}_{:,1:k} \ \ \mathbf{\Sigma}_{1:k,1:k} \ \ \mathbf{V}_{:,1:k}^T$$

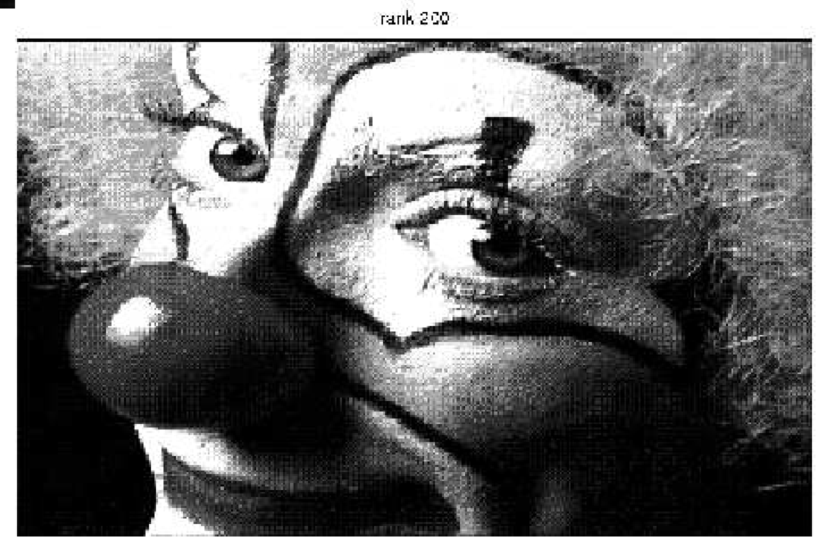



Equivalent to PCA

# Truncated SVD




```
load clown; % built-in image
[U,S,V] = svd(X,0);
k = 20;
Xhat = (U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
image(Xhat);
```

# SVD and EVD

- If A is symmetric positive definite, then svals(A)=evals(A), leftSvecs(A)=rightSvecs(A)=evecs(A) modulo sign changes

```
>> A=randpd(3)
A =
    0.9302    0.4036    0.7065
    0.4036    0.8049    0.4521
    0.7065    0.4521    0.5941

>> [U,Lam]=eig(A)
U =
    0.5476    0.5148    0.6597
    0.1872   -0.8437    0.5030
   -0.8155    0.1520    0.5584
Lam =
    0.0159         0         0
         0    0.4772         0
         0         0    1.8361
```

```
>> [U,S,V]=svd(A)
U =
   -0.6597    0.5148   -0.5476
   -0.5030   -0.8437   -0.1872
   -0.5584    0.1520    0.8155
S =
    1.8361         0         0
         0    0.4772         0
         0         0    0.0159
V =
   -0.6597    0.5148   -0.5476
   -0.5030   -0.8437   -0.1872
   -0.5584    0.1520    0.8155
```

# SVD and EVD

- For arbitrary real matrix A
- leftSvecs(A) = evecs(A A’)
- rightSvecs(A) = evecs(A’ A)
- Svals(A)^2 = evals(A’ A) = evals(A A’)

# SVD for least squares

- We have $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T$

$$
\begin{aligned}
\hat{\mathbf{w}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \\
\mathbf{X}^T\mathbf{X}\mathbf{w} &= \mathbf{X}^T\mathbf{y} \text{ (premultiply by } \mathbf{X}^T\mathbf{X}) \\
\mathbf{V}\mathbf{D}\mathbf{U}^T\mathbf{U}\mathbf{D}\mathbf{V}^T\mathbf{w} &= \mathbf{V}\mathbf{D}\mathbf{U}^T\mathbf{y} \text{ (SVD expansion)} \\
\mathbf{V}\mathbf{D}^2\mathbf{V}^T\mathbf{w} &= \mathbf{V}\mathbf{D}\mathbf{U}^T\mathbf{y} \text{ (since } \mathbf{U}^T\mathbf{U} = \mathbf{I} \text{ and } \mathbf{D}\mathbf{D} = \mathbf{D}^2) \\
\mathbf{D}^2\mathbf{V}^T\mathbf{w} &= \mathbf{D}\mathbf{U}^T\mathbf{y} \text{ (premultiply by } \mathbf{V}^T) \\
\mathbf{V}^T\mathbf{w} &= \mathbf{D}^{-1}\mathbf{U}^T\mathbf{y} \text{ (premultiply by } \mathbf{D}^{-2}) \\
\mathbf{w} &= \mathbf{V}\mathbf{D}^{-1}\mathbf{U}^T\mathbf{y} \text{ (premultiply by } \mathbf{V})
\end{aligned}
$$

```
[U,D,V]=svd(X,0);
Dinv = diag(1./(diag(D)));
w = V*Dinv*U'*y;
```

What if $D_j = 0$ (so rank of X is less than d)?

# Pseudo inverse

- If D_j=0, use

$$\mathbf{w} = \mathbf{V}\mathbf{D}^{\dagger}\mathbf{U}^T\mathbf{y} \stackrel{\text{def}}{=} \mathbf{X}^{\dagger}\mathbf{y}, \;\; \mathbf{D}^{\dagger} = \text{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0)$$

```
function B = pinv(A)
[U,S,V] = svd(A,0);
s = diag(S);
r = sum(s > tol); % rank
w = diag(ones(r,1) ./ s(1:r));
B = V(:,1:r) * w * U(:,1:r)';
```

- Of all solutions w that minimize ||Xw – y||, the pinv solution also minimizes ||w||

```
w = X\y;
w2 = pinv(X)*y;
[norm(w) norm(w2)]
>> 10.8449   10.8440
```

# This time

- Geometry of least squares (again)
- SVD – more slowly
- LMS
- Ridge regression

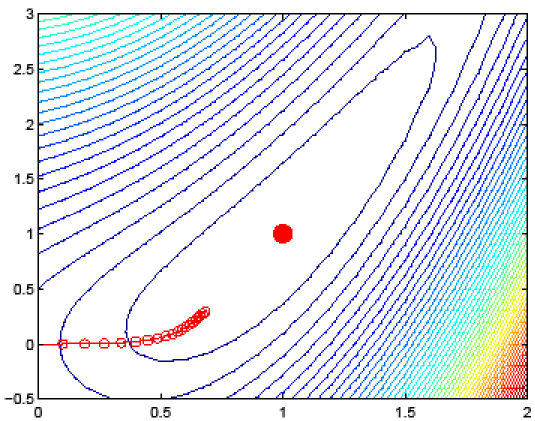
# Gradient descent

- QR and SVD take O(d$^3$) time
- We can find the MLE by following the gradient

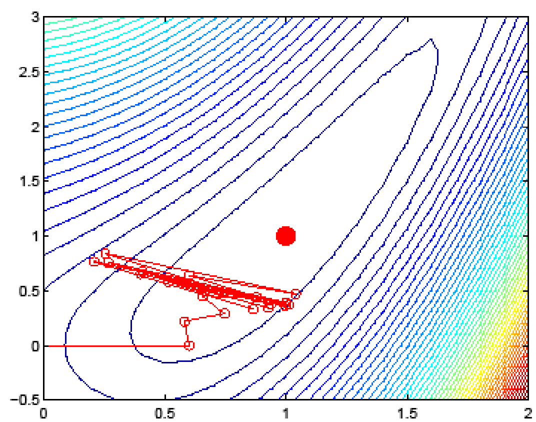
$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta_k \mathbf{g}(\mathbf{w}_k)$$

$$\mathbf{g}(\mathbf{w}) \propto \mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{y}) = \sum_{i=1}^{n} \mathbf{x}_i(\mathbf{w}^T\mathbf{x}_i - y_i)$$

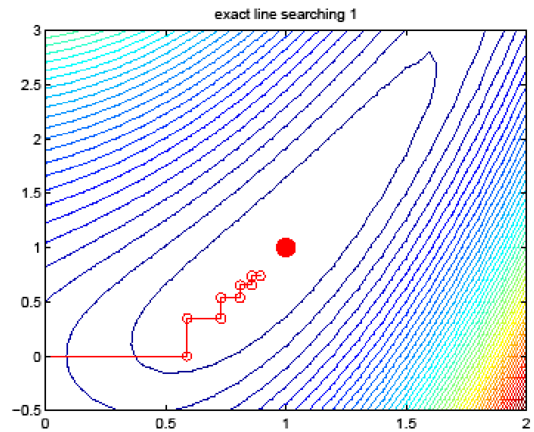
- O(d) per step, but may need many steps



η=0.1



η=0.6




Exact line search

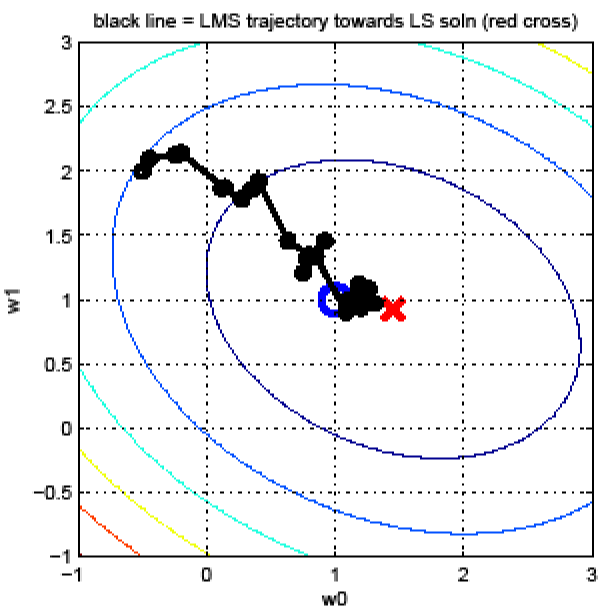
# Stochastic gradient descent

- Approximate the gradient by looking at a single data case

$$\mathbf{g}(\mathbf{w}_k) \approx \mathbf{x}_i(\mathbf{w}^T\mathbf{x}_i - y_i)$$

Least Mean Squared
Widrow-Hoff
Delta-rule

- Can be used to learn online




**Algorithm 1**: LMS algorithm

*Initialize* $\mathbf{w}$
$t \leftarrow 0$
**repeat**
    $t \leftarrow t + 1$
    $i \leftarrow t \bmod n$
    $\mathbf{w} \leftarrow \mathbf{w} + \eta(y_i - \mathbf{w}^T\mathbf{x}_i)\mathbf{x}_i$
    $\eta \leftarrow \eta \times s$
**until** *converged*

# This time

- Geometry of least squares (again)
- SVD – more slowly
- LMS
- Ridge regression

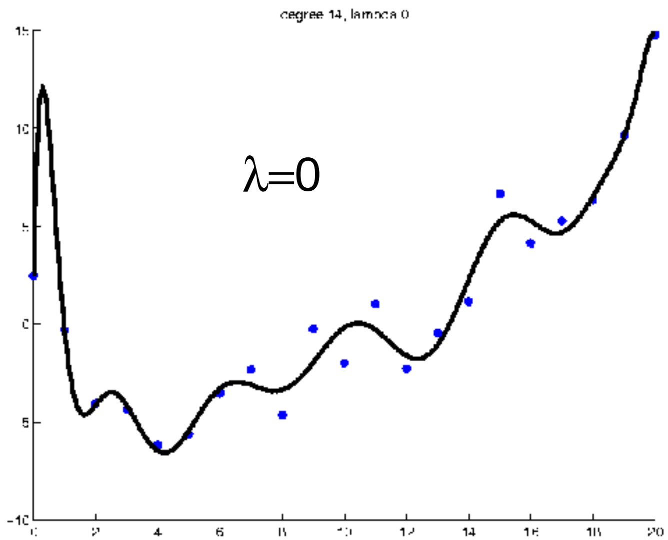
# Ridge regression

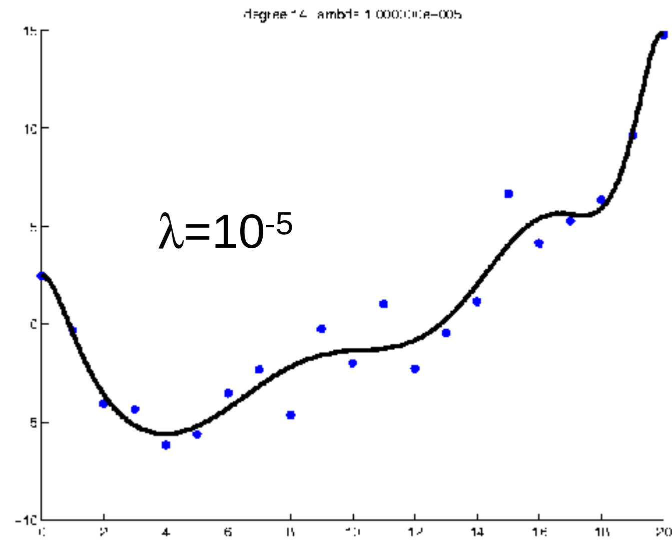
- Minimize penalized negative log likelihood

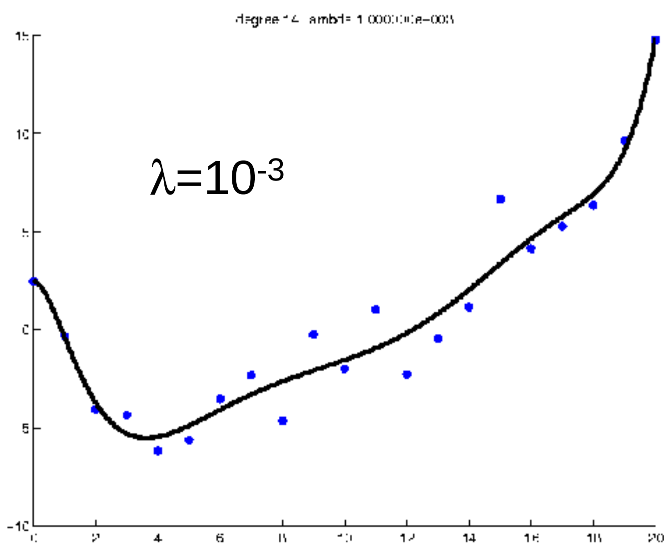
$$-\ell(\mathbf{w}) + \lambda ||\mathbf{w}||_2^2$$

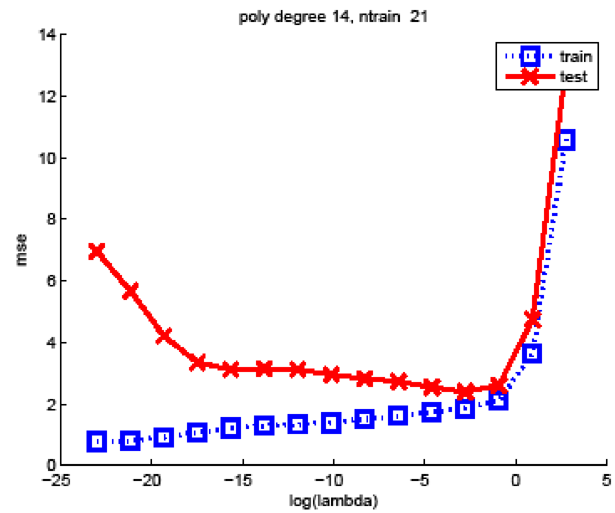
- Weight decay, shrinkage, L2 regularization, ridge regression

# Regularization D=14



# Why it works

- Coefficients if $\lambda=0$ (MLE)

```
-0.18, 10.57, -110.28, -245.63, 1664.41, 2647.81, -965
27669.94, 19319.66, -41625.65, -16626.90, 31483.81, 54
```

- Coefficients if $\lambda=10^{-3}$

```
-1.54,  5.52,  3.66, 17.04, -2.63, -23.06, -0.37, -8.49
7.92,  5.40,  8.29,  7.75,  1.78,  2.03, -8.42,
```

- Small weights mean the curve is almost linear (same is true for sigmoid function)

# Ridge regression

- The objective function is

$$\mathbf{w} = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \mathbf{w} - w_0)^2 + \lambda \sum_{j=1}^{d} w_j^2$$

- We don't shrink w_0. We should standardize first.
- Constrained formulation

$$\mathbf{w} = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \mathbf{w} - w_0)^2 \text{ s.t. } \sum_{j=1}^{d} w_j^2 \leq t$$

- Find the penalized MLE

$$J(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}$$

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$

See book

# QR

- Recall

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

- Expanded data:

$$\tilde{\mathbf{X}} = \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{I}_d \end{pmatrix}, \quad \tilde{\mathbf{y}} = \begin{pmatrix} \mathbf{y} \\ \mathbf{0}_{d\times 1} \end{pmatrix}$$

$$J(\mathbf{w}) = (\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\mathbf{w})^T(\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda\mathbf{w}^T\mathbf{w}$$

$$\hat{\mathbf{w}}_{ridge} = \tilde{\mathbf{X}} \setminus \tilde{\mathbf{y}}.$$

# SVD

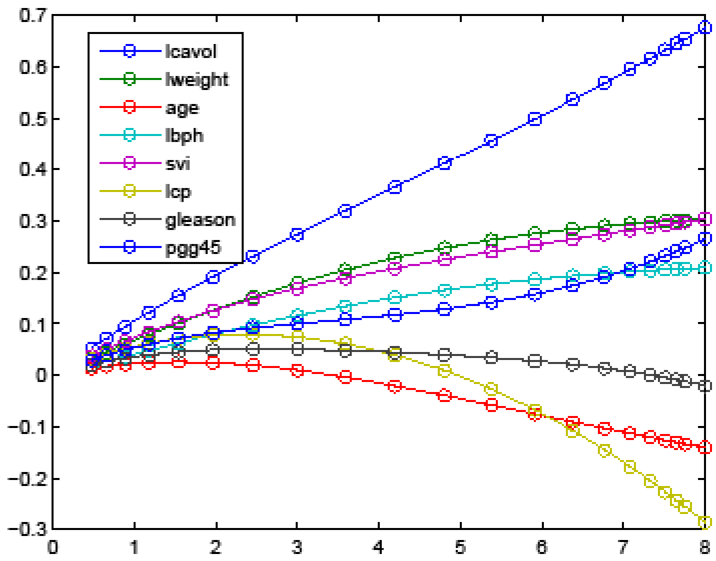
- Recall

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

- Homework: let X=U D $V^T$.

$$\mathbf{w} = \mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{D}\mathbf{U}^T\mathbf{y}$$

- Cheap to compute for many lambdas (regularization path), useful for CV



# Ridge and PCA

- We have

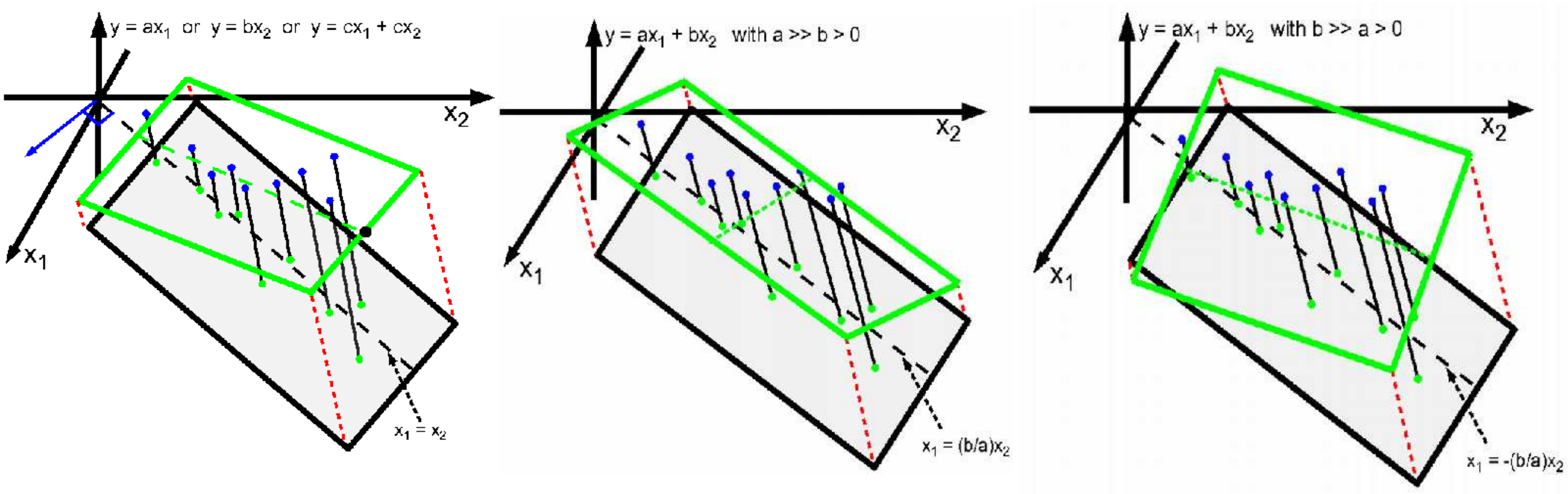
$$
\begin{aligned}
\hat{\mathbf{y}} &= \mathbf{X}\hat{\mathbf{w}}_{ridge} = \mathbf{U}\mathbf{D}\mathbf{V}^T\mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{D}\mathbf{U}^T\mathbf{y} \\
&= \mathbf{U}\tilde{\mathbf{D}}\mathbf{U}^T\mathbf{y} = \sum_{j=1}^{d} \mathbf{u}_j \tilde{D}_{jj} \mathbf{u}_j^T \mathbf{y} \\
\tilde{D}_{jj} &\stackrel{\text{def}}{=} [\mathbf{D}(\mathbf{D}^2 + \lambda I)^{-1}\mathbf{D}]_{jj} = \frac{d_j^2}{d_j^2 + \lambda} \\
\hat{\mathbf{y}} &= \mathbf{X}\hat{\mathbf{w}}_{ridge} = \sum_{j=1}^{d} \mathbf{u}_j \frac{d_j^2}{d_j^2 + \lambda} \mathbf{u}_j^T \mathbf{y} \\
\hat{\mathbf{y}} &= \mathbf{X}\hat{\mathbf{w}}_{ls} = (\mathbf{U}\mathbf{D}\mathbf{V}^T)(\mathbf{V}\mathbf{D}^{-1}\mathbf{U}^T\mathbf{y}) = \mathbf{U}\mathbf{U}^T\mathbf{y} = \sum_{j=1}^{d} \mathbf{u}_j \mathbf{u}_j^T \mathbf{y}
\end{aligned}
$$

$d_j^2/(d_j^2 + \lambda) \leq 1$ Filter factors

# Ridge and PCA

- $D_j^2$ are the eigenvalues of empirical cov mat $X^T X$.
- Small d_j are directions j with small variance: these get shrunk the most, since most ill-determined

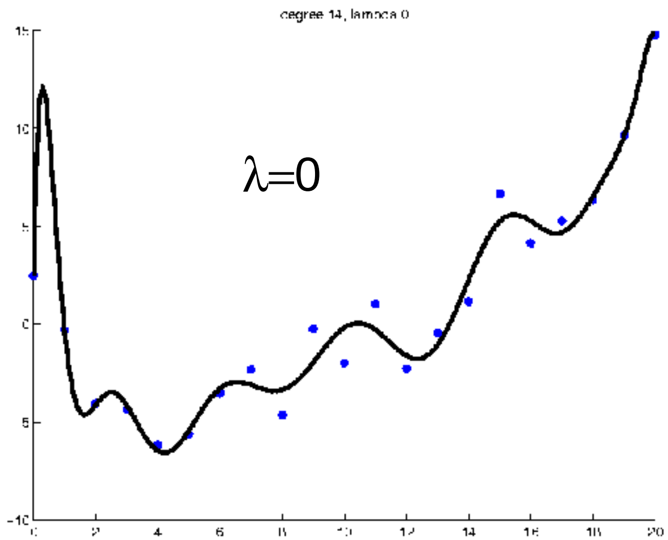
$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}_{ridge} = \sum_{j=1}^{d} \mathbf{u}_j \frac{d_j^2}{d_j^2 + \lambda} \mathbf{u}_j^T \mathbf{y}$$

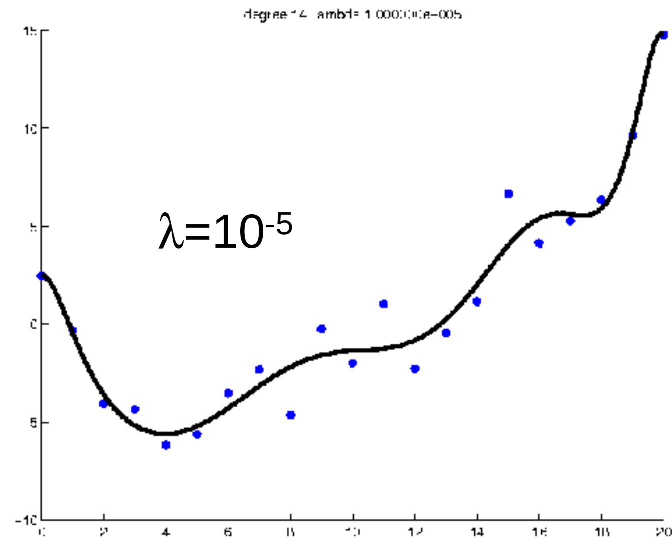


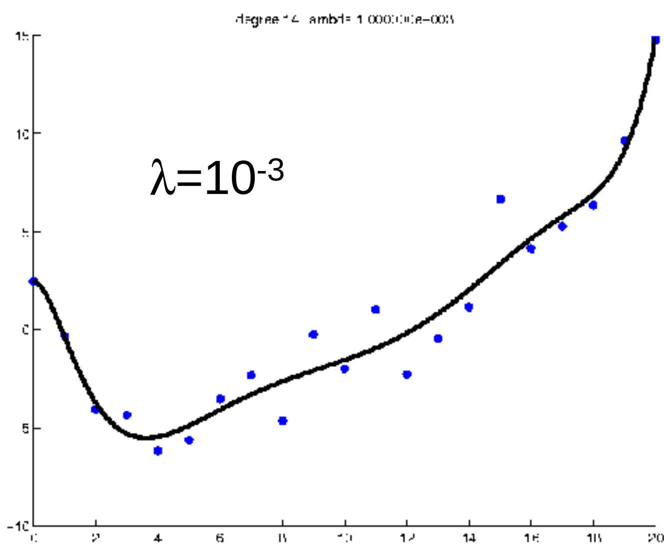
# Principal components regression

- Can set Z=PCA(X,K) then w=regress(X,y) using a pcaTransformer object
- PCR sets (transformed) dimensions K+1,…,d to zero, whereas ridge uses all weighted dimensions. Ridge predictions usually more accurate.
- Feature selection (see later) sets (original) dimensions K+1,…,d to zero. Ridge is usually more accurate, but may be less interpretable.

# Degrees of freedom




All have D=14 but clearly differ in their effective complexity

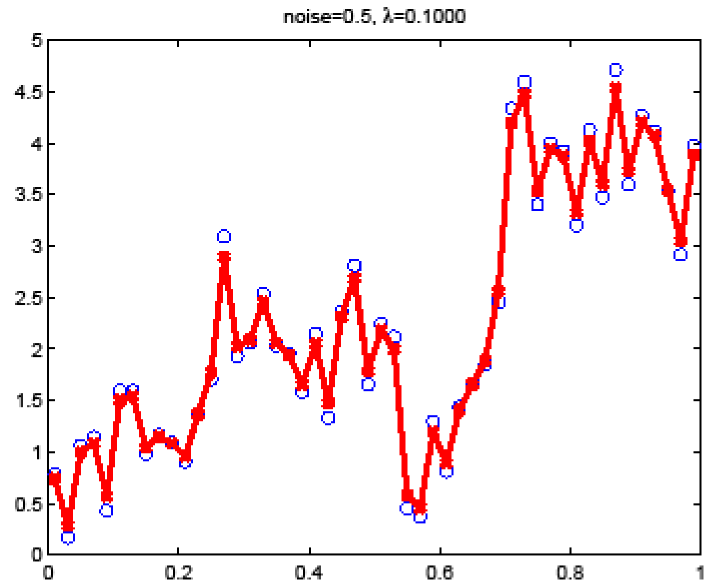
$$\hat{\mathbf{y}} = \mathbf{S}(\mathbf{X})\mathbf{y}$$

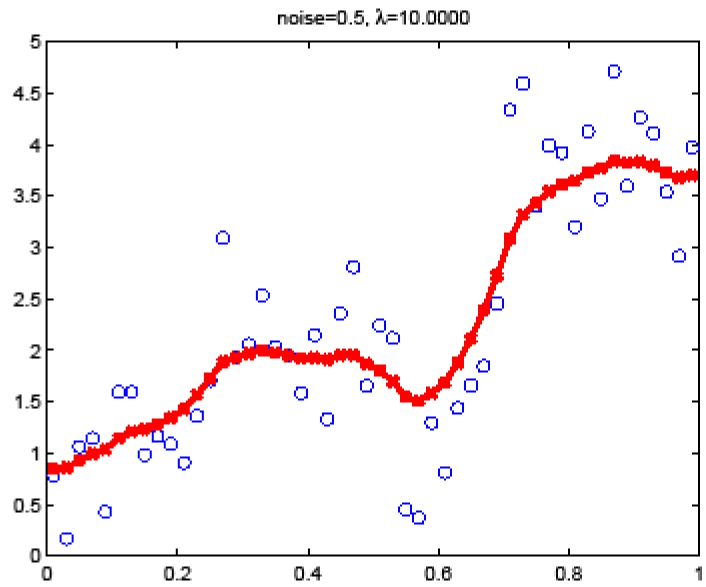
$$df(\mathbf{S}) \stackrel{\text{def}}{=} \text{trace}(\mathbf{S})$$

$$df(\lambda) = \sum_{j=1}^{d} \frac{d_j^2}{d_j^2 + \lambda}$$

# Tikhonov regularization

$$\min_f \frac{1}{2}\int_0^1 (f(x) - y(x))^2 dx + \frac{\lambda}{2}\int_0^1 [f'(x)]^2 dx$$



# Discretization

$$\min_f \frac{1}{2}\int_0^1 (f(x)-y(x))^2 dx + \frac{\lambda}{2}\int_0^1 [f'(x)]^2 dx$$

$$\min_{\mathbf{f}} \frac{1}{2}\sum_{i=1}^{n-1}(f_i-y_i)^2 + \frac{\lambda}{2}\sum_{i=1}^{n-1}(f_{i+1}-f_i)^2$$

$$\min_{\mathbf{f}} \frac{1}{2}\sum_{i=1}^{n}(f_i-y_i)^2 + \frac{\lambda}{4}\sum_{i=1}^{n}\left[(f_i-f_{i-1})^2+(f_i-f_{i+1})^2\right]$$

Boundary conditions: $f_0=f_1$, $f_{n+1}=f_n$

# Matrix form

$$\min_{\mathbf{f}} \frac{1}{2}\sum_{i=1}^{n}(f_i - y_i)^2 + \frac{\lambda}{4}\sum_{i=1}^{n}\left[(f_i - f_{i-1})^2 + (f_i - f_{i+1})^2\right]$$

$$J(\mathbf{w}) = ||\mathbf{y} - \mathbf{w}||^2 + \lambda||\mathbf{D}\mathbf{w}||^2$$

$$\mathbf{D} = \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix}$$

$$||\mathbf{D}\mathbf{w}||^2 = \mathbf{w}^T(D^T D)\mathbf{w} = \sum_{i=1}^{n-1}(w_{i+1} - w_i)^2$$

$$\mathbf{D}^T\mathbf{D} = \begin{pmatrix} 1 & -1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{pmatrix}$$

# QR

$$\min_{\mathbf{w}} \left\| \begin{pmatrix} I_n \\ \sqrt{\lambda} D \end{pmatrix} \mathbf{w} - \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix} \right\|^2$$

*Listing 1:* :

```
D = spdiags(ones(N-1,1)*[-1 1], [0 1], N-1, N);
A = [speye(N); sqrt(lambda)*D];
b = [y; zeros(N-1,1)];
w = A \ b;
```