Announcements

Matlab Grader homework, emailed Thursday,

1 and 2 (of less than 9) homeworks Due 21 April, Binary graded.

Jupyter homework?: translate matlab to Jupiter, TA Harshul h6gupta@eng.ucsd.edu or me

I would like this to happen.

“GPU” homework. NOAA climate data in Jupyter on the datahub.ucsd.edu, released 17 April.

Projects: Any computer language. Access to Jupyterhub with GPU

Podcast might work eventually.

Today:

- Stanford CNN
- Gaussian processes for concert hall
- Linear models for regression

Wednesday 10 April

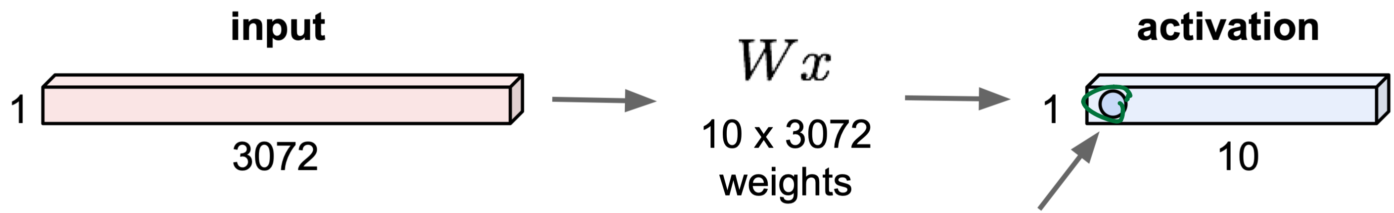
Stanford CNN, Linear models for regression/classification (Bishop 3 and 4), CNN

FNN

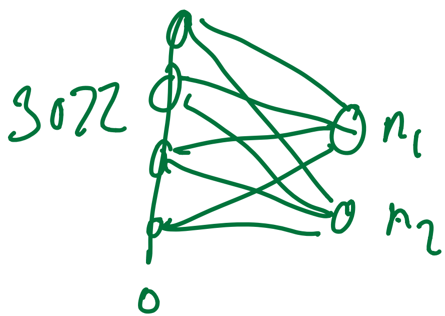
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

Each neuron looks at the full input volume



1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

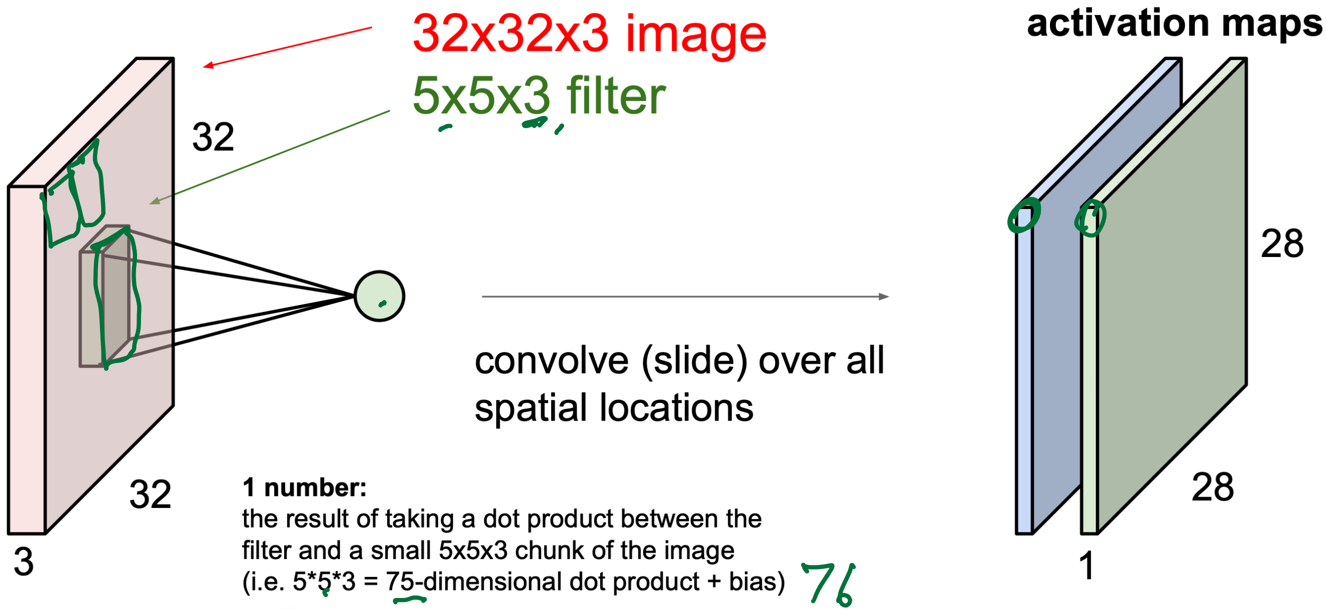


$W^T X$

CNN

Convolution Layer

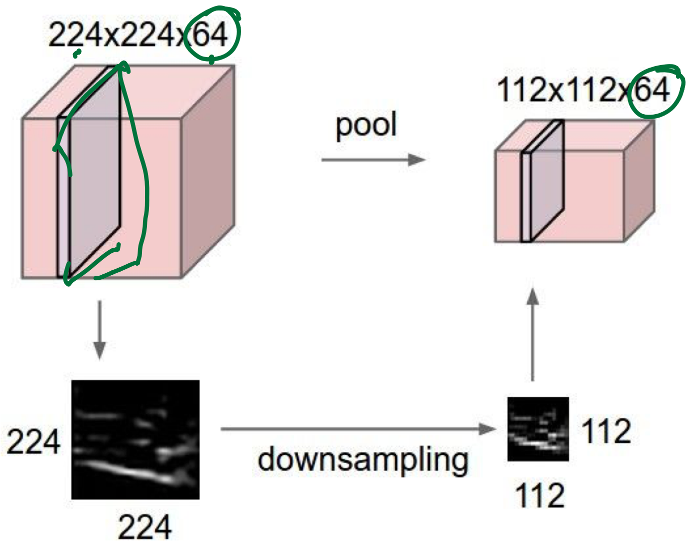
consider a second, green filter



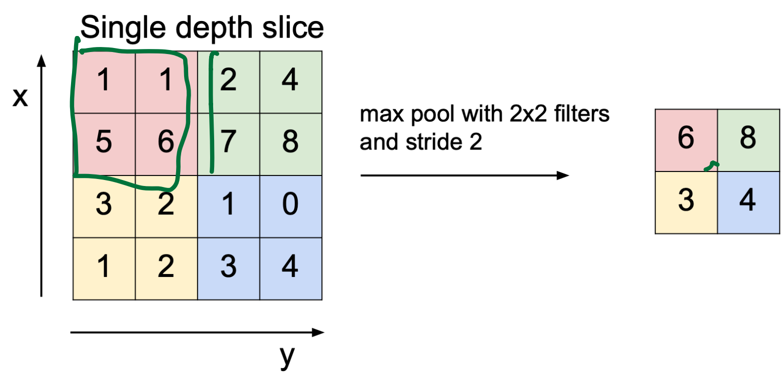
$w^T x + b$

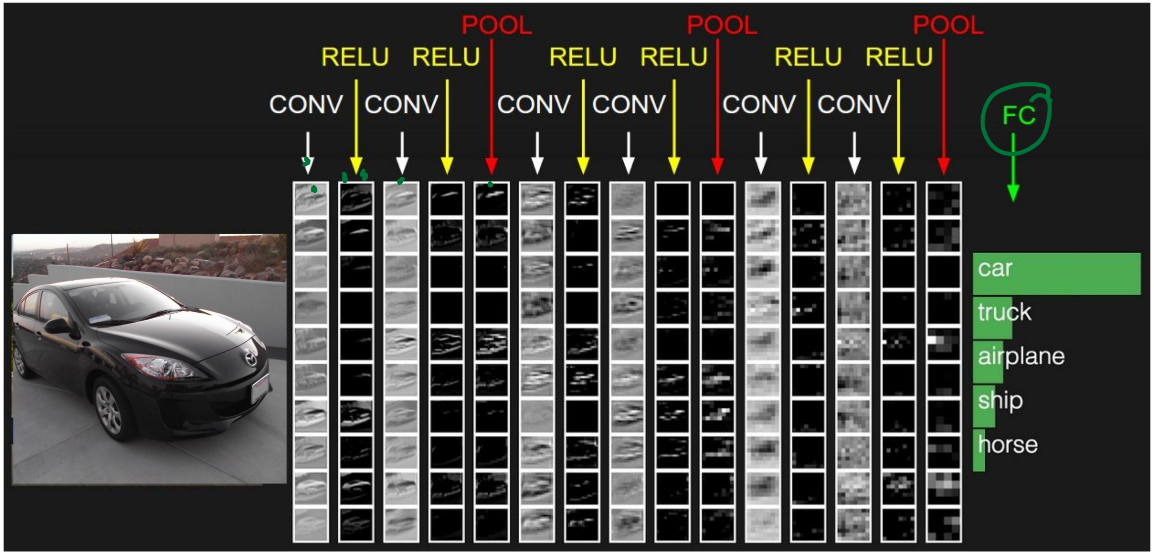
Pooling layer

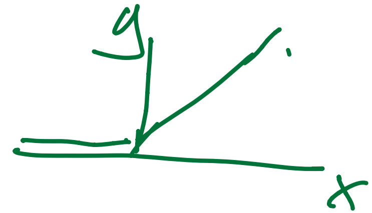
- makes the representations smaller and more manageable
- operates over each activation map independently:



MAX POOLING







$y = \max(0, x)$

Summary

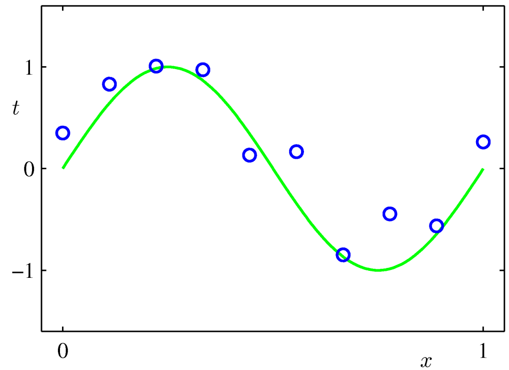
- ConvNets stack CONV,POOL,FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like
  [(CONV-RELU)*N-POOL?]*M-(FC-RELU)*K,SOFTMAX
  where N is usually up to ~5, M is large, 0 <= K <= 2.
  - but recent advances such as ResNet/GoogLeNet challenge this paradigm

Linear regression: Linear Basis Function Models (1)

Generally

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

- where $\phi_j(x)$ are known as *basis functions*.
- Typically, $\phi_0(x) = 1$, so that w_0 acts as a bias.
- Simplest case is linear basis functions: $\phi_d(x) = x_d$.

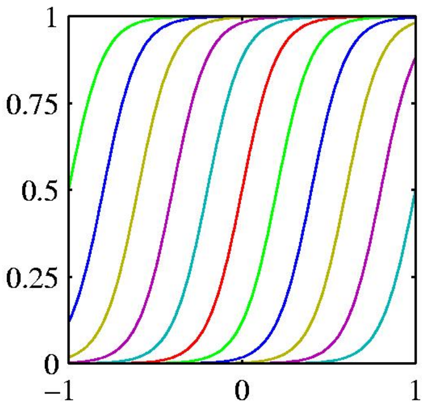


$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

http://playground.tensorflow.org/

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \quad \phi = \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}$$

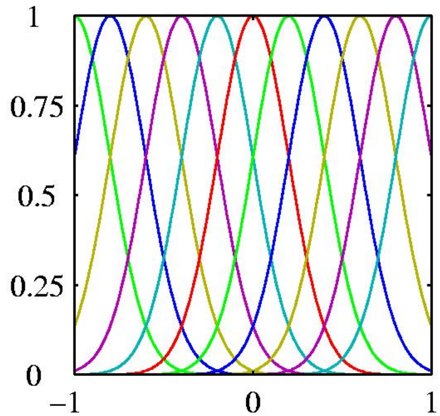
Some types of basis function in 1-D



Sigmoids

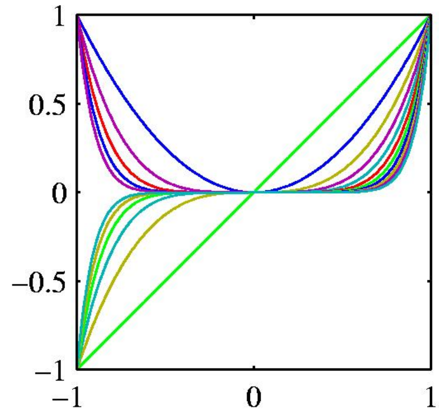
$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$$

$$\sigma(a) = \frac{1}{1+\exp(-a)}.$$



Gaussians

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$



Polynomials

$$\phi_j(x) = x^j.$$

Sigmoid and Gaussian basis functions can also be used in multilayer neural networks, but neural networks learn the parameters of the basis functions. This is more powerful but also harder and messier.

Two types of linear model that are equivalent with respect to learning

bias

$$y(\mathbf{x},\mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots = \mathbf{w}^T \mathbf{x}$$

$$y(\mathbf{x},\mathbf{w}) = w_0 + w_1 \phi_1(\mathbf{x}) + w_2 \phi_2(\mathbf{x}) + \ldots = \underline{\mathbf{w}^T \Phi(\mathbf{x})}$$

- The first and second model has the same number of adaptive coefficients as the number of basis functions +1.
- Once we have replaced the data by basis functions outputs, fitting the second model is exactly the same the first model.
  - No need to clutter math with basis functions

Maximum Likelihood and Least Squares (1)

- Assume observations from a deterministic function with added Gaussian noise:

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \quad \text{where} \quad p(\epsilon|\beta) = \mathcal{N}(\epsilon|0, \beta^{-1})$$

- or,

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}).$$

- Given observed inputs, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, and targets $\mathbf{t} = [t_1, \ldots, t_N]^{\mathrm{T}}$, we obtain the likelihood function

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}).$$

Maximum Likelihood and Least Squares (2)

Taking the logarithm, we get

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{w},\beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n),\beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta \underline{E_D(\mathbf{w})}\end{aligned}$$

Where the sum-of-squares error is

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

$$\frac{\partial E}{\partial w} = \sum_{n=1}^{N} \phi_n^T \{t_n - w^T \phi_n\}$$

$$\underline{\underline{\Phi}} = \begin{array}{c} \xrightarrow{\quad M \quad} \\ \begin{bmatrix} - & \phi^T(x_1) & - \\ - & \phi^T(x_2) & - \\ & \downarrow & \end{bmatrix} N \end{array}$$

$$\underline{\underline{\Phi}}^T \underline{t} - \underline{\underline{\Phi}}^T \underline{\underline{\Phi}}\, \underline{w} = 0$$

$$\underline{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \end{bmatrix} N$$

$$w_{ML} = (\underline{\underline{\Phi}}^T \underline{\underline{\Phi}})^{-1} \Phi^T t$$

Maximum Likelihood and Least Squares (3)

Computing the gradient and setting it to zero yields

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{w},\beta) = \beta \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right\} \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} = \mathbf{0}.$$

Solving for w,

$$\mathbf{w}_{\mathrm{ML}} = \overbrace{\left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}}}^{\text{The Moore-Penrose pseudo-inverse, } \boldsymbol{\Phi}^{\dagger}.} \mathbf{t}$$

where

$$\boldsymbol{\Phi} = \begin{array}{cc} M & \\ \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}. & N \end{array}$$

Maximum Likelihood and Least Squares (4)

Maximizing with respect to the bias, w_0, alone,

$$w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j \overline{\phi_j}$$

$$= \underbrace{\frac{1}{N}\sum_{n=1}^{N} t_n}_{\bar{t}} - \sum_{j=1}^{M-1} w_j \underbrace{\frac{1}{N}\sum_{n=1}^{N} \phi_j(\mathbf{x}_n)}_{\overline{\phi_j}}.$$

We can also maximize with respect to β, giving

$$\frac{\partial p}{\partial \beta} = 0$$

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N}\sum_{n=1}^{N}\{t_n - \mathbf{w}_{\mathrm{ML}}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

Geometry of Least Squares

$N \begin{bmatrix} \varphi_1 & \varphi_2 & \cdots \end{bmatrix} = \begin{bmatrix} \phi(x_1) \\ \phi(x_2) \\ \vdots \end{bmatrix} N$

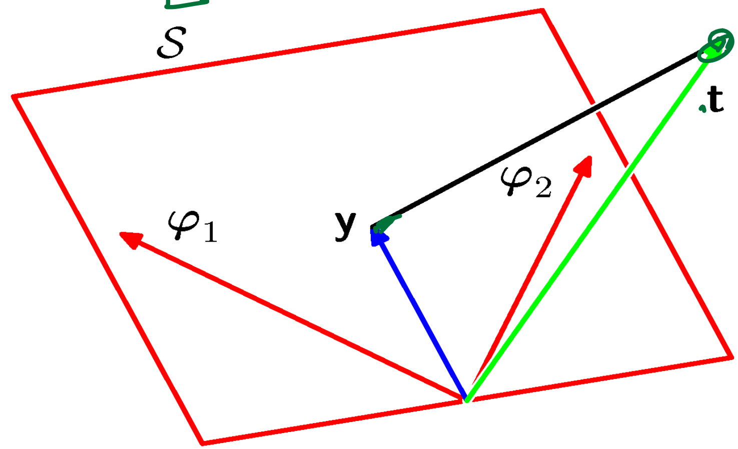
Consider

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{w}_{\mathrm{ML}} = [\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_M]\,\mathbf{w}_{\mathrm{ML}}.$$

$$\mathbf{y} \in \mathcal{S} \subseteq \mathcal{T} \qquad \mathbf{t} \in \mathcal{T}$$

N-dimensional

M-dimensional



S is spanned by

$$\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_M$$

w_{ML} minimizes the distance between t and its orthogonal projection on S, i.e. y.

$\varphi_1 = \begin{bmatrix} \phi(x_1) \\ \phi(x_2) \\ \vdots \end{bmatrix} N$

Least mean squares: An alternative approach for big datasets

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \boxed{\nabla E_{n(\tau)}}$$

weights after seeing training case tau+1

learning rate

squared error derivatives w.r.t. the weights on the training case at time tau.

This is **"on-line" learning**. It is efficient if the dataset is redundant and simple to implement.

- It is called **stochastic gradient descent** if the training cases are picked randomly.
- Care must be taken with the learning rate to prevent divergent oscillations. Rate must decrease with tau to get a good fit.

$$\frac{\partial E}{\partial w} = \cancel{\sum_{n=1}^{N}} \phi_n (t_n - w^T \phi_n) \leftarrow$$

Regularized least squares

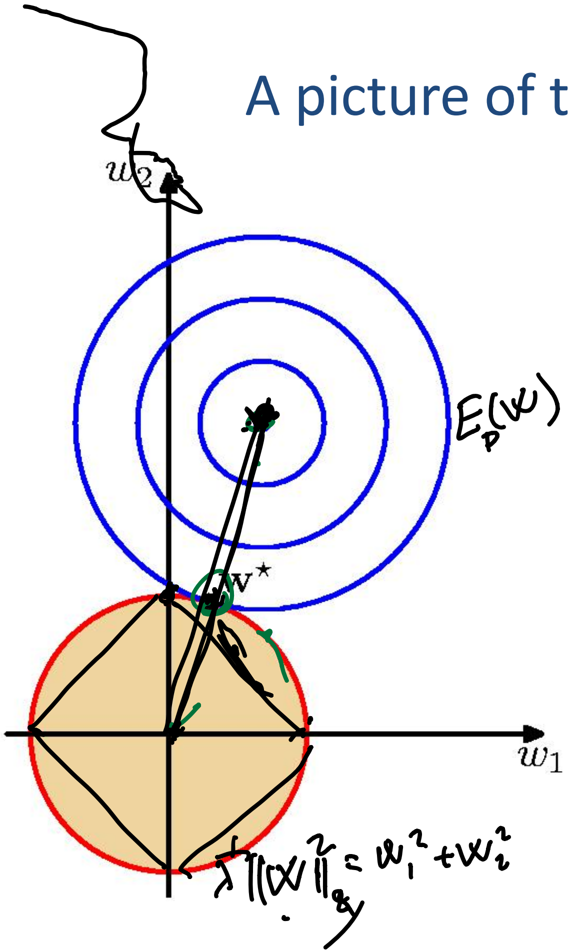
$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

The squared weights penalty is mathematically compatible with the squared error function, giving a closed form for the optimal weights:

$$\underline{\mathbf{w}^*} = (\lambda\,\mathbf{I} + \mathbf{X}^T\mathbf{X})^{-1}\,\mathbf{X}^T\,\mathbf{t}$$

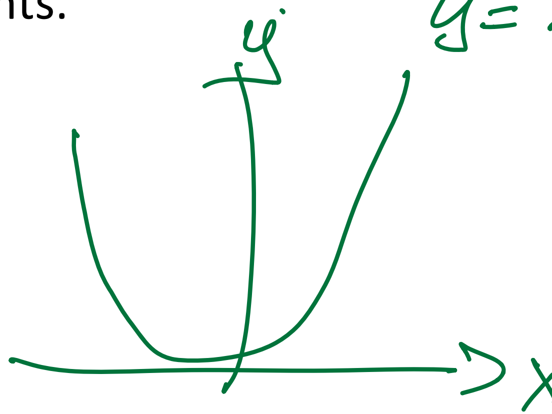
identity matrix

A picture of the effect of the regularizer



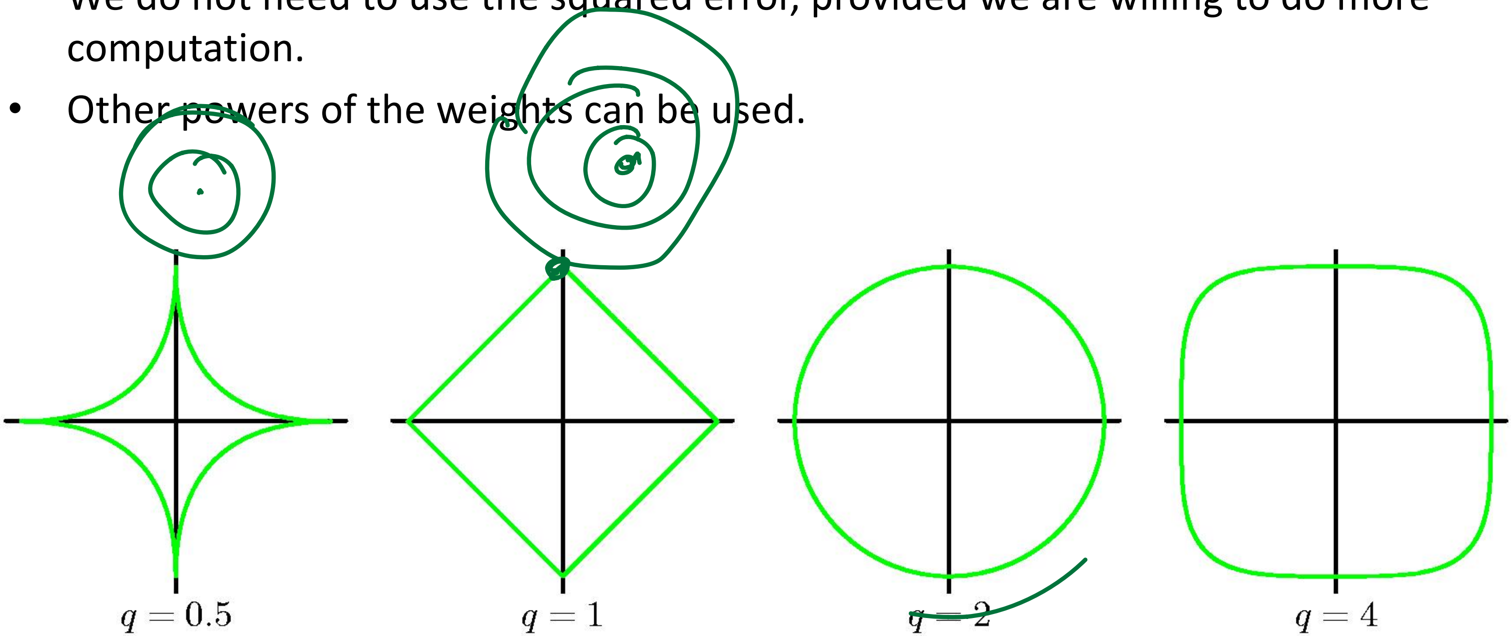
- The overall cost function is the sum of two parabolic bowls.
- The sum is also a parabolic bowl.
- The combined minimum lies on the line between the minimum of the squared error and the origin.
- The L2 regularizer just **shrinks** the weights.

$y = x^2$



Other regularizers

- We do not need to use the squared error, provided we are willing to do more computation.
- Other powers of the weights can be used.



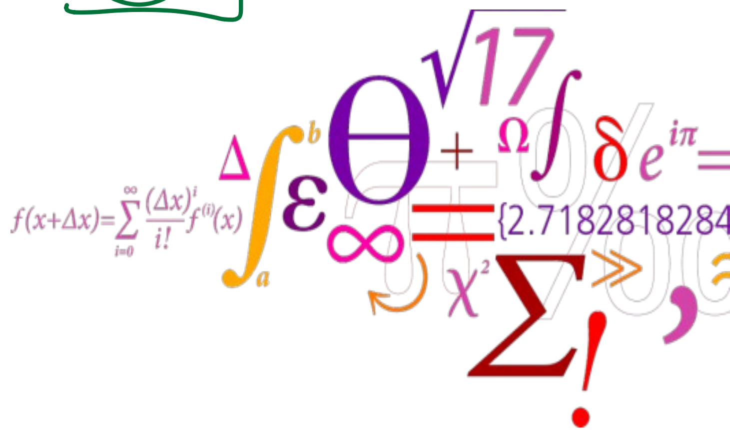
Transfer function reconstruction for outdoor sound field control

Diego Caviedes Nozal

27/03/2019

$$x(t) \longrightarrow \boxed{h(t)} \longrightarrow y(t)$$

$$y(t) = x(t) * h(t)$$



DTU Electrical Engineering
Department of Electrical Engineering

Related to the course

Linear models for regression (Lecture 4)

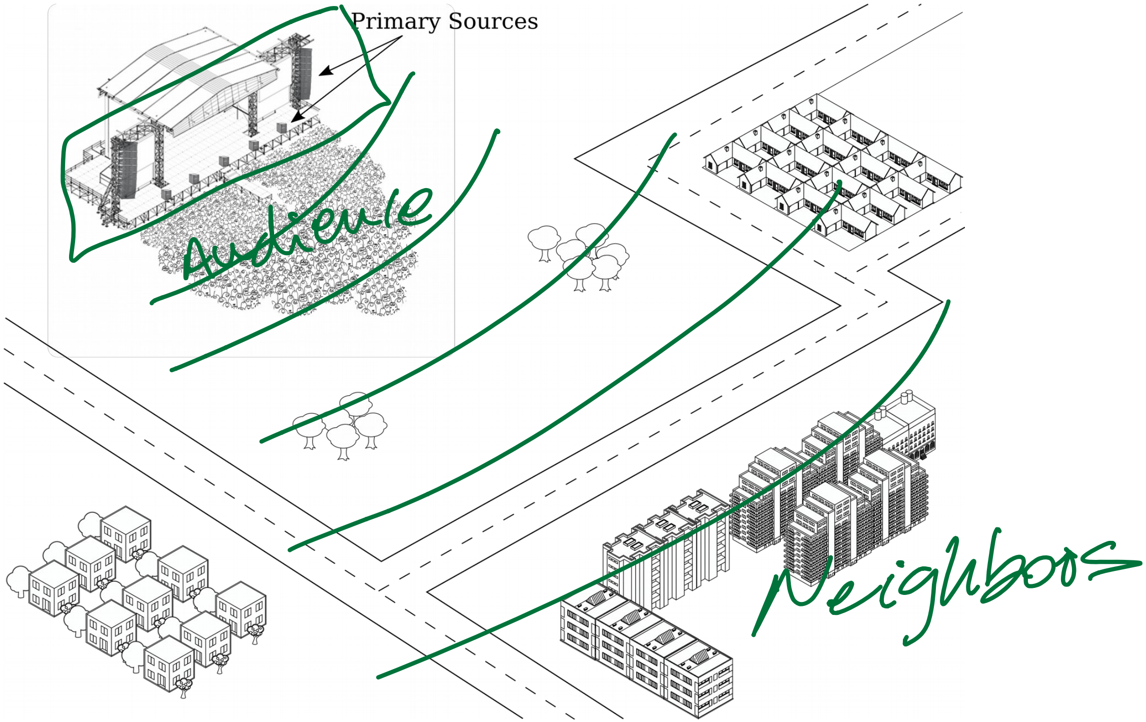
$y = Ax$

Non-linear models for regression

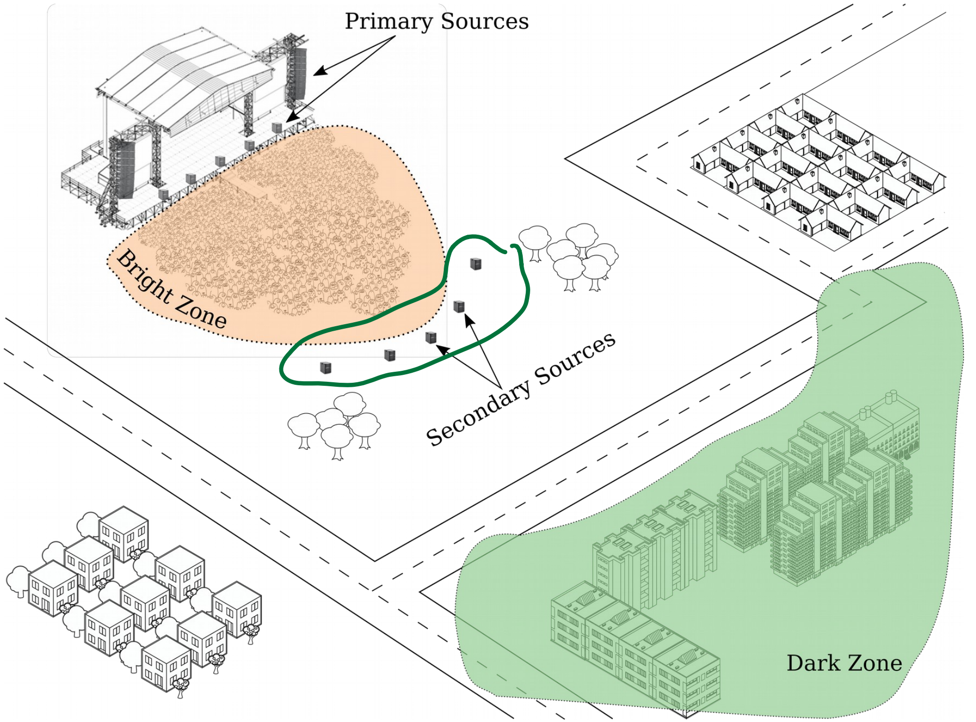
Gaussian Processes (Lecture 3)

Bayes rule (Lecture 1)

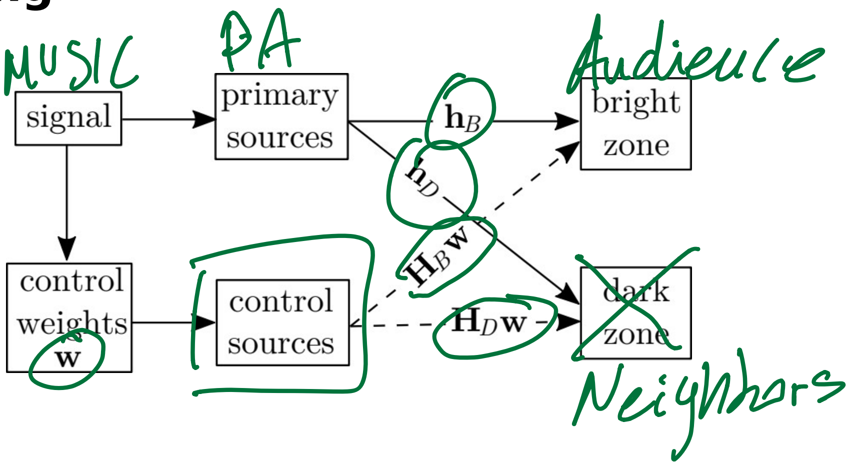
The problem



Our goal



Sound zoning



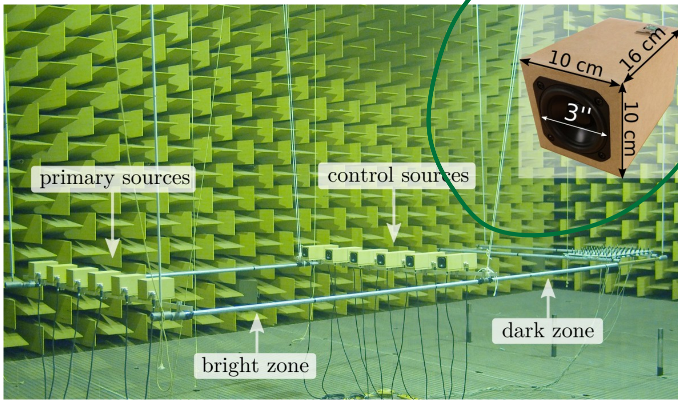
Objectives

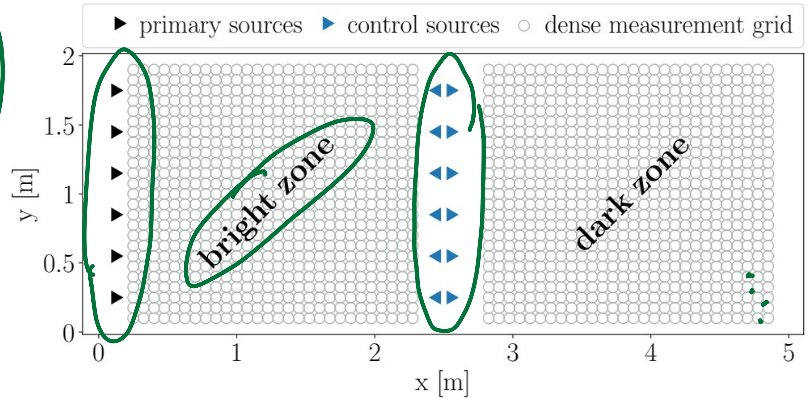
1) Cancellation of sound from the primary sources in a *dark zone* using a set of secondary control sources.

2) Minimization of the sound radiated by the control sources into the *bright zone*.

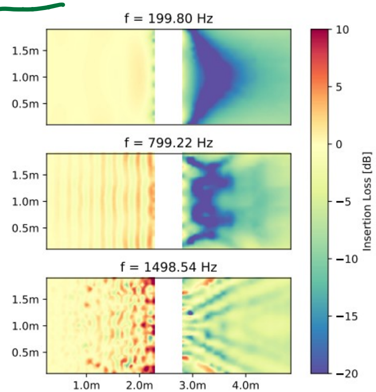
$$\underset{\mathbf{w}}{\text{minimize}} \quad \overset{2)}{\kappa||\mathbf{H}_B\mathbf{w}||_2^2} + \underbrace{(1-\kappa)\overset{1)}{||\mathbf{H}_D\mathbf{w} + \mathbf{h}_D||_2^2}} \qquad \kappa \in [0,1]$$

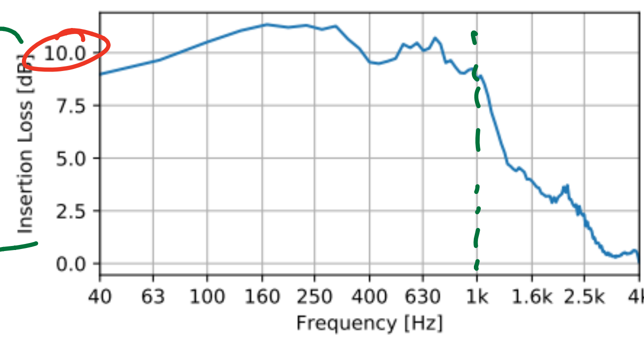
Real experiments: Anechoic conditions



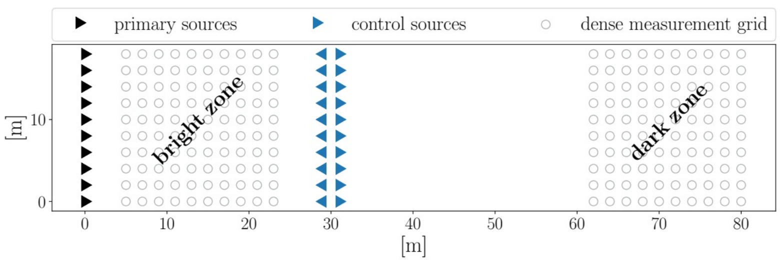


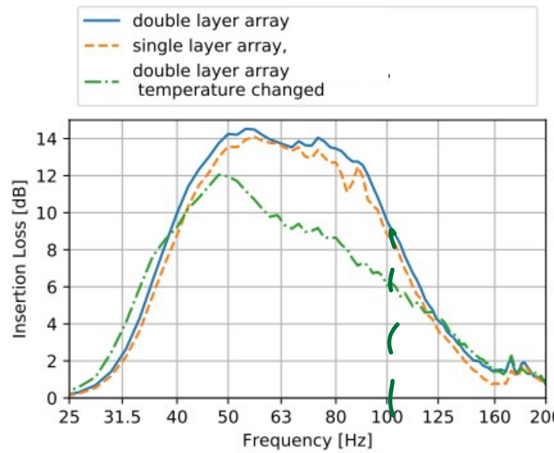
$$IL = 10\log_{10}\left(\frac{\|h_D\|^2}{\|H_D w + h_D\|^2}\right)$$





Real experiments: Outdoor conditions





Measuring transfer functions: Too many issues

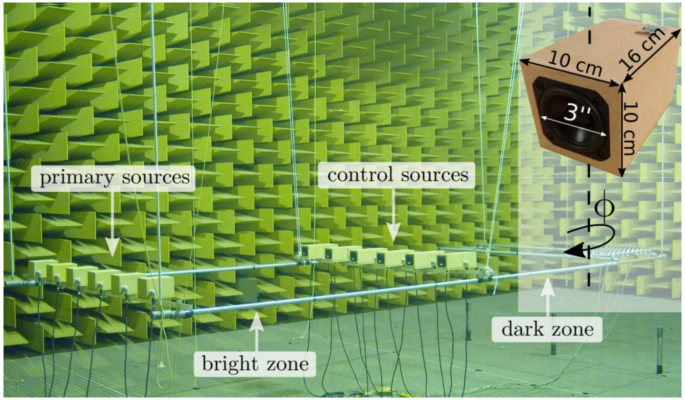
- Measuring hundreds of transfer functions to sample the control zones is not possible in real open air concerts.

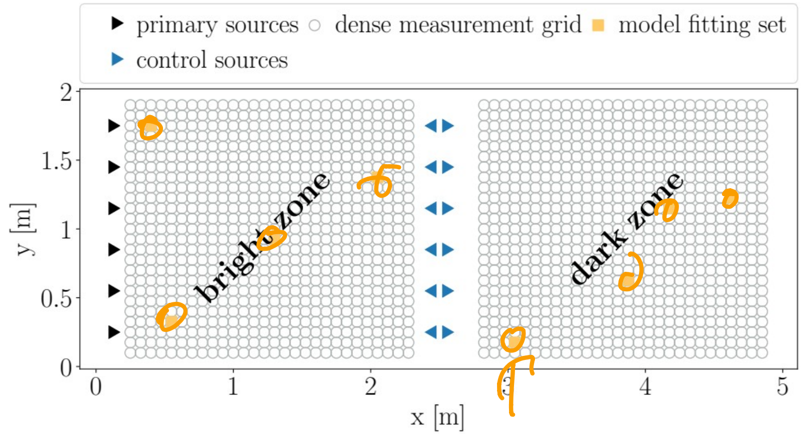
- The acoustic transfer functions must be representative of the conditions that the sound field control is applied.

Different approach: **Sound propagation models to estimate the transfer functions.**

  - Use sparse measurements to fit the model.

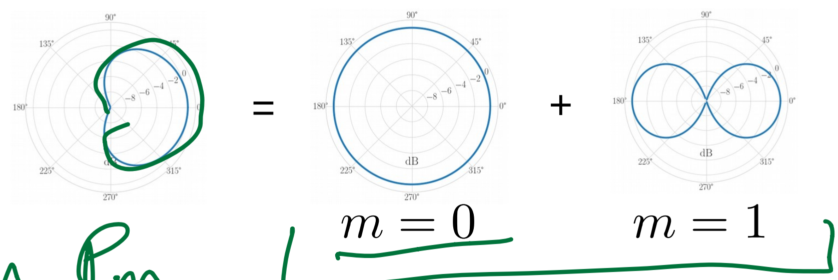
Modeling: Anechoic conditions





Source model: spherical harmonics

$$\hat{h}(k, \mathbf{r}) = \sum_{m=0}^{M-1} \boxed{a_m} \mathrm{h}_m^{(2)}(kr)\mathrm{P}_m(\cos(\phi))$$



$y = w^T \phi(x)$

$h = a^T S$ $\quad S = h_m P_m$

Modeling: Anechoic conditions

Considering no sensor mismatch neither between mics nor loudspeakers…

...the recorded transfer functions at a single frequency between N_L sources and N_M positions

$$\mathbf{h} = \hat{\mathbf{h}} + \mathbf{n}$$

where

$$\hat{\mathbf{h}} = \mathbf{S}\mathbf{a}$$

with $\hat{\mathbf{h}} \in \mathbb{C}^{N_L N_M}$, $\mathbf{a} \in \mathbb{C}^{M}$ and $\mathbf{S} \in \mathbb{C}^{N_L N_M \times M}$ with elements

$$s_{mi} = \mathrm{h}_m(kr_i)\mathrm{P}_m(\cos(\phi_i))$$

Modeling: Anechoic conditions

How do we find $\mathbf{a}$?
Bayesian Inference

$$\pi(\mathbf{a}\,|\,\mathbf{h}) \propto \pi(\mathbf{h}\,|\,\mathbf{a})\pi(\mathbf{a})$$

Where priors

$$\mathbf{n} \sim \mathcal{CN}(0, \tau^{-1}\mathbf{I})$$
$$\mathbf{a} \sim \mathcal{CN}(0, \delta^{-1}\mathbf{I})$$

$$\tau \sim \mathcal{G}(\alpha, \beta), \;\; \delta \sim \mathcal{G}(\alpha, \beta)$$

Likelihood

$$\pi(\mathbf{h}\,|\,\mathbf{a}, \tau) \sim \mathcal{CN}(\hat{\mathbf{h}}, \tau^{-1}\mathbf{I}) \propto \exp(-\tau^2||\mathbf{Sa} - \mathbf{h}||^2)$$

And posterior

$$\pi(\mathbf{a}, \tau, \delta\,|\,\mathbf{h}) \propto \pi(\mathbf{h}\,|\,\mathbf{a}, \tau)\pi(\mathbf{a}\,|\,\delta)\pi(\tau)\pi(\delta)$$

$$(\mathbf{a}, \tau, \delta)_{\mathrm{MAP}} = \underset{\mathbf{a},\tau,\delta}{\mathrm{argmax}}\; \pi(\mathbf{a}, \tau, \delta\,|\,\mathbf{h})$$

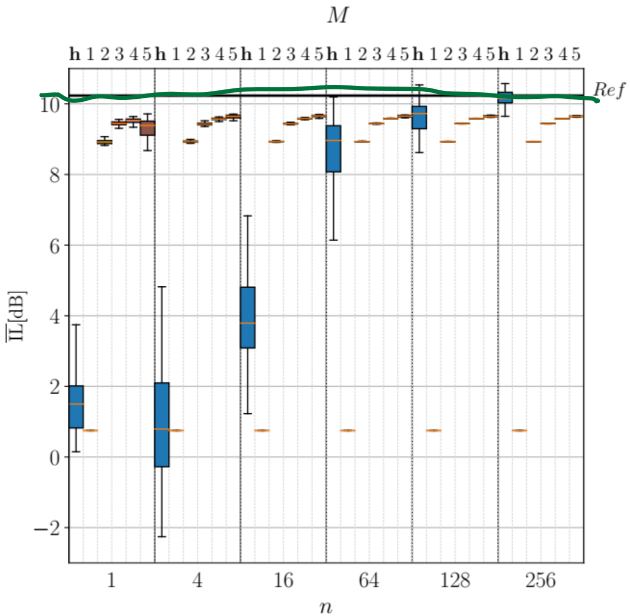
$$\hat{h}(k, \mathbf{r}_*) = \mathbf{s}_*\mathbf{a}_{\mathrm{MAP}}^{\mathrm{T}}$$

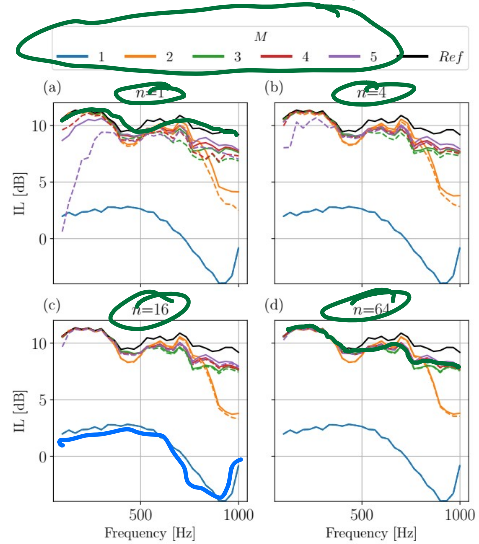
Bayesian Languages (e.g. STAN)

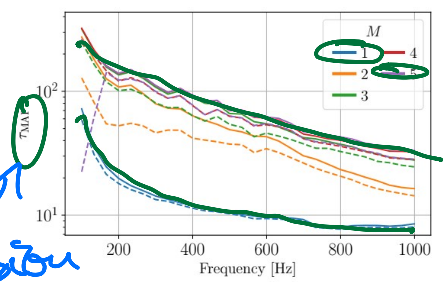
```
// ------------------ Spherical Harmonics ----------------------
data {
  int<lower=0> N_meas;                         // Total Number of independent measurements.
  int<lower=0> M;                              // Number of spherical harmonics modes.
  vector[N_meas] d;                            // Distance loudspeaker-measurement.
  vector[N_meas] pt_real;                      // Measured Pressure at receiver. Real part
  vector[N_meas] pt_imag;                      // Measured Pressure at receiver. Imaginary part
  vector[N_meas] legendre[M];                  // Legendre polynomials
  vector[N_meas] bessel[M];                    // Spherical bessel functions
  vector[N_meas] neumann[M];                   // Spherical Neumann functions
  real a;                                      // Prior Hyperparameters
  real b;                                      // Prior Hyperparameters
}
parameters {
  vector[M] Ar;                                // real part of the source strength
  vector[M] Ai;                                // imaginary part of the source strength
  real<lower=0> tau;
  real<lower=0> delta;
}
transformed parameters{
  vector[N_meas] mu_real;                      // mean of the real part of the pressure
  vector[N_meas] mu_imag;                      // mean of the imaginary part of the pressure
  real<lower=0> inv_delta;
  real<lower=0> inv_tau;
  inv_tau = 1/tau;
  inv_delta = 1/delta;
  // Loop over modes
  mu_real = 0 * d;
  mu_imag = 0 * d;
  for (m in 1:M){
    mu_real += d .* (Ar[m] * bessel[m] + Ai[m] * neumann[m]) .* legendre[m];
    mu_imag += d .* (Ai[m] * bessel[m] - Ar[m] * neumann[m]) .* legendre[m];
  }
}
model {
  tau ~ gamma(a, b);
  delta ~ gamma(a, b);
  Ar ~ normal(0, inv_delta);
  Ai ~ normal(0, inv_delta);
  pt_real ~ normal(mu_real, inv_tau);
  pt_imag ~ normal(mu_imag, inv_tau);
}
```

Modeling: Anechoic conditions

Insertion loss





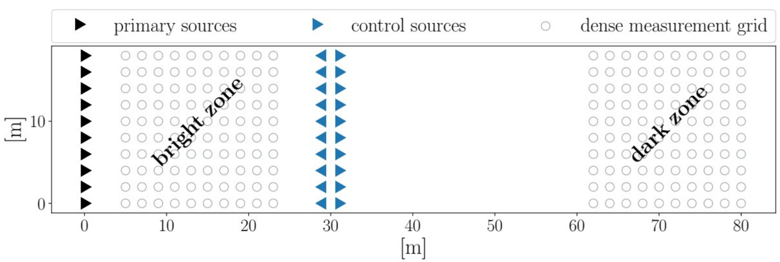


precision

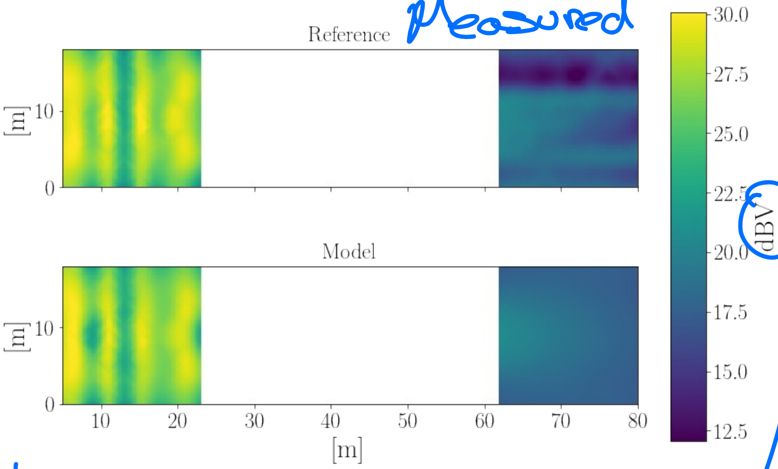
τ

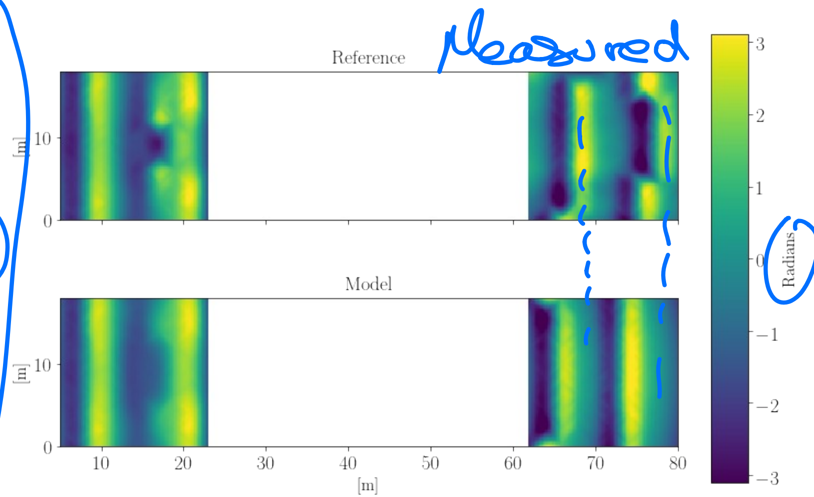
Modeling: Outdoor conditions

- Complex scenario (geometry).
- Complex medium (refraction, turbulences...).



Modeling: Outdoor conditions





Semiparametric GP

Reformulate the model (a little bit)

Physics

$$\mathbf{h} = \hat{\mathbf{h}} + \mathbf{d} + \mathbf{n}$$

Data

$$h \sim \mathcal{N}(\hat{h}, K_n)$$

Where

$$\mathbf{d} \sim \mathcal{GP}(\mathbf{0},\ \mathbf{K})$$

$$K_n = K + (\tau^{-1})^2 I$$

It introduces flexibility to the covariance matrix.

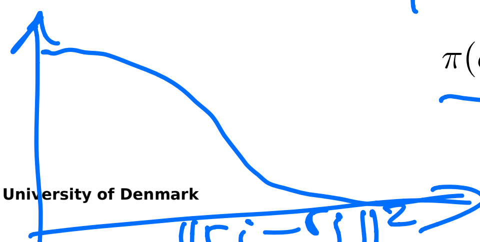
The main issue is to define a kernel that makes sense for the problem (probably spatially periodic?).

We started with something easy: radial basis function

$$\mathbf{K} = \kappa(\mathbf{R}, \mathbf{R}) = \begin{bmatrix} \kappa(\mathbf{r}_1, \mathbf{r}_1) & \dots & \kappa(\mathbf{r}_1, \mathbf{r}_n) \\ \vdots & \kappa(\mathbf{r}_i, \mathbf{r}_i) & \vdots \\ \kappa(\mathbf{r}_N, \mathbf{r}_1) & \dots & \kappa(\mathbf{r}_n, \mathbf{r}_n) \end{bmatrix}$$

$$\boxed{\kappa(\mathbf{r}_i, \mathbf{r}_j) = \alpha^2 \exp\left(-\frac{1}{2\rho^2}||\mathbf{r}_i - \mathbf{r}_j||^2\right)}$$

$$\pi(\alpha) \sim \mathcal{N}\left(0,\ \sigma_\alpha^2\right),\ \ \pi(\rho) \sim \mathcal{N}\left(0,\ \sigma_\rho^2\right).$$



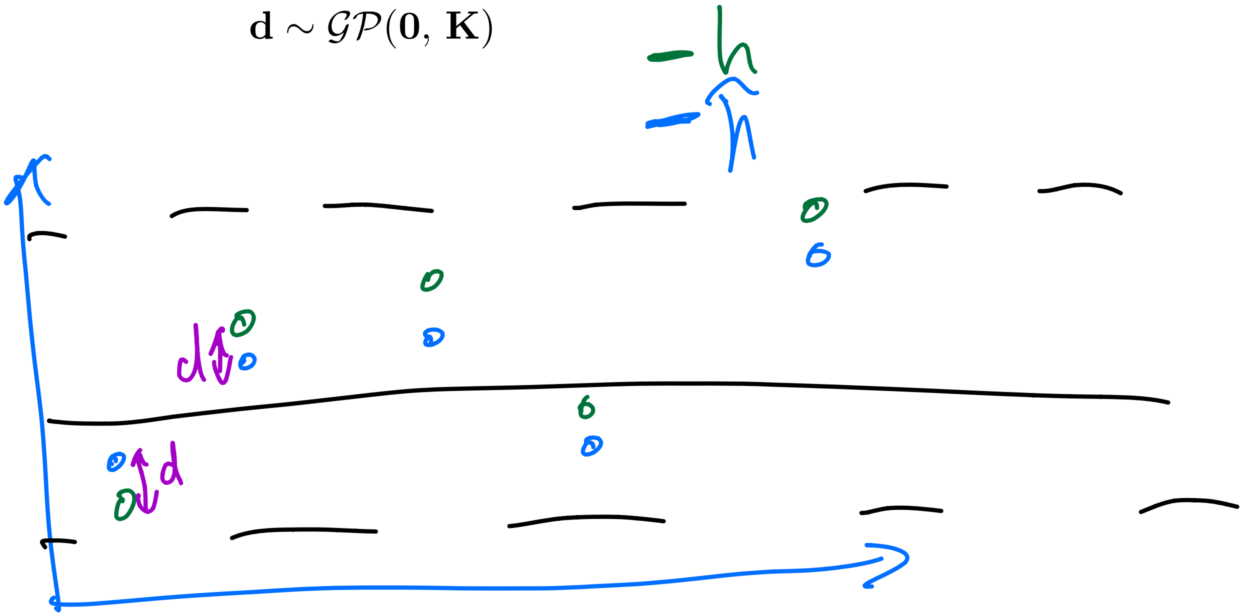
Semiparametric GP

Reformulate the model (a little bit)

$$\mathbf{h} = \hat{\mathbf{h}} + \mathbf{d} + \mathbf{n}$$

Where

$$\mathbf{d} \sim \mathcal{GP}(\mathbf{0},\, \mathbf{K})$$



Transfer function prediction

How do we predict elsewhere? A bit more complicated than before

$$\pi(\mathbf{a}, \tau, \delta \,|\, \mathbf{h}) \propto \pi(\mathbf{h} \,|\, \mathbf{a}, \tau)\pi(\mathbf{a} \,|\, \delta)\pi(\tau)\pi(\delta)$$

$$(\mathbf{a}, \tau, \delta)_{\mathrm{MAP}} = \underset{\mathbf{a},\tau,\delta}{\operatorname{argmax}} \ \pi(\mathbf{a}, \tau, \delta \,|\, \mathbf{h})$$

$$\hat{h}_{\mathrm{MAP}}(k, \mathbf{r}_*) = \mathbf{s}_* \mathbf{a}_{\mathrm{MAP}}^{\mathrm{T}}$$

$$\overline{\mathbf{h}}_* = \hat{\mathbf{h}}_*^{\mathrm{MAP}} + \left(\mathbf{K}_*^{\mathrm{MAP}}\right)^\top \left(\mathbf{K}_n^{\mathrm{MAP}}\right)^{-1} \left(\mathbf{h} - \hat{\mathbf{h}}_{\mathrm{MAP}}\right)$$

$$\mathrm{cov}(\mathbf{h}_*) = \mathbf{K}_{**}^{\mathrm{MAP}} - \left(\mathbf{K}_*^{\mathrm{MAP}}\right)^\top \left(\mathbf{K}_n^{\mathrm{MAP}}\right)^{-1} \mathbf{K}_*^{\mathrm{MAP}}$$

$$\mathbf{K}_n^{\mathrm{MAP}} = \mathbf{K}^{\mathrm{MAP}} + \frac{1}{\tau^2}\mathbf{I}$$

Transfer function prediction