Calculus, finite differences
Interpolation, Splines, NURBS

CMSC 828 D
Least Squares, SVD, Pseudoinverse

- \( Ax = b \) \( A \) is \( m \times n \), \( x \) is \( n \times 1 \) and \( b \) is \( m \times 1 \).
- \( A = USV^t \) where \( U \) is \( m \times m \), \( S \) is \( m \times n \) and \( V \) is \( n \times n \).
- \( USV^t x = b \). So \( SV^t x = U^t b \).
- If \( A \) has rank \( r \), then \( r \) singular values are significant.
  \[ V^t x = \text{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0) U^t b \]
  \[ x = V \text{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0) U^t b \]
  \[ x_r = \sum_{i=1}^{r} \frac{u_i^t b}{\sigma_i} v_i \quad \sigma_r > \varepsilon, \quad \sigma_{r+1} \leq \varepsilon \]

- Pseudoinverse \( A^+ = V \text{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0) U^t \)
  - \( A^+ \) is a \( n \times m \) matrix.
  - If rank \( (A) = n \) then \( A^+ = (A^t A)^{-1} A \)
  - If \( A \) is square \( A^+ = A^{-1} \)
Well Posed problems

• Hadamard postulated that for a problem to be “well posed”
  1. Solution must exist
  2. It must be unique
  3. Small changes to the input data should cause small changes to the solution

• Many problems in science and computer vision result in “ill-posed” problems.
  – Numerically it is common to have condition 3 violated.

• Recall from the SVD

\[ x = \sum_{i=1}^{n} \frac{u_i^T b}{\sigma_i} v_i \sigma_r > \epsilon, \quad \sigma_{r+1} \leq \epsilon \]

• If \( \sigma \)s are close to zero small changes in the “data” vector \( b \) cause big changes in \( x \).

• Converting ill-posed problem to well-posed one is called \textit{regularization}. 
Regularization

• Pseudoinverse provides one means of regularization

• Another is to solve \((A + \varepsilon I)x = b\)  
  \[
x = \sum_{i=1}^{n} \frac{\sigma_i}{\varepsilon + \sigma_i^2} (u_i^T b) v_i
\]

• Solution of the regular problem requires minimizing of \(\|Ax-b\|^2\)

• This corresponds to minimizing
  \[
  \|Ax-b\|^2 + \varepsilon \|x\|^2
  \]
  – Philosophy – pay a “penalty” of \(O(\varepsilon)\) to ensure solution does not blow up.
  – In practice we may know that the data has an uncertainty of a certain magnitude … so it makes sense to optimize with this constraint.

• Ill-posed problems are also called “ill-conditioned”
Outline

• Gradients/derivatives
  – needed in detecting features in images
    • Derivatives are large where changes occur
  – essential for optimization

• Interpolation
  – Calculating values of a function at a given point based on known values at other points
  – Determine error of approximation
  – Polynomials, splines

• Multiple dimensions
Derivative

• In 1-D
\[ \frac{df}{dx} = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \]

• Taylor series: for a continuous function
\[
\begin{align*}
f(x + h) &= f(x) + h \frac{df}{dx} \bigg|_x + \frac{h^2}{2} \frac{d^2 f}{dx^2} \bigg|_x + \cdots + \frac{h^n}{n!} \frac{d^n f}{dx^n} \bigg|_x + \cdots \\
f(x - h) &= f(x) - h \frac{df}{dx} \bigg|_x + \frac{h^2}{2} \frac{d^2 f}{dx^2} \bigg|_x + \cdots + (-1)^n \frac{h^n}{n!} \frac{d^n f}{dx^n} \bigg|_x + \cdots
\end{align*}
\]

• Geometric interpretation
  – Approximate smooth curve by values of tangent, curvature, etc.
Remarks

• Mean value theorem:
  – $f(b)-f(a)=(b-a)\frac{df}{dx}/c \quad a<c<b$
  – There is at least one point between $a$ and $b$ on the curve where the slope matches that of the straight line joining the two points

• $df/dx=0$
  – represents a minimum, maximum or saddle point of the curve $y=f(x)$
  – $d^2f/dx^2 > 0$ minimum, $d^2f/dx^2 < 0$ maximum
  – $d^2f/dx^2 = 0$ saddle point
Finite differences

• Approximate derivatives at points by using values of a function known at certain neighboring points

• Truncate Taylor series and obtain an expression for the derivatives

• Forward differences: use value at the point and forward

\[
\frac{df}{dx}
\]

\[=
\]

\[ \frac{h^{-1} (f(x + h) - f(x)) - \frac{h}{2} \frac{d^2 f}{dx^2}}{h} + O(h^2) \]

• Backward differences

\[
\frac{df}{dx}
\]

\[=
\]

\[ \frac{h^{-1} (f(x) - f(x - h)) + \frac{h}{2} \frac{d^2 f}{dx^2}}{h} + O(h^2) \]
Finite Differences

• Central differences
  – Higher order approximation

\[
2 \frac{df}{dx} \bigg|_x = \frac{f(x + h) - f(x)}{h} - \frac{h}{2} \frac{d^2 f}{dx^2} \bigg|_x + \frac{f(x) - f(x - h)}{h} + \frac{h}{2} \frac{d^2 f}{dx^2} \bigg|_x + O(h^2)
\]

\[
\frac{df}{dx} \bigg|_x = \frac{f(x + h) - f(x - h)}{2h} + O(h^2)
\]

– However we need data on both sides
– Not possible for data on the edge of an image
– Not possible in time dependent problems (we have data at current time and previous one)
Approximation

- Order of the approximation $O(h), O(h^2)$
- Sidedness, one sided, central etc.
- Points around point where derivative is calculated that are involved are called the “stencil” of the approximation.
- Second derivative

\[
0 = \frac{f(x + h) - f(x)}{h} - \frac{h}{2} \frac{d^2 f}{dx^2} \left|_x \right. - \frac{f(x) - f(x - h)}{h} + \frac{h}{2} \frac{d^2 f}{dx^2} \left|_x \right. + O(h^2)
\]

\[
\frac{d^2 f}{dx^2} = \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} + O(h)
\]

- One sided difference of $O(h^2)$

\[
\frac{df}{dx} = \frac{-3f(x) + 4f(x + h) - f(x + 2h)}{2h} + O(h^2)
\]
Polynomial interpolation

• Instead of playing with Taylor series we can obtain fits using polynomial expansions.
  – 3 points fit a quadratic $ax^2 + bx + c$
    • Can calculate the 1st and 2nd derivatives
  – 4 points fit a cubic, etc.

• Given $x_1, x_2, x_3, x_4$ and values $f_1, f_2, f_3, f_4$

\[
\begin{bmatrix}
1 & x_1 & x_1^2 & x_1^3 \\
1 & x_2 & x_2^2 & x_2^3 \\
1 & x_3 & x_3^2 & x_3^3 \\
1 & x_4 & x_4^2 & x_4^3 \\
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
\end{bmatrix}
=
\begin{bmatrix}
f_1 \\
f_2 \\
f_3 \\
f_4 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
\end{bmatrix}
=
\begin{bmatrix}
1 & x_1 & x_1^2 & x_1^3 \\
1 & x_2 & x_2^2 & x_2^3 \\
1 & x_3 & x_3^2 & x_3^3 \\
1 & x_4 & x_4^2 & x_4^3 \\
\end{bmatrix}^{-1}
\begin{bmatrix}
f_1 \\
f_2 \\
f_3 \\
f_4 \\
\end{bmatrix}
\]

• Vandermonde system – fast algorithms for solution.
• If more data than degree .. Can get a least squares solution.
• Matlab functions polyfit, polyval
Remarks

• Can use the fitted polynomial to calculate derivatives
• If equation is solved analytically this provides expressions for the derivatives.
• Equation can become quite ill conditioned
  – especially if equations are not normalized.
  \[ ax^2 + bx + c \] can also be written as \[ a^* (x-x_0)^2 + b^* (x-x_0) + c^* \]
  – Find the polynomial through \( x_0-h, x_0, x_0+h \)

\[
\begin{bmatrix}
1 & -h & h^2 \\
1 & 0 & 0 \\
1 & h & h^2
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2
\end{bmatrix}
= 
\begin{bmatrix}
f_{-1} \\
f_0 \\
f_1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 1 & 0 \\
-\frac{1}{2h} & 0 & \frac{1}{2h} \\
\frac{1}{2h^2} & -\frac{1}{h^2} & \frac{1}{2h^2}
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2
\end{bmatrix}
= 
\begin{bmatrix}
f_{-1} \\
f_0 \\
f_1
\end{bmatrix}
\]

\[-a_0 = f_0, \quad a_1 = \frac{(f_1-f_{-1})}{2h}, \quad a_2 = \frac{(f_{-1}-2f_0+f_1)}{2h^2}\]

– Gives the expected values of the derivatives.
Polynomial interpolation

• Results from Algebra
  – Polynomial of degree \( n \) through \( n+1 \) points is unique
  – Polynomials of degree less than \( x^n \) is an \( n \) dimensional space.
  – \( 1, x, x^2, \ldots, x^{n-1} \) form a basis.
    • Any other polynomial can be represented as a combination of these basis elements.
  – Other sets of independent polynomials can also form bases.

• To fit a polynomial through \( x_0, \ldots, x_n \) with values \( f_0, \ldots, f_n \)
  – Use Lagrangian basis \( l_k \).
    \[ l_k = \prod_{i=0, i \neq k}^{n} \frac{x-x_i}{x_k-x_i}, \quad k = 0, \ldots, n \]
    \[ p(x) = a_0 l_0 + a_1 l_1 + \ldots + a_n l_n. \]
  – Then \( a_i = f_i \)
  – Many polynomial bases: Chebyshev, Legendre, Laguerre …
  – Bernstein, Bookstein …
Increasing $n$

- As $n$ increases we can increase the polynomial degree.
- However the function in between is very poorly interpolated.
- Becomes ill-posed.
- For large $n$ interpolant blows up.

- Idea:
  - Taylor series provides good local approximations
  - Use local approximations

- Splines
Spline interpolation

- Piecewise polynomial approximation
  - E.g. interpolation in a table
  - Given \( x_k, x_{k+1}, f_k \) and \( f_{k+1} \) evaluate \( f \) at a point \( x \) such that \( x_k < x < x_{k+1} \)
    \[
    f(x) = \begin{cases} 
    f_{k+1} \frac{x - x_k}{x_{k+1} - x_k} + f_k \frac{x - x_{k+1}}{x_k - x_{k+1}}, & x_k \leq x \leq x_{k+1} \\
    0, & \text{otherwise} 
    \end{cases}
    \]

- Construct approximations of this type on each subinterval
  - This method uses Lagrangian interpolants

- Endpoints are called *breakpoints*

- For higher polynomial degree we need more conditions
  - e.g. specify values at points inside the interval \([x_k < x < x_{k+1}]\)
  - Specifying function and derivative values at the end points \( x_k, x_{k+1} \) leads to cubic Hermite interpolation
Cubic Spline

- Splines – name given to a flexible piece of wood used by draftsmen to draw curves through points.
  - Bend wood piece so that it passes through known points and draw a line through it.
  - Most commonly used interpolant used is the cubic spline
  - Provides continuity of the function, 1st and 2nd derivatives at the breakpoints.
  - Given n+1 points we have n intervals \( \{x_i, f_i\}, \ i = 1, \ldots, n + 1 \)
  - Each polynomial has four unknown coefficients
    - Specifying function values provides 2 equations
    - Two derivative continuity equations provides two more

\[
P_i(x) = f_i \quad i = 1, \ldots, n + 1
\]

\[
P_i''(x) = P_i''(x) \quad i = 2, \ldots, n
\]

\[
P_{i-1}'(x) = P_i'(x) \quad i = 2, \ldots, n
\]

- Left with two free conditions. Usually chosen so that second derivatives are zero at ends
Interpolating along a curve

- Curve can be given as \( x(s) \) and \( y(s) \)
- Given \( x_i, y_i, s_i \)
- Can fit splines for \( x \) and \( y \)
- Can compute tangents, curvature and normal based on this fit
- Things like intensity can vary along the curve. Can also fit \( I(s) \)
Two and more dimensions

- **Gradient**
  \[ \nabla f = \frac{\partial f}{\partial x} e_1 + \frac{\partial f}{\partial y} e_2 = \frac{\partial f}{\partial x_1} e_i \]

- **Directional derivative in**
  \[ \nabla f \cdot n = \frac{\partial f}{\partial x} e_1 \cdot n + \frac{\partial f}{\partial y} e_2 \cdot n = \frac{\partial f}{\partial x_1} n_i \]
  the direction of a vector **n**

**Geometric interpretation**

- \( \nabla f \) is normal to the surface \( f(x) = c \)
- \( n = \nabla f / |\nabla f| \)

**Taylor series**

\[
\begin{align*}
f(x + h) &= f(x) + h \cdot \nabla f(x) + \frac{1}{2} (hh) : \nabla \nabla f(x) + O(|h|^3) \\
f(x + h) &= f(x) + h_i \frac{\partial f}{\partial x_i} + \frac{1}{2} h_i h_j \frac{\partial}{\partial x_i} \frac{\partial f}{\partial x_j} + O(|h|^3)
\end{align*}
\]
Finite differences

- Follows a similar pattern. One dimensional partial derivatives are calculated the same way.
- Multiple dimensional operators are computed using multidimensional stencils.

\[ \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \frac{f_{i+1,j} + f_{i,j+1} - 4f_{i,j} + f_{i,j-1} + f_{i-1,j}}{h^2} \]
Interpolation

- Polynomial interpolation in multiple dimensions
- Pascals triangle
- Least squares
- Move to a local coordinate system
Tensor product splines

- Splines form a local basis.
- Take products of one dimensional basis functions to make a basis in the higher dimension.
NURBS

- Used for precisely specifying n-d data.
- October 3 Tapas Kanungo, NURBS: Non-Uniform Rational B-Splines
Derivative of a matrix

Suppose $f(\mathbf{x})$ is a scalar-valued function of $d$ variables $x_i$, $i = 1, 2, \ldots d$, which we represent as the vector $\mathbf{x}$. Then the derivative or gradient of $f$ with respect to this vector is computed component by component, i.e.,

$$
\nabla f(\mathbf{x}) = \text{grad} f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = 
\begin{pmatrix}
\frac{\partial f(\mathbf{x})}{\partial x_1} \\
\frac{\partial f(\mathbf{x})}{\partial x_2} \\
\vdots \\
\frac{\partial f(\mathbf{x})}{\partial x_d}
\end{pmatrix}. \tag{12}
$$

If we have an $n$-dimensional vector-valued function $\mathbf{f}$ (note the use of boldface), of a $d$-dimensional vector $\mathbf{x}$, we calculate the derivatives and represent them as the Jacobian matrix

$$
\mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = 
\begin{pmatrix}
\frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_d} \\
\vdots & \ddots & \vdots \\
\frac{\partial f_n(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_d}
\end{pmatrix}. \tag{13}
$$

If this matrix is square, its determinant (Sect. A.2.5) is called simply the Jacobian or occasionally the Jacobian determinant.
Jacobian and Hessian

We first recall the use of second derivatives of a scalar function of a scalar $x$ in writing a Taylor series (or Taylor expansion) about a point:

$$f(x) = f(x_0) + \left. \frac{df(x)}{dx} \right|_{x=x_0} (x - x_0) + \frac{1}{2!} \left. \frac{d^2f(x)}{dx^2} \right|_{x=x_0} (x - x_0)^2 + O((x - x_0)^3). \quad (20)$$

Analogously, if our scalar-valued $f$ is a instead a function of a vector $\mathbf{x}$, we can expand $f(\mathbf{x})$ in a Taylor series around a point $\mathbf{x}_0$:

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \left[ \frac{\partial f}{\partial \mathbf{x}} \right]_{\mathbf{x}=\mathbf{x}_0}^t (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2!} (\mathbf{x} - \mathbf{x}_0)^t \left[ \frac{\partial^2f}{\partial \mathbf{x}^2} \right]_{\mathbf{x}=\mathbf{x}_0}^t (\mathbf{x} - \mathbf{x}_0) + O(||\mathbf{x} - \mathbf{x}_0||^3), \quad (21)$$

where $\mathbf{H}$ is the Hessian matrix, the matrix of second-order derivatives of $f(\cdot)$, here