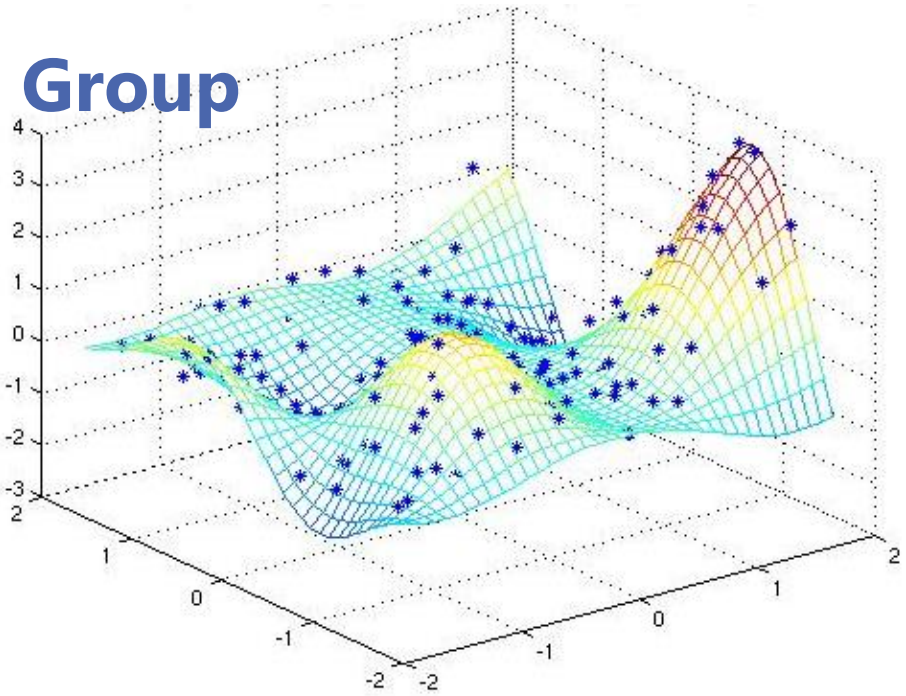
Gaussian Process Regression and Active Learning for Visual Situations – a Brief Tutorial

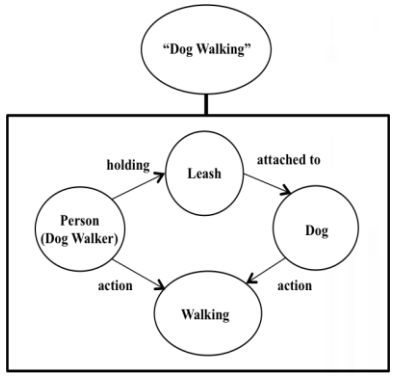
Anthony Rhodes
Melanie Mitchell Research Group

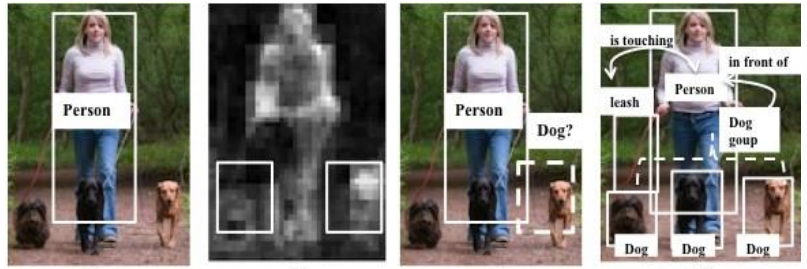
Portland State UNIVERSITY



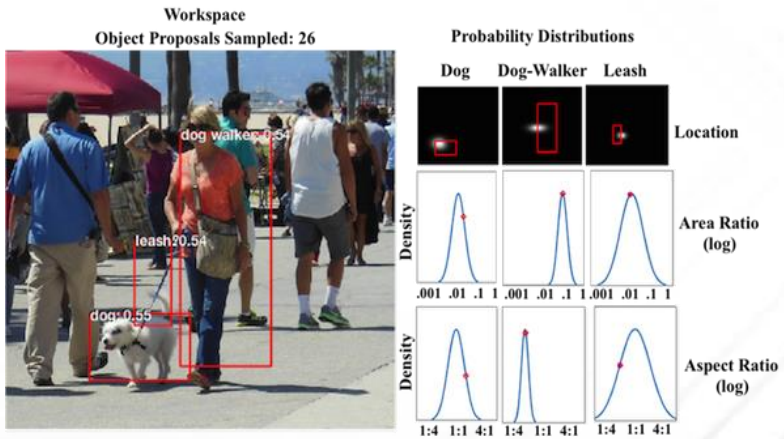
Active Object Localization in Visual Situations

- *Situate* is a computer vision framework for active object localization is visual situations.
- We define a 'visual situation', *e.g.* 'dog-walking', as an abstract concept whose image instantiations are linked more by their common spatial and semantic structure than by low-level visual similarity.



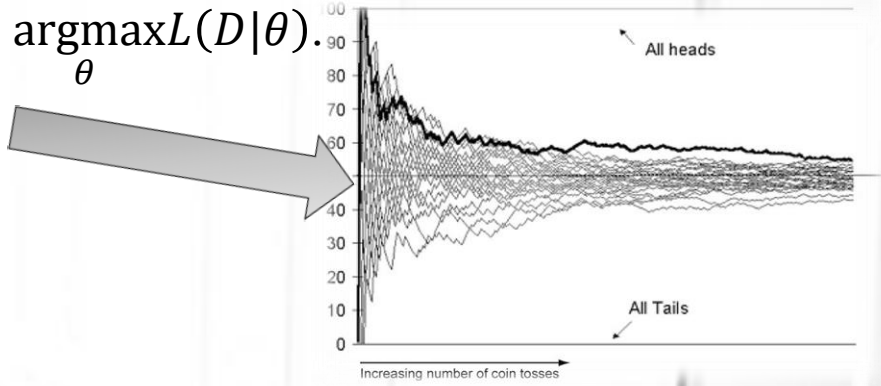


- Our system combines given and learned knowledge of the structure of a particular situation, and adapts that knowledge to a new situation instance as it actively searches for objects.



1. The Bayesian Framework

- (2) General paradigms for statistics and statistical inference: *frequentist* vs. *Bayesian*.
- Frequentists: Parameters are fixed; there is a (Platonic) model; parameters remain constant.
- Bayesians: Data are fixed; data are observed from realized sample; we encode prior beliefs; parameters are described probabilistically.
- Frequentists commonly use the *MLE* (**maximum likelihood estimate**) as a cogent *point estimate* of the model parameters of a probability distribution: $\hat{\theta}_{MLE} = \underset{\theta}{\text{argmax}} L(D|\theta)$.
- Using the *Law of Large Numbers (LLN)*, $\lim_{n\to\infty} P(|\bar{X}_n - \mu| > \varepsilon) = 0$, one can consequently show that: $\hat{\theta}_{MLE} \xrightarrow{P} \theta$.



Potential issues with frequentist approach: philosophical reliance on long-term 'frequencies', *the problem of induction* (Hume) and the black swan paradox, as well as the presence of limited exact solutions for a small class of settings.

1. The Bayesian Framework (cont'd)

- In the Bayesian framework, conversely, probability is regarded as a measure of uncertainty pertaining to the practitioner's knowledge about a particular phenomenon.
- The prior belief of the experimenter is not ignored but rather encoded in the process of calculating probability.
- As the Bayesian gathers new information from experiments, this information is used, in conjunction with prior beliefs, to update the measure of certainty related to a specific outcome. These ideas are summarized elegantly in the familiar *Bayes' Theorem*:

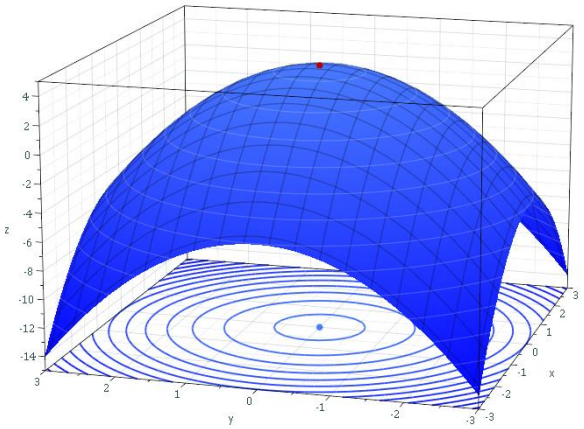
$$P(H|D) = \frac{P(D|H)P(H)}{P(D)}$$

- Where H here connotes '*hypothesis*' and D connotes '*data*'; the leftmost probability is referred to as the *posterior* (of the hypothesis), and the numerator factors are called the *likelihood* (of the data) and the *prior* (on the hypothesis), respectively; the denominator expression is referred to as the *marginal likelihood*.
- Typically, the point estimate for a parameter used in Bayesian statistics is the *mode* of the *posterior distribution*, known as the **maximum a posterior** (MAP) estimate, which is given as:

$$\hat{\theta}_{MAP} = \underset{\theta}{\text{argmax}} P(D|\theta)P(\theta)$$

2. Bayesian Optimization

- *Function optimization* is formulated as the problem of optimizing a function over a compact set A (*i.e.* a bounded subset of a finite-dimensional Euclidean space). More formally, the aim of global function optimization is to determine:

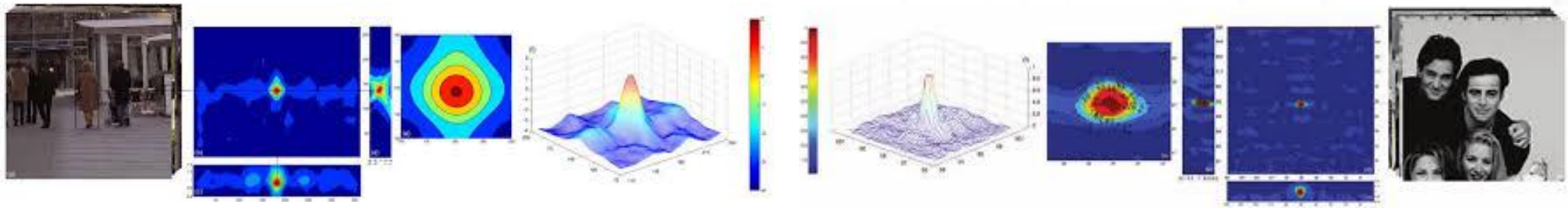


$$\max_{x \in A \subset \mathbb{R}^d} f(x)$$

- We call *f(x)* the *objective function* in this setting. For local optimization, we relax the maximality condition whereby we determine a point x^* such that: $f(x^*) \geq f(x'), \forall x \in A \; s.t. \|x^* - x\| < \delta.$
- Ideally, the objective function is known in the sense that we can write it in closed form.
- However, in many application domains the objective function can be non-covex, 'expensive' to evaluate or it can only be approximated through Monte Carlo or some such data simulation process.

2. Bayesian Optimization (cont'd)

- For the purposes of the Situate model, we wish to efficiently locate target objects in a test image based on a pre-trained classifier (or some proxy function *qua* an oracle) response.
- Gualdi *et al* [2012] present evidence that classifier responses form 'basins of attraction' around target objects, radiating outward from the center of an object, as shown below.



- We wish to optimize a *surrogate function* (*i.e.* a cheap approximation) for *classifier response* (our true objective function) over a test image in order to *actively* determine the most likely location of an object of interest.
- Why is this problem challenging? Several reasons: Calls to a classifier can be computationally expensive (we'd like to minimize these operations); classifier response signals are noisy and (probably) non-convex, with an intractable derivative and in general they do not admit of a closed-form representation.

2. Bayesian Optimization: Methodology

Fortunately, as we now show, Bayesian optimization is an exceptionally well-suited method for solving the problem of function optimization under these challenging circumstances.

A Bayesian Framework:

Let $x_i \in \mathbb{R}^d$ be the *i*th sample/observation from a data set $D_{1:t} = \{x_{1;t}, f(x_{1:t})\}$ consisting of *T* total sample pairs of inputs and outputs under f. In the Bayesian setting we would like to estimate the posterior distribution $P(f|D_{1:t})$ of the objective function given these data. To do so, we simply apply *Bayes' Rule* so that the posterior distribution is obtained by multiplying the likelihood and prior distributions as follows:

$$P(f|D_{1:t}) \propto P(D_{1:t}|f)P(f)$$

2. Bayesian Optimization: Methodology (cont'd)

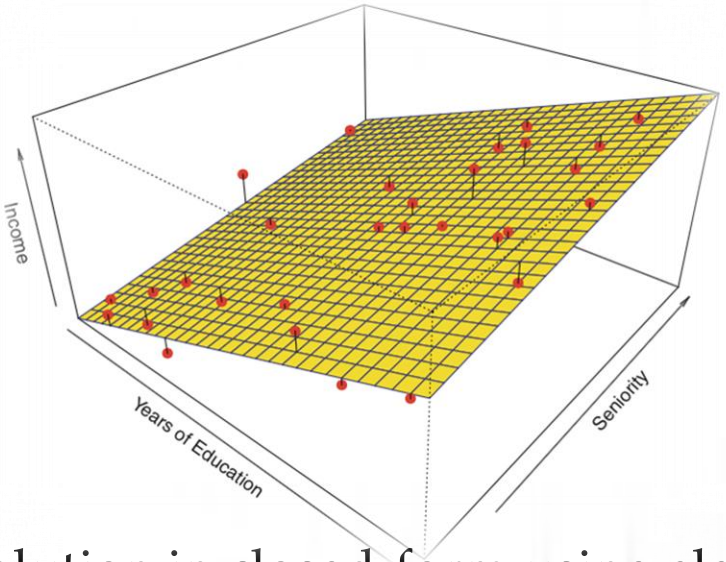
$$P(f|D_{1:t}) \propto P(D_{1:t}|f)P(f)$$

- Having combined our prior knowledge with the information gleaned from our evidence, we next aim to maximize the posterior iteratively.
- However, because we do not have access to the objective function directly, we choose instead to optimize our *surrogate function*.
- Crucially, we must then decide where to sample next according to the principle of *maximum expected utility*, which is itself a secondary optimization challenge.
- For our problem, we identify utility using *acquisition functions* that strike a balance between minimizing uncertainty and greedy optimization for the surrogate to the posterior. This method is formally defined in the context of active learning, which we explore in short order.
- Continuing our general algorithm in this fashion, we aim to iteratively decrease the distance between the true global maximum of the objective function and the expected maximum given the model.

3. Preliminaries: OLS Regression

- There exist (2) Equivalent approaches to data-fitting with OLS regression. Given a data set of d covariates and response variable y: $D = \{(x, y)\}_{1:n}, x \in \mathbb{R}^d, y \in \mathbb{R}$, we essay to find the optimal **linear** fit for the data by minimizing the residual least squares loss function:

$$L(a_{1:d}) = \sum_n \left(\hat{f}(x) - y \right)^2$$



- (Approach 1) We can solve for the optimal solution in closed-form using elementary techniques from multivariable calculus, whereupon we set:

$$\frac{\partial L}{\partial a_i} = 2 \hat{f}_{a_i} \sum_n \left(\hat{f}(x) - y \right) = 0 \ \forall \ 1 \leq i \leq n$$

- This technique yields the so-called '**Normal equations**' (*i.e.* the OLS solution) in closed-form.

3. Preliminaries: OLS Regression (cont'd)

- (Approach 2) More commonly, if we represent the OLS problem as finding the coefficients that minimize the residual error of the overdetermined system:

$$Ax = b, \;\; A \in \mathbb{R}^{n \times d}, x \in \mathbb{R}^{d}, b \in \mathbb{R}^{n}$$

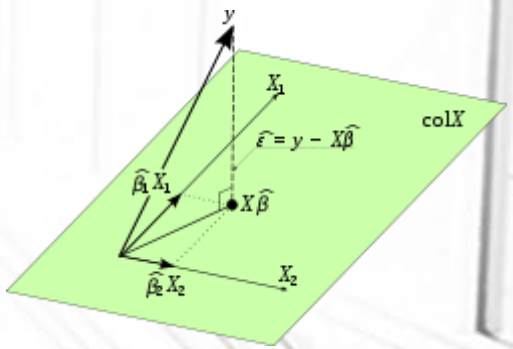
- In this case, projecting the vector *b* orthogonally onto *Col(A)* yields the Normal equations in matrix form:

$$A^T Ax = A^T b$$

- If we solve this equation (which is always soluble) we arrive at the OLS solution (equivalent to the previous slide) in matrix form.

$$\hat{x} = \left(A^T A\right)^{-1} A^T b, \;\; A \in \mathbb{R}^{n \times d}, x \in \mathbb{R}^{d}, b \in \mathbb{R}^{n}$$

- This same approach can be used to solve a great variety of related regression problems, including polynomial and logistic regression.

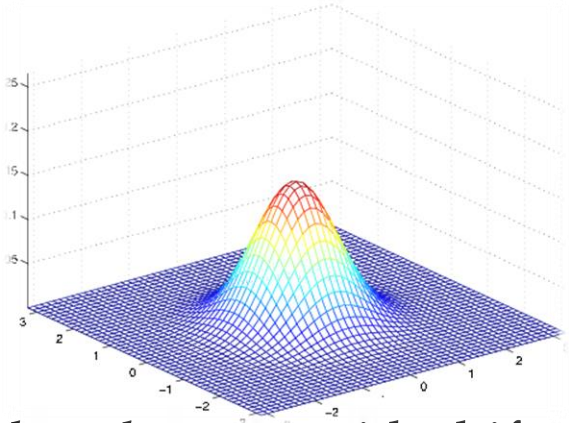


4. Preliminaries: The Multivariate Gaussian Distribution

- Recall that the multivariate Gaussian (*MVG*) distribution is fully parameterized, together, by its mean vector (μ)and covariance matrix (Σ).

- The *pdf* for a MVG in *d* dimensions is defined as follows:

$$N\left(x\middle|\mu,\Sigma\right)\triangleq\frac{1}{\left(2\pi\right)^{d/2}\left|\Sigma\right|^{1/2}}\exp\left[-\frac{1}{2}\left(x-\mu\right)^{T}\Sigma^{-1}\left(x-\mu\right)\right]$$



- MVG distributions are higher-dimensional realizations of *bell-shaped curves*, with shift-location controlled by the mean vector, and scaling-shape controlled by the covariance matrix. It is trivial to see that a conditional MVG distribution is itself a MVG.

- Using the notion of *Schur complements* in conjunction with the *matrix inversion lemma*, one may derive the following useful closed-form formulas for marginal and conditional MVGs:

- Suppose x=(x_1,x_2) is jointly Gaussian with parameters: $\mu=\begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix},\Sigma=\begin{pmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{pmatrix},\Lambda=\Sigma^{-1}$

  Then, the marginal(s) and posterior conditional are given by:

$$p\left(x_1\right)=N\left(x_1\middle|\mu_1,\Sigma_{11}\right),\ p\left(x_2\right)=N\left(x_2\middle|\mu_2,\Sigma_{22}\right),\ \ p\left(x_1\middle|x_2\right)=N\left(\mu_{1|2},\Sigma_{1|2}\right),$$

$$\mu_{1|2}=\mu_1+\Sigma_{12}\Sigma_{22}^{-1}\left(x_2-\mu_2\right),\ \ \Sigma_{1|2}=\Sigma_{11}-\Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

5. The Weight-Space View of Regression (a Bayesian Approach)

- With standard linear regression we assume the function outputs are modeled as a linear combination of the input values and a vector of weights (parameters):

$$f(x) = x^T w, \;\; where: y = f(x) + \varepsilon, \;\; \varepsilon \sim N\left(0, \sigma^2\right), \;\; x, w \in \mathbb{R}^d$$

- To 'solve' the regression problem in the Bayesian manner, we first express the posterior distribution over the weights and then we 'turn the Bayesian crank' (*i.e.* use Bayes' Rule) in order to formulate this posterior in terms of the known likelihood and prior; finally we compute the *MAP* estimate.

- $$p(w|X,y) = \frac{p(y|X,w)p(w)}{p(y|X)} = \frac{p(y|X,w)p(w)}{\int p(y|X)\,dw} = \frac{N_y(X^T w, \sigma^2 I_T) N_w(0, \Sigma_p)}{\int N_y(X^T w, \sigma^2 I_T) N_w(0, \Sigma_p) dw} = N_y(X^T w, \sigma^2 I_T) N_w(0, \Sigma_p)$$

- This quantity results in an *Chi-square* distribution (in terms of w) – which is *prima facie* unhelpful.

- However, with some clever algebraic manipulations, it is possible to reformulate this posterior distribution [*see Rasmussen and Williams for full derivation] so that:

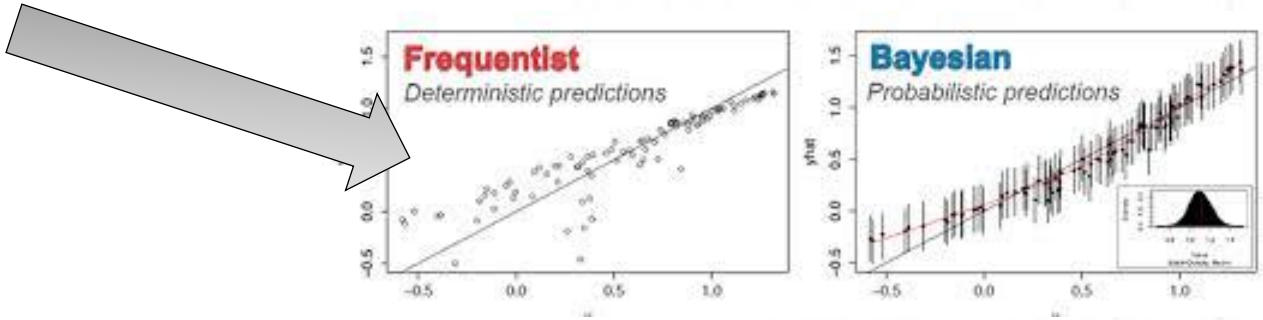
$$p(w \mid X, y) \sim N\left(\bar{w}, A^{-1}\right), \;\; where: \bar{w} = \sigma^{-2} A^{-1} Xy \in \mathbb{R}^d, A = \sigma^{-2} XX^T + \Sigma_p^{-1} \in \mathbb{R}^{d \times d}$$

5. The Weight-Space View of Regression (cont'd)

- Now, in order to generate a *posterior surrogate prediction* for new data points x_*, we use:

$$p(f_* \mid x_*, X, y) = \int p(f_* \mid x_*, X, y)\, p(w \mid X, y)\, dw = N_{f_*}\left(x_*^T \bar{w}, x_*^T A^{-1} x_*\right)$$

- Intuitively, this result is a consequence of multiplying the test input by the mean of the posterior over the weights, where the predictive variance is a quadratic form of the test input with the posterior covariance over the weights (both properties follow from the *linearity of expectation*).
- In summary, we now have a probabilistic grounding for linear regression.



- Note that the preceding model has limited expressiveness due to its linearity.
- Fortunately, this limitation is easily remedied by introducing ***basis functions***, which we consider next. As we show, the posterior surrogate predictive model will translate felicitously to the domain of basis functions.

5. The Weight-Space View of Regression (basis functions)

- To this end we introduce a basis function $\phi(x): \mathbb{R}^d \to \mathbb{R}^N$; many candidates for basis functions exist in particular, for the setting of polynomial regression we would set $\phi(x) = (1, x, x^2, ..., x^{N-1})$.
- Our linear regression is now expressed by: $f(x) = \phi(x)^T w, with\ w \in \mathbb{R}^N$. This representation is often referred to as *explicit* in the feature space. It consequently follows from the previous results that the predictive distribution with respect to the feature space is given as:

$$p(f_* \mid x_*, X, y) = N_{f_*}\left(\phi(x_*)^T \bar{w}, \phi(x_*)^T A^{-1} \phi(x_*)\right)$$

- One potential problem with this approach is that it necessitates the inverse of a (possibly large dimension) *NxN* matrix ($O(N^3)$). This dilemma is further compounded by the fact that larger values of *N* tend to conversely improve the expressiveness of our model.
- Fortunately, it is possible to convey this posterior distribution in terms of *implicit features* (*i.e.* with respect to a kernel transformation) using the so-called 'kernel trick.' Aided with the *Woodbury matrix identity* (we suppress these details for clarity), we can reformulate the above equation as follows:

$$p(f_* \mid x_*, X, y) = N\left(\phi(x_*)^T \Sigma_d \Phi\left(K + \sigma^2 I\right)^{-1} y, \phi(x_*)^T \Sigma_d \phi(x_*) - \phi(x_*)^T \Sigma_p \Phi\left(K + \sigma^2 I\right)^{-1} \Phi^T \Sigma_d \phi(x_*)\right)$$

- Where we define $\phi(X) = \Phi$ as the image of the design matrix with respect to the basis function in column-vector form and $K = \Phi^T \Sigma_d \Phi$.

5. The Weight-Space View of Regression (basis functions, cont'd)

$$p(f_* \mid x_*, X, y) = N\left(\phi(x_*)^T \Sigma_d \Phi\left(K + \sigma^2 I\right)^{-1} y, \phi(x_*)^T \Sigma_d \phi(x_*) - \phi(x_*)^T \Sigma_p \Phi\left(K + \sigma^2 I\right)^{-1} \Phi^T \Sigma_d \phi(x_*)\right)$$

- Notice that all of the computations in the above formulae occur **in the feature space** (*i.e.* $\mathbb{R}^n$, with $n < N$, where n is the size of the data set).
- We have thus devised an arbitrarily flexible probabilistic framework for linear regression. In particular, our procedure 'lifts' computations from the feature space to a relatively small dimensional space whose size is determined by the cardinality of the training data.
- There exists a comparable method for solving the regression problem from an alternative, **function-space viewpoint**.
- Gaussian processes (GPs), as we detail next, apply inference over a 'feature space' of functions (where N = ∞!) in a structurally simpler and more directly interpretable fashion than the foregoing weight-space approach.
- The critical drawback of the GP regime, as we show, rests in the computational burden associated with posterior calculations for large data sets. To circumvent this potential problem, we apply active learning in the data collection process and later show that our procedure renders excellent results even with modest-sized data sets.

6. Gaussian Processes and the Function-Space View of Regression

- For our purposes, we wish to first collect a small sample of expensive data generated by a trained classifier response.
- Next, we want to build a plausible estimate (a *surrogate*) of the function of classifier responses over a training image and to then 'optimize' this potentially non-convex function with respect to a likely object bounding-box for the target of interest.
- Unfortunately, our data collection process does not yield a closed-form expression for this objective function nor does it yield explicit information about its derivative.
- Gaussian processes offer several benefits in this challenging setting.
- First, with a potentially low computational overhead, as we show below, we can build a regression model over objective functions with tunable smoothness and scale parameters.
- Furthermore, the use of Gaussian processes for regression in this manner facilitates the efficiency of later steps in our proposed object localization algorithm, including the use of reinforcement-inspired learning for adaptive improvements to the search process and bounding-box optimization over the regression approximation to classifier response.

6. Gaussian Processes and the Function-Space View of Regression (cont'd)

- ***Definition***. A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

- A GP is uniquely defined by the choice of its mean and covariance functions:

$$f(x) \sim GP\left(m(x), \kappa(x, x')\right)$$
$$m(x) = \mathrm{E}\left[f(x)\right]$$
$$\kappa(x, x') = \mathrm{E}\left[\left(f(x) - m(x)\right)\left(f(x') - m(x')\right)^T\right]$$

- A Gaussian Process defines a prior over an *infinite-dimensional* function space.

- GP regression (GPR) is nonetheless nominally performed in an n-dimensional space (where n is the sample size); we say that the GPR method is consequently *data-driven* (*i.e.* non-parametric).

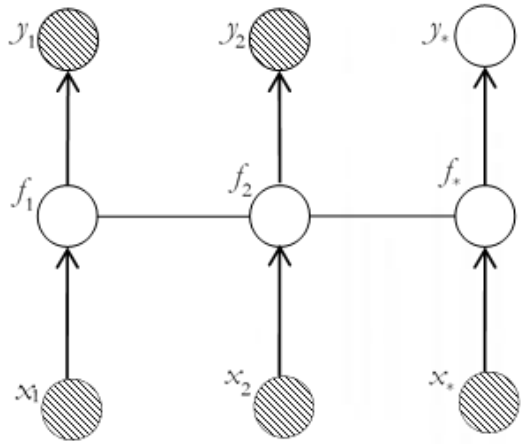
- Note that inference over the infinite function-space vs. the data-driven finite-dimensional space is equivalent for GPR. Why? The inclusion of unobserved data points does not affect the maximum of the posterior function distribution. $\left(\max p\left(f_* \mid D_{1:t+1}\right) = \max p\left(f_* \mid \tilde{D}_\infty\right)\right)$

6. Gaussian Processes and the Function-Space View of Regression (cont'd II)

- Our GPR inference problem can be articulated as a marginalization over the function space:

$$p(f_* \mid D_{1:t+1}) = p(f_* \mid f, D_{1:t+1}) = \int p(f_* \mid f, x_*) p(f \mid D_{1:t}) df \rightarrow p(f \mid \tilde{D}_\infty) = p(f \mid D_{1:t}) = N(f \mid \mu, K)$$

- Where a prior over functions is also provided in the full specification of the GP; $K_{ij} = \kappa(x_i, x_j)$

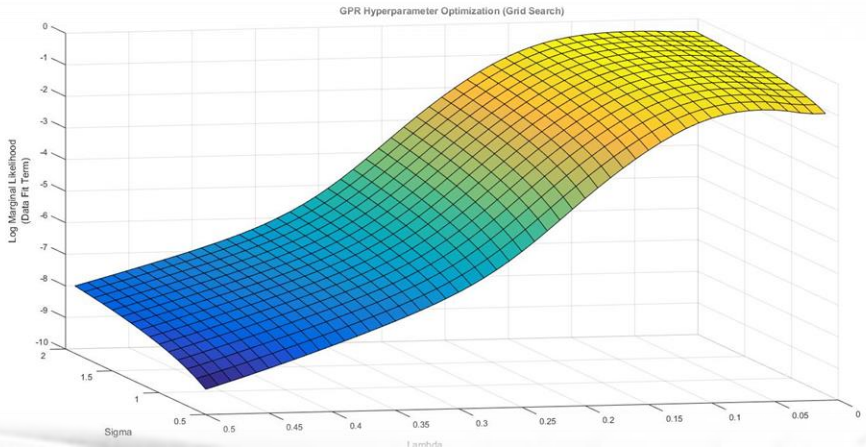


- Unlike the usual regression setting, our goal here is not to learn a distribution over function parameters (or parameters of a distribution) but rather, we are determining a distribution over functions.

- Because of its non-parametric nature, the chief source of model bias for GP regression lies in the assumption of its jointly Gaussian data. Whereas ordinary regression is limited in its expressibility by the designation of a model complexity parameter (*e.g.* polynomial degree), with GP regression, the value of the model is limited only by the *quality* of the data.

7. Kernel/Covariance Functions

- Many choices of kernel functions (and their associated hyperparameters) for GPR exist, including stationary/non-stationary kernels, isotropic kernels, etc.
- Throughout, we use the squared-exponential kernel, as it possess several desirable properties, including the fact that SE kernel is the most widely-studied and frequently used in machine learning; a function generated by a GPR with the SE kernel is infinitely differentiable and serves as a universal function approximator [*Andrew Wilson, 2014].
- $$\kappa_{SE}(x, x') = \sigma_f^2 \exp\left[-\frac{1}{2\lambda^2}(x - x')^2\right]$$
- Where, of the two tuneable hyperparameters, σ determines the vertical variation of the model, and λ is the model length scale.
- We optimize these hyperparameters for Situate using an exhaustive grid search:



8. GPR for Situate with Noise-Free Data

- We now explicitly specify GPR for classifier response with object localization for Situate with noise-free data.
- Suppose, as before, that we have a training set consisting of n pairs of inputs and noise-free classifier responses, denoted $D_{1:n} = \{x_{1;n}, f(x_{1:n})\}$.
- Our goal is to soundly predict the (new) objective function outputs f_* for a test set X_* of dimension (X_* *x d)* where n_* is the size of the test set (*e.g.* when n_*=1 we perform 'on-line' regression); d=2 for 'spatial' GPR with Situate.
- Intuitively, our predictive model should perfectly interpolate values for the training set in the noiseless case with no uncertainty.
- Moreover, as in the case of the classical function interpolation paradigm, the uncertainty in the predictive model should naturally increase in proportion to the distance of each test point from the training data. Distance-based kernel functions, particularly the *SE* function, harnesses these desirable properties.
- The definition of the GP regression problem yields: $\begin{pmatrix} f \\ f_* \end{pmatrix} \sim N\left(\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \begin{pmatrix} K & K_* \\ K_*^T & K_{**} \end{pmatrix}\right)$

Where K is the n *x* n covariance matrix on the training data, $K_*=k(x,x_*)$ is the 'mixed' covariance matrix of dimension n *x* n_*, and $K_{**}=k(x_*,x_*)$ is the covariance matrix for the test data.

8. GPR for Situate with Noise-Free Data (cont'd)

- From the aforementioned results for MVGs, the posterior of the GP has the following form:

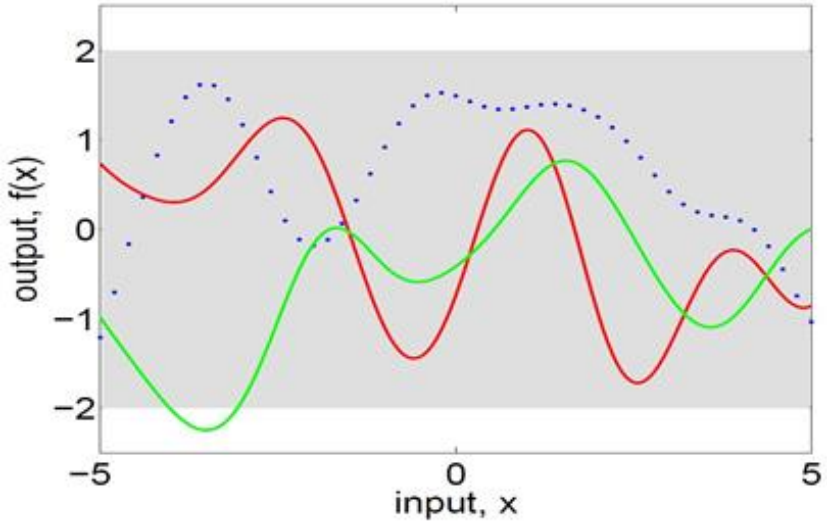
$$p\left(f^{*}\middle|X_{*},X,f\right)=N\left(f^{*}\middle|\mu_{*},\Sigma_{*}\right),\ \ with$$

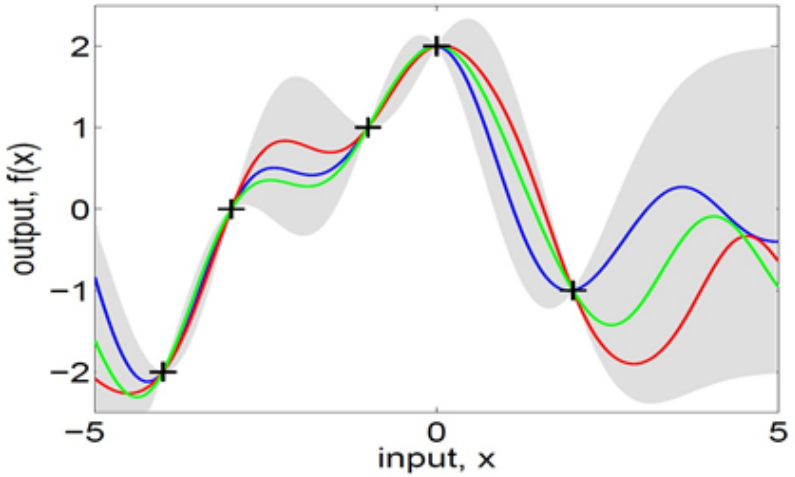
$$p\left(f^{*}\middle|X_{*},X,f\right)=N\left(f^{*}\middle|\mu_{*},\Sigma_{*}\right),\ \ and\ \Sigma_{*}=K_{**}-K_{*}^{T}K^{-1}K_{*}$$

- From these expressions we can see that the chief computational burden of GPR rests in the inversion of the K matrix which (generally) requires $O(n^3)$ operations.
- Despite these ostensible issues, we hypothesize that if one takes the necessary precautions to ensure that our data set is small, dynamically chosen and of sufficiently high quality, then our GPR model should be highly efficient for object localization.
- Our ability to properly leverage information gleaned from the expensive classifier response is governed, we argue, by the ingenuity in the use of this costly data as opposed to applying a computational brute force approach.
- Given the GP regression predictor model above, it is now possible to perform function prediction/approximation. For Situate, we can think of the GP prior as informed by various situational/contextual cues.

8. GPR for Situate with Noise-Free Data (cont'd II)

- One can employ GPR in this framework as a probabilistic, *generative* function model.





- The figure on the left shows samples drawn from a GPR prior (*i.e.* without data); the image on the right depicts samples taken from the posterior distribution; note the instance of perfect interpolation achieved due to noise-free modeling.

*Image credit: Rasmussen & Williams (2006)

9. GPR for Situate with Noisy Data

- The previous GPR approach is easily adapted to the case of *noisy data*; note that in many applications, classifiers yield a noisy signal.
- Consider the data set: $\hat{D}_{1:n} = \left\{ x_{1;n}, y\left(x_{1:n}\right) \right\}, with\ y(x) = f(x) + \varepsilon, \varepsilon \sim N\left(0, \sigma_y^2\right)\ (IID)$ Let:

$$\hat{\kappa}_{SE}(x, x') = \sigma_f^2 \exp\left[-\frac{1}{2\lambda^2}(x - x')^2 \right] + \sigma_y^2 \delta_{xx'}, \hat{K} = K + \sigma_y^2 I_n.$$

- The joint density of the observed data and the predicted, noisy is output is modeled by:

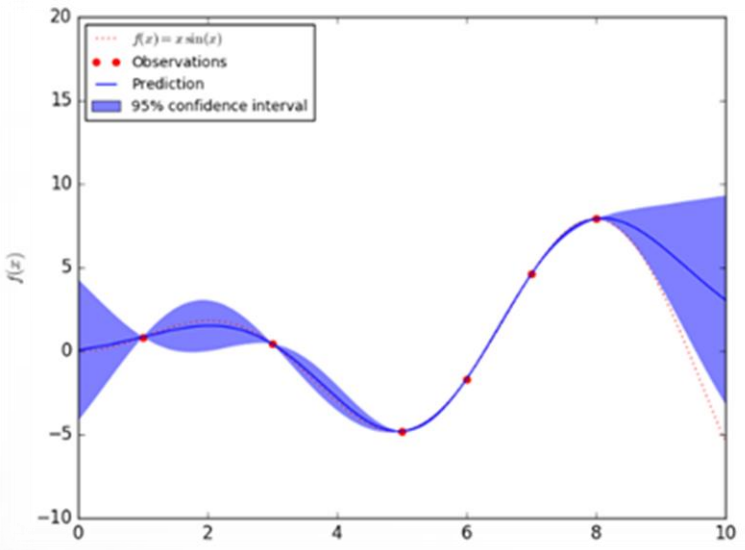
$$\begin{pmatrix} y \\ f_* \end{pmatrix} \sim N\left(\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \begin{pmatrix} \hat{K} & K_* \\ K_*^T & K_{**} \end{pmatrix} \right)$$

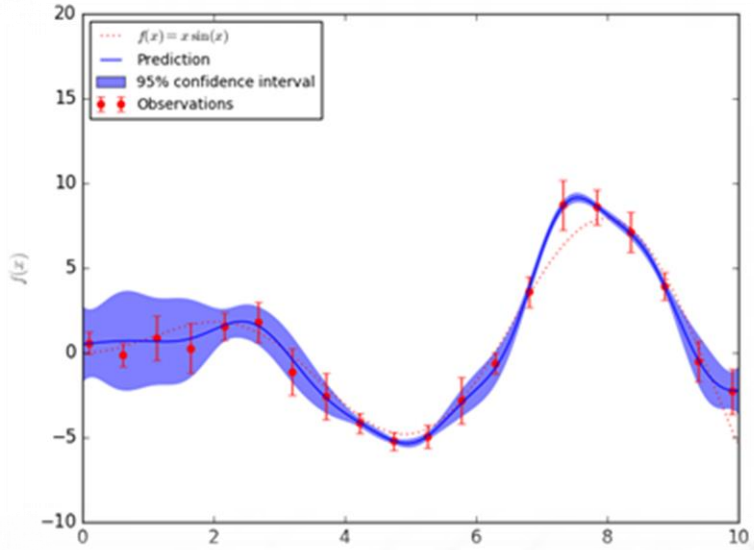
- As before, we arrive at the key posterior predictive equations for noisy GPR:

$$p\left(f_* \middle| X_*, X, y \right) = N\left(f_* \middle| \mu_*, \Sigma_* \right),\ \ \mu_* = \mu\left(X_*\right) + K_*^T \hat{K}^{-1}\left(y(X) - \mu(X) \right),\ \ \Sigma_* = K_{**} - K_*^T \hat{K}^{-1} K_*$$

9. GPR for Situate with Noisy Data (cont'd)

- In the figure below a comparison of GP regression with no noise (left) and noisy observations (right) is shown.
- In both cases the surrogate (*i.e.* predicted) function is contrasted with a ground truth function. In both cases, a 95% confidence interval is displayed.
- Note that this confidence band is narrower in the noiseless case and also for instances in close proximity to other points or clusters of points for which the model demonstrates high predictive confidence. Interpolation is, of course, imperfect even for observed points due to the inclusion of noise for the image on the right.





10. Active Learning Queries

- Next we consider the task of efficiently querying the pixel space of a target image for new data points which we then use to build our GP-generated response surface to guide the object localization process.
- In choosing new data points, we naturally want to collect the 'best' data available at a minimum computational cost.
- But as we stated from the outset, our data collection process is expensive primarily due to the fact that it relies on calculating a classifier response and a subsequent response surface.
- We accordingly favor collecting 'better' data as opposed to simply extracting more data (*viz.*, good data vs. big data) when determining new data queries. Under these conditions, *active learning* is an ideal methodological approach for building a model with few – but highly informative – data points.

10. Active Learning Queries (cont'd)

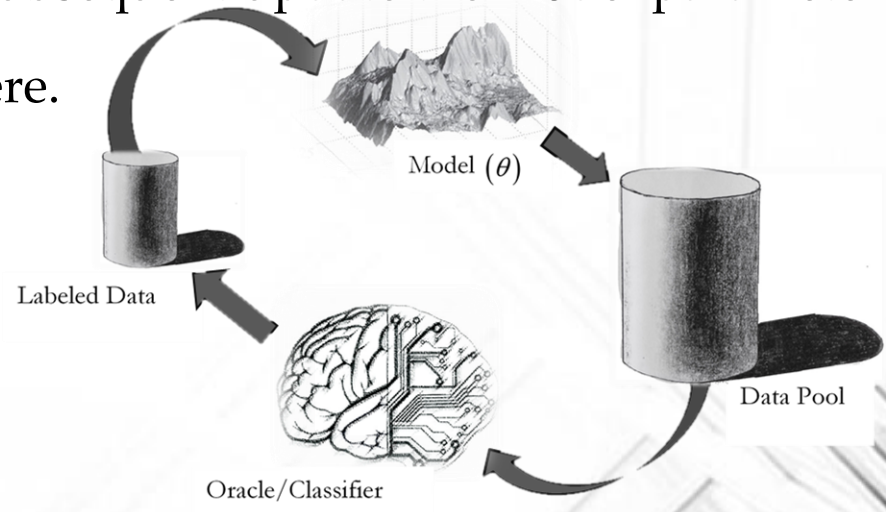
- Active learning is a particular case of *semi-supervised learning* where the bulk of our data (raw target image pixels) are unlabeled with respect to a target object classifier response initially, but the learning *agent*, *i.e*. the algorithm, may actively query a classifier/oracle to obtain output labels for desired data points.

- The main undertaking of the agent in active learning is to make a decision as to which data points to query (formulating a query strategy); this decision is encapsulated formally through an acquisition function that depends on a design parameter, ξ_a.

- After acquiring new data points, the model parameters $\theta=(\mu,K)$ defining the GPR model are updated, and the process continues in this fashion until terminating.

- Here we determine the expected 'utility' of the active data query process using the current model parameters θ; we then maximize this utility and subsequent update the model parameters.

- The active learning, *query-update cycle* is shown here.



10. Active Learning Queries (cont'd II)

- Ideally, in addition to *exploring* regions of high uncertainty, our agent should also *exploit*, to some degree, 'regions of promise', respecting our target object.
- This compromise between exploratory actions, whose goal is to gather information and predacious actions used to exploit known information in order to achieve a particular objective is known as the *exploration-exploitation tradeoff*.
- In order to give our active learning approach a theoretical foundation, we can express *utility* (as is convention) as prior-posterior gain in Shannon information:

$$U\left(y,\xi_a\right)=D_{KL}\left(p\left(\theta\,|\,y,\xi_a\right)\middle\|\,p\left(\theta\,|\,\xi_a\right)\right)$$

- Here we may think of utility as conveying the information gained when we update our prior belief to the posterior, after having acquired a new observation.
- It is possible to show (details omitted for brevity) that $U\left(\xi_a\right)=I\left(\theta;y\right)$ (I is *mutual information*); this indicates that in order to 'solve' the data query question lying at the heart of active learning, one should intuitively choose a design parameter, *i.e.* a query strategy, that maximizes the degree to which the model parameters and observations 'share' information (namely they are highly *dependent*).
- This means that our response surface will converge for an optimal experimental design to the true objective function, as desired.

11. Acquisition Functions

- In the framework of Bayesian optimization, ***acquisition functions*** are used to guide the search for the optimum of the surrogate (classifier response) to the true objective function (object location).

- Intuitively, acquisition functions are defined in such a way that *high acquisition* indicates greater likelihood of an objective function maximum; we accordingly define acquisition functions that encapsulate a data query experimental design that favors either regions of large response values, large uncertainty, or a combination of both.

- More formally, we sample the surrogate $f(\cdot)$, *i.e.* the realization of the Gaussian process, at the point(s) , $\arg\max_x u\left(x \mid D_{1:n}\right)$ where $u(\cdot)$ is an acquisition function.

- We define (2) experimental acquisition functions:

- (1) We define the *probability of improvement* over the incumbent $f(x^+)$ as:

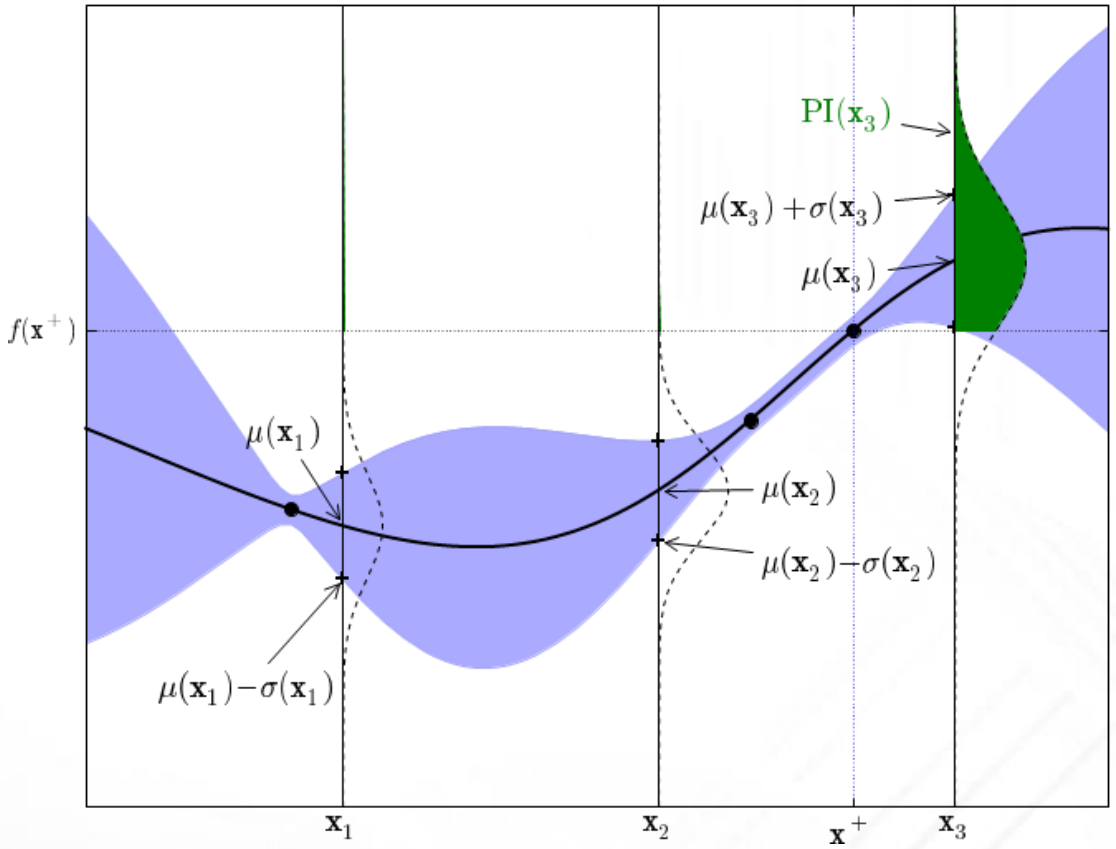
$$u_1 \triangleq PI_\xi(x) = P\left(f(x) \geq f\left(x^+\right) + \xi\right) = \Phi\left(\frac{\mu(x) - f\left(x^+\right) - \xi}{\sigma(x)}\right)$$

With $x^+ = \arg\max_{x_i \in D_{1:t}} f\left(x_i\right)$ and $\Phi(\cdot)$ is defined as the Gaussian cumulative distribution function, *i.e.*

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-t^2/2} dt$$ and $\xi \geq 0$ is the design parameter.

11. Acquisition Functions (cont'd)

- In the figure we display a Gaussian process showing the region of *probable improvement*. The maximum observation is at $\mathbf{x}^+$.
- The darkly-shaded area in the superimposed Gaussian above the dashed line can be used as a measure of improvement, I($\mathbf{x}$). The model predicts almost no possibility of improvement by observing at $\mathbf{x}_1$ or $\mathbf{x}_2$, while sampling at $\mathbf{x}_3$ is more likely to improve on f($\mathbf{x}^+$).
-



11. Acquisition Functions (cont'd)

- A second, alternative acquisition function which accounts for both the probability of improvement *and* also the magnitude of improvement a point can potentially yield we call the *expected improvement*, in line with the work of Močkus. [*Mockus, Jones 1998]
- Concretely, we should choose a new data point that maximizes the expected improvement with respect to $f(\mathbf{x}^+)$, the current surrogate maximum. Define the *improvement function*:
- $$I(x) = \max\left\{0,\, f_{t+1}(x) - f\left(x^+\right)\right\}$$
- We thus set our new data point to: $x_{n+1} = \arg\max_x E\left(I(x) \mid D_{1:n}\right).$
- The *likelihood of improvement I* for the posterior is accordingly distributed:

$$\ell(I) = \frac{1}{\sqrt{2\pi}\sigma(x)} \exp\left[-\frac{\left(\mu(x) - f\left(x^+\right) - I\right)^2}{2\sigma^2(x)}\right]$$

- It follows that $E(\mathrm{I})$, *the expected value of the improvement* can be expressed as
- $$EI(x, Z) = \begin{cases} \mu(x) - f\left(x^+\right)\Phi(Z) + \sigma(x)\phi(Z) & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases}$$

With $Z = \dfrac{\mu(x) - f\left(x^+\right)}{\sigma(x)}$, and $\phi(\cdot)$, $\Phi(\cdot)$ are the *pdf* and *cdf* of the Gaussian distribution, respectively.

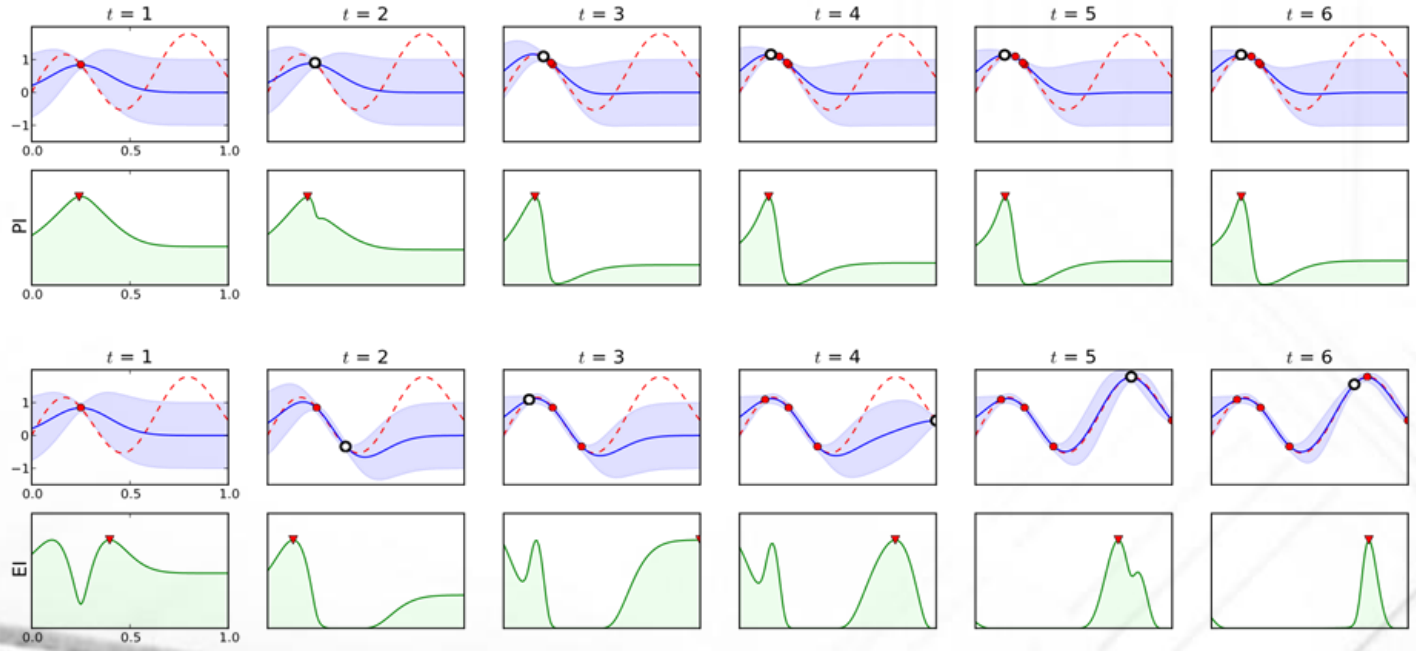
11. Acquisition Functions (cont'd II)

- As in the previous case of the probability improvement acquisition function, we introduce the parameter ξ_a as a means to fine-tune the exploitation-exploration trade-off of our algorithm.

- This gives the following definition for the *expected improvement* query function:

$$u_2 \triangleq EI\left(x, Z_\xi\right) = \begin{cases} \left(\mu(x) - f\left(x^+\right) - \xi\right)\Phi(Z) + \sigma(x)\phi(Z) & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases} \quad \text{with } Z_\xi = \frac{\mu(x) - f\left(x^+\right) - \xi}{\sigma(x)}$$

- Below are example iterations of both *PI* and *EI*-based GPR approximations.



12. The GPSituate Algorithm (proposed)

(*) Given test image with target object (t_{obj}); test image is compressed for computational speed-up at run-time.

(1) **Initialize/learn model parameters and settings, including**: α (*proxy intensity value* used for oracle); β (*Bayesian parameter* used to balance GP learning and reliance on conditioned priors through Situate); γ (a spatial parameter related to the intensity proxy function), ζ (a policy-related parameter); σ, λ (these are SE, *viz.*, squared-exponential, kernel parameters); various sample batch sizes.

Choice of policy for GPR: PI (probability improvement), EI (expected improvement), possibly others.

Note: all values can be set to change dynamically – in fact, optimally, they should be assigned dynamically.

(2) Sample from learned salience/spatial and BB scale priors for t_{obj} in Situate; compute $f_{proxy}(x)$ for this sample set; this yields a batch, 'labeled' set: $D_t = \{(x_i, f_{proxy}(x_i))\}$ for iteration 't'.

(3) Build the GPR using: $p(f^* | X_*, X, f) = N(f^* | \mu_*, \Sigma_*))$ for the posterior mean, $\mu(x, y)$, and posterior variance, $\sigma^2(x, y)$.

12. The GPSituate Algorithm (cont'd)

(4) Choose new sample points based on policy chosen in (2) and situational data conveyed through Situate; we use the Bayesian parameter β (this should change dynamically with each step), which should reflect how much confidence we have in the GPR 'evidence' vs. the degree to which we 'trust' the current Situate (conditional) density models. As a basic example, consider an iteration of the procedure with sample batch size n=10; here we might sample 5 points using a PI/EI policy optimization, 3 sample points are based on MVG learned densities and current contextual data (i.e. provisional localizations) and 2 sample points are chosen stochastically (possibly just using max entropy/uncertainty).

(5) Repeat steps (3) and (4) until GPR algorithm satisfies halting conditions.

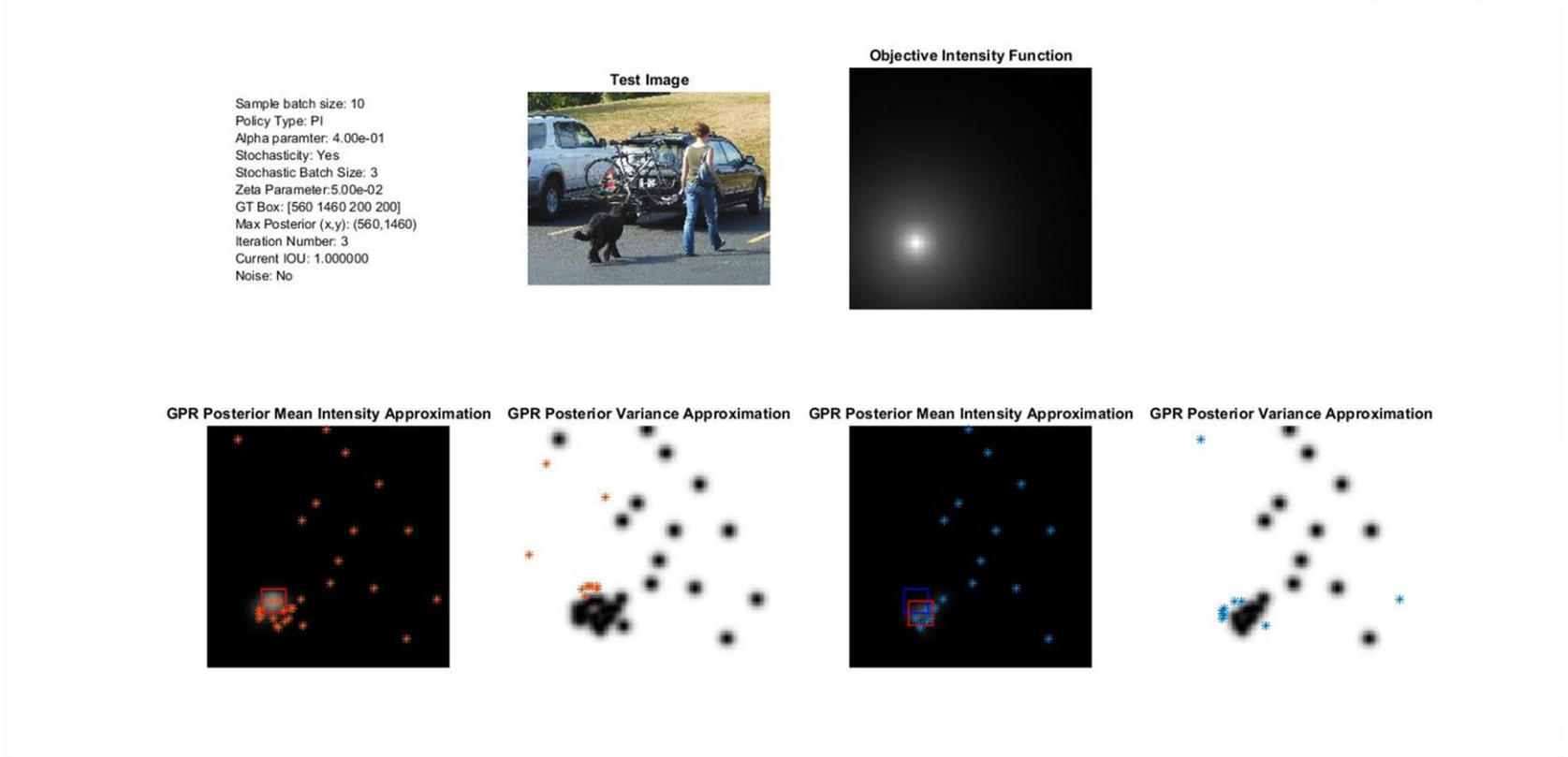
(6) Sample 'x' number of bounding box size parameters (using Situate engine); check $IOU\left(BB_{GT}, \{BB_P\}\right) \geq 0.5$ for this set of proposals.

12. GPSituate Algorithm Experiments

- For my experiments I devised a reasonable oracle, which I call the intensity proxy:

$$f_{proxy}(\vec{x}) = \alpha \cdot IOU(BB_{GT}, BB_P) + (1-\alpha)\exp\left[\frac{-\gamma\left\|GT_{center} - \vec{x}\right\|}{img_{scale\_factor}}\right]$$

Example 1: Dog localization



- Figure: Using sample batch size=10 and PI active search policy, the dog was perfectly localized in only three iterations of the GPR algorithm. Top-left box: stats; top-right: objective intensity function (learned using proxy function); lower-right two boxes: (left) GPR posterior mean intensity for iteration 2; blue points are previously sampled points; blue box is proposal BB, red box is GT; (right) GPR posterior variance approximation for iteration 2; blue points show the next sample points proposed by PI algorithm (note the excellence of the choice of points), included are stochastically-chosen points to ensure exploratory actions; lower-left two boxes – similar data for iteration 3, the final iteration.

12. GPSituate Algorithm Experiments (cont'd)

Example 2: Walker localization

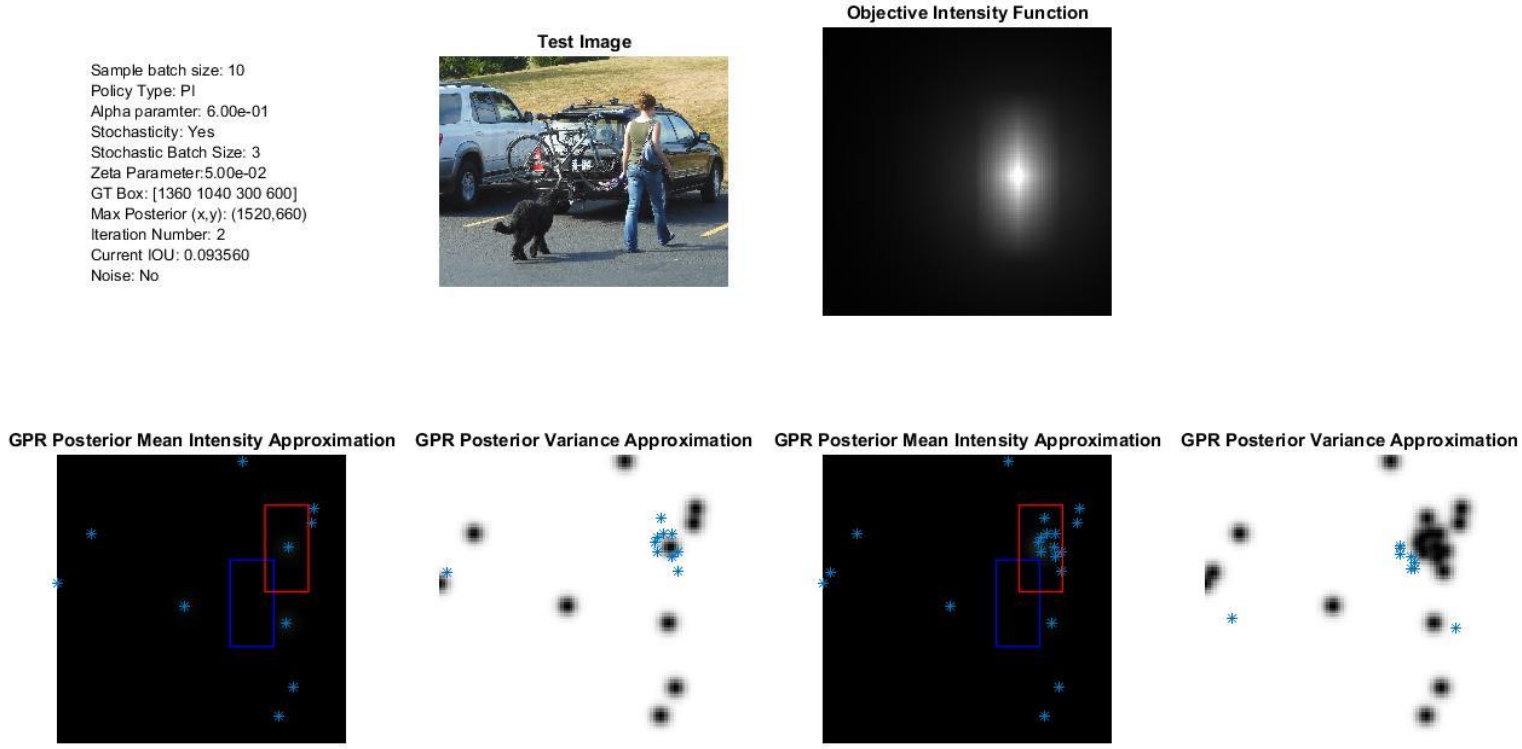


Figure. The first two iterations of GPR are shown; first iteration on left, second on right. Note that the active learning search first searches, radially, near the point of greatest interest; in the next iteration the algorithm is already nearly certain as to the direction to move to ensure object localization (lower-left direction). The object was perfectly localized after six iterations (note that the 'learning parameter' is set low here and so this convergence could be sped up optimally, in under six iterations.

12. GPSituate Algorithm Experiments (cont'd II)

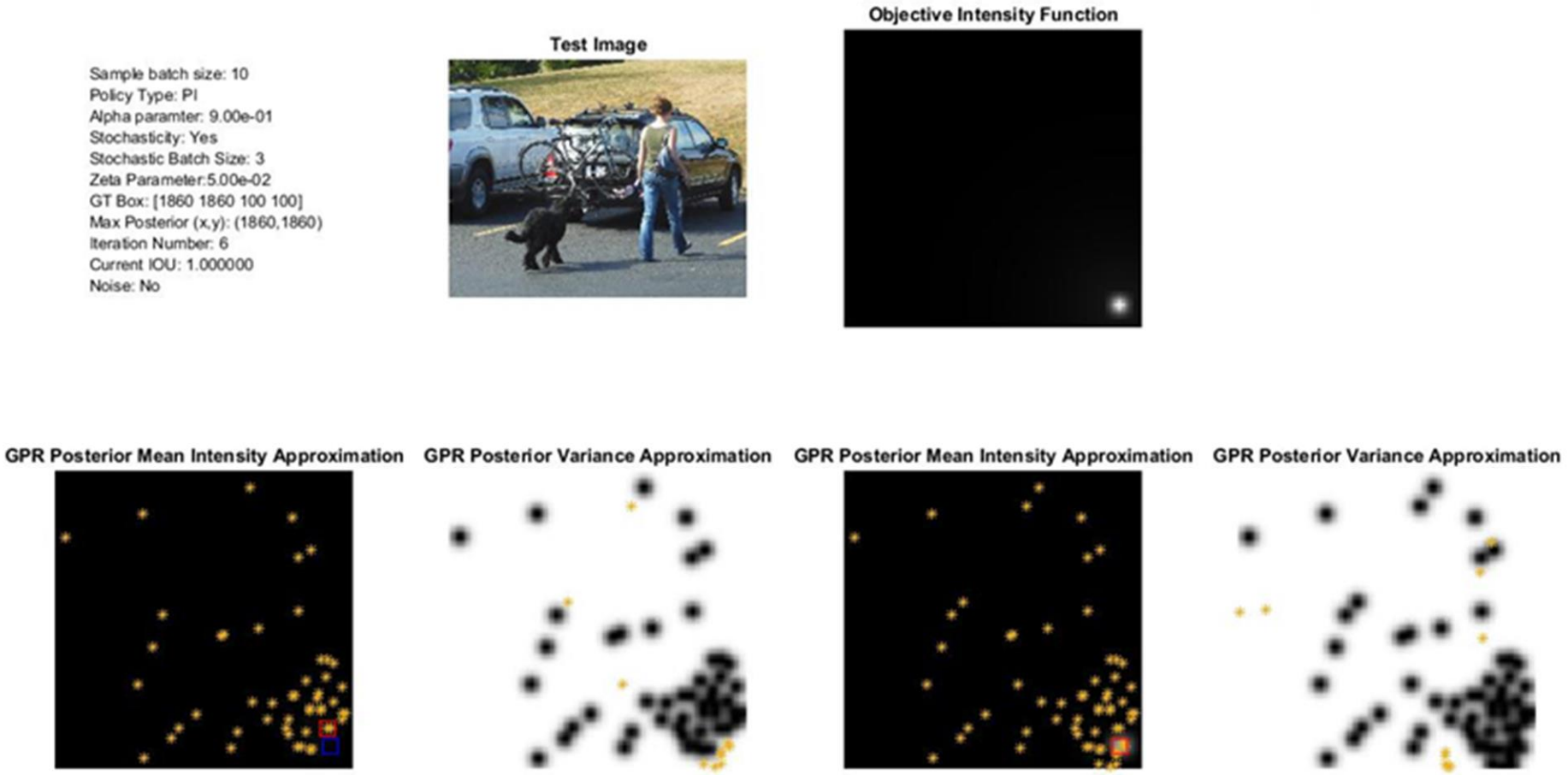


Figure. GPR algorithm perfectly localizes object for very difficult localization task (small target object with weak signal/objective function) in only six iterations.

12. GPSituate Algorithm Experiments (with noise)

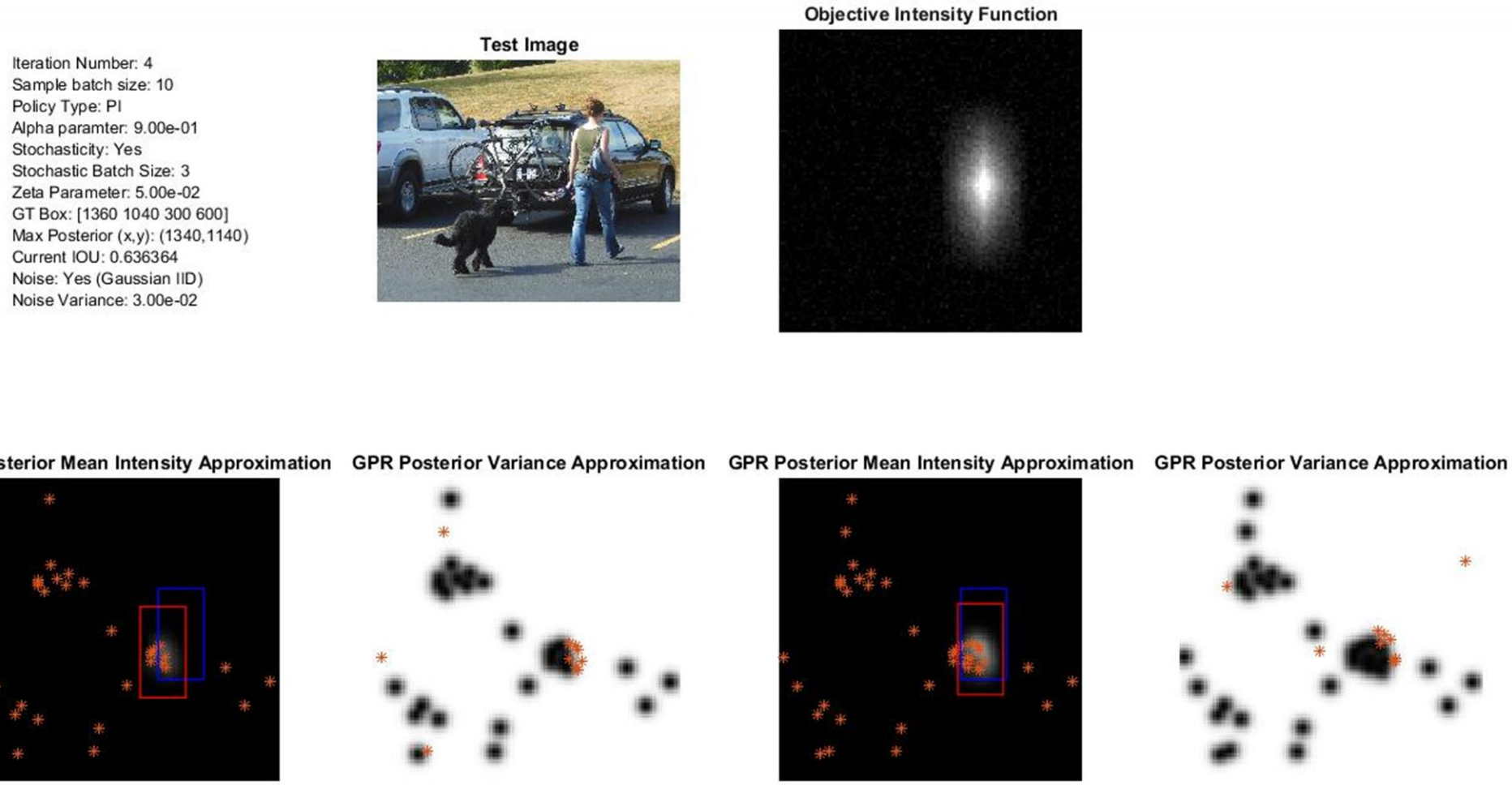


Figure. Noise localization with sigma = .03.

12. GPSituate Algorithm Experiments (with noise, cont'd)

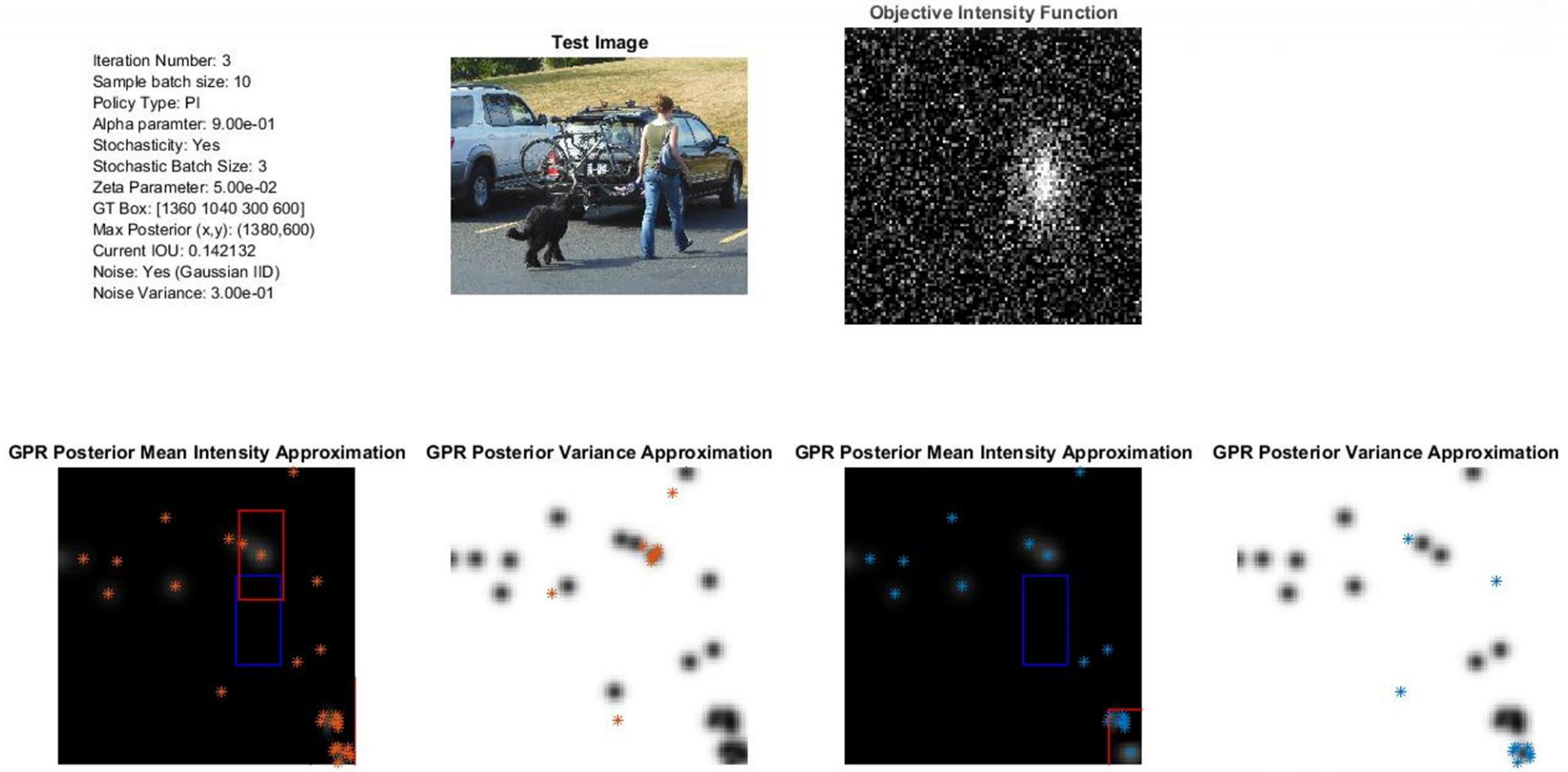


Figure. Noise localization with sigma = .3.

13. Some General Comments

(*) Primary Hypothesis: Using the general framework of Bayesian optimization, GP regression can be used to efficiently guide active object localization in computer vision.

-- Test simulations have confirmed this hypothesis overwhelmingly.

-- By means of analogy, the task of the GPSituate algorithm is to locate a 'needle in a haystack' with extreme efficiency. Commonly; the algorithm locates the exact center of a bounding box, say in a grid of 40,000 pixels (a 2000x2000 image is downsampled to 200x200 for computational efficiency – incidentally the method works whether we downsample or not) within a pixel or two in just a few iterations.

-- If we compare this task, for example, with the UAV-based (NIPS paper, 2015) presented by professor Khan (winter term @ PSU), our task is both: (i) more difficult (we aim to localize several objects in a space with a BB proposal, whereas the task in his work is to simply locate a single 'pixel') and (ii) more conceptually intricate, as we utilize contextual cues and the apparatuses associated with Situate.

-- GPSituate is highly modular; it works exceptionally well with an oracle (even a 'weak' oracle, see examples below); in addition, it can be used with any type of classifier, including CNN-based classifiers.

-- GPSituate utilizes information ascertained from *all* proposal windows (including negative 'localizations').

-- GPSituate requires NO TRAINING (ostensibly); computations are in real-time using low-dimensional GP-based calculations (naturally if one uses a CNN for classification, prior density models, salience models, etc., these processes require their own training regimes).

-- GPSituate can be implemented with my visualization code for real-time demos in talks, etc.

-- The proposed GPR algorithm can be used in a variety of different fields, including applications beyond computer vision such as: real-word object localization, spatial-search, regression-based inference (economics, epidemiology, etc.).