Computational Systems Biology
Deep Learning in the Life Sciences

6.802 6.874 20.390 20.490 HST.506

David Gifford

Lecture 8

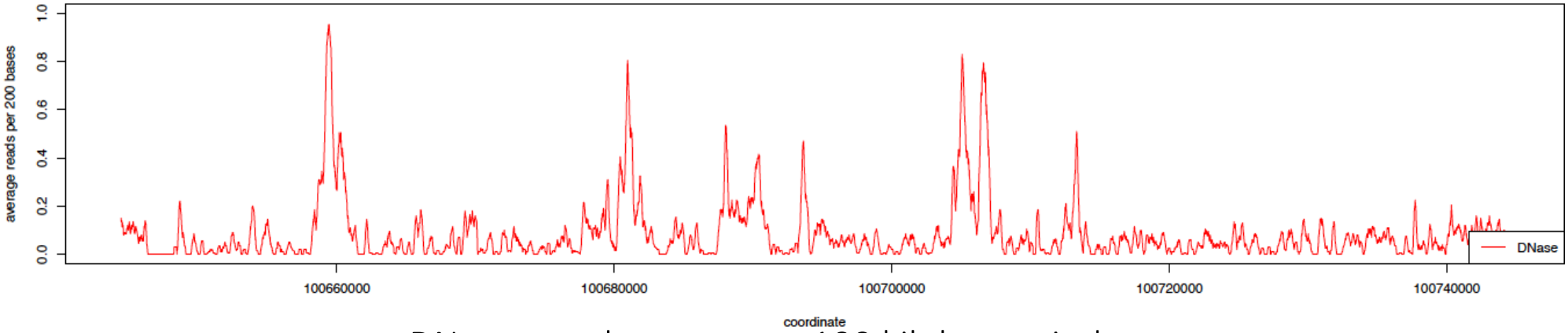
March 3, 2020

Characterizing Uncertainty
Experiment Planning

http://mit6874.github.io

Predicting chromatin accessibility

Can we predict chromatin accessibility directly from DNA sequence?



DNase-seq data across a 100 kilobase window
(Chromosome 14 K562 cells)

Motivation –

1. Understand the fundamental biology of chromatin accessibility

2. Predict how genomic variants change chromatin accessibility

Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks.

David R. Kelley
Jasper Snoek
John L. Rinn

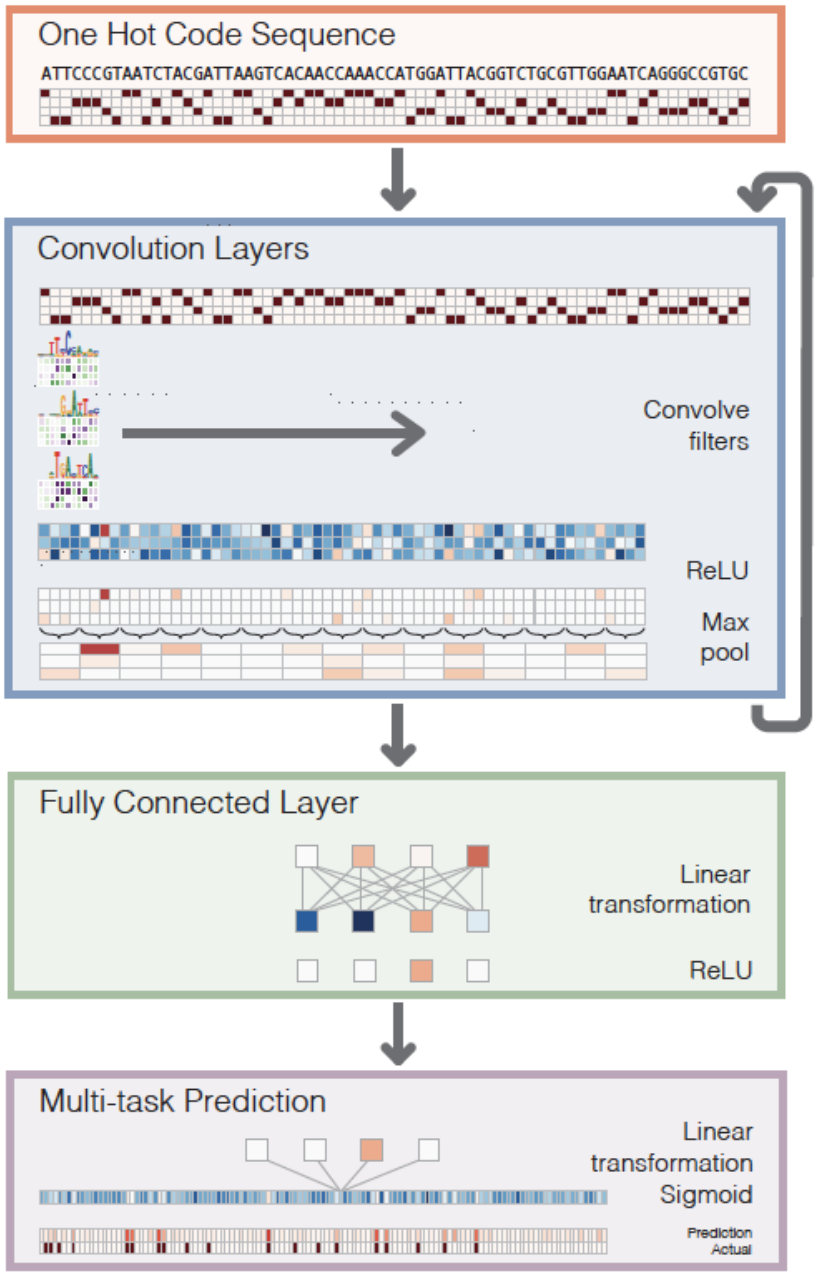
Genome Research, March 2016

Bassett architecture for accessibility prediction

Input:
600 bp

1.9 million
training examples

Output:
168 bits

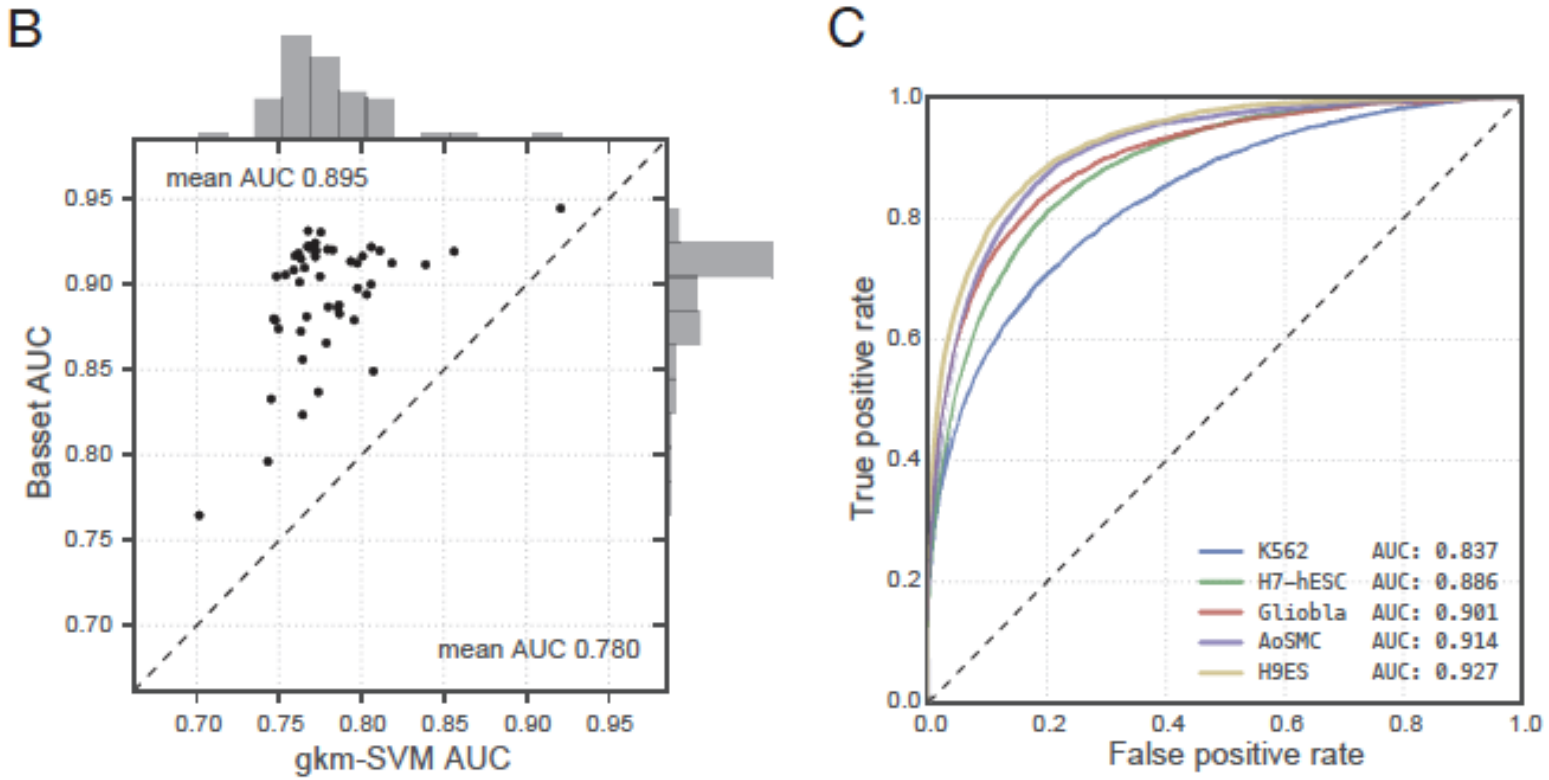


300 filters
3 conv layers
3 FC layers

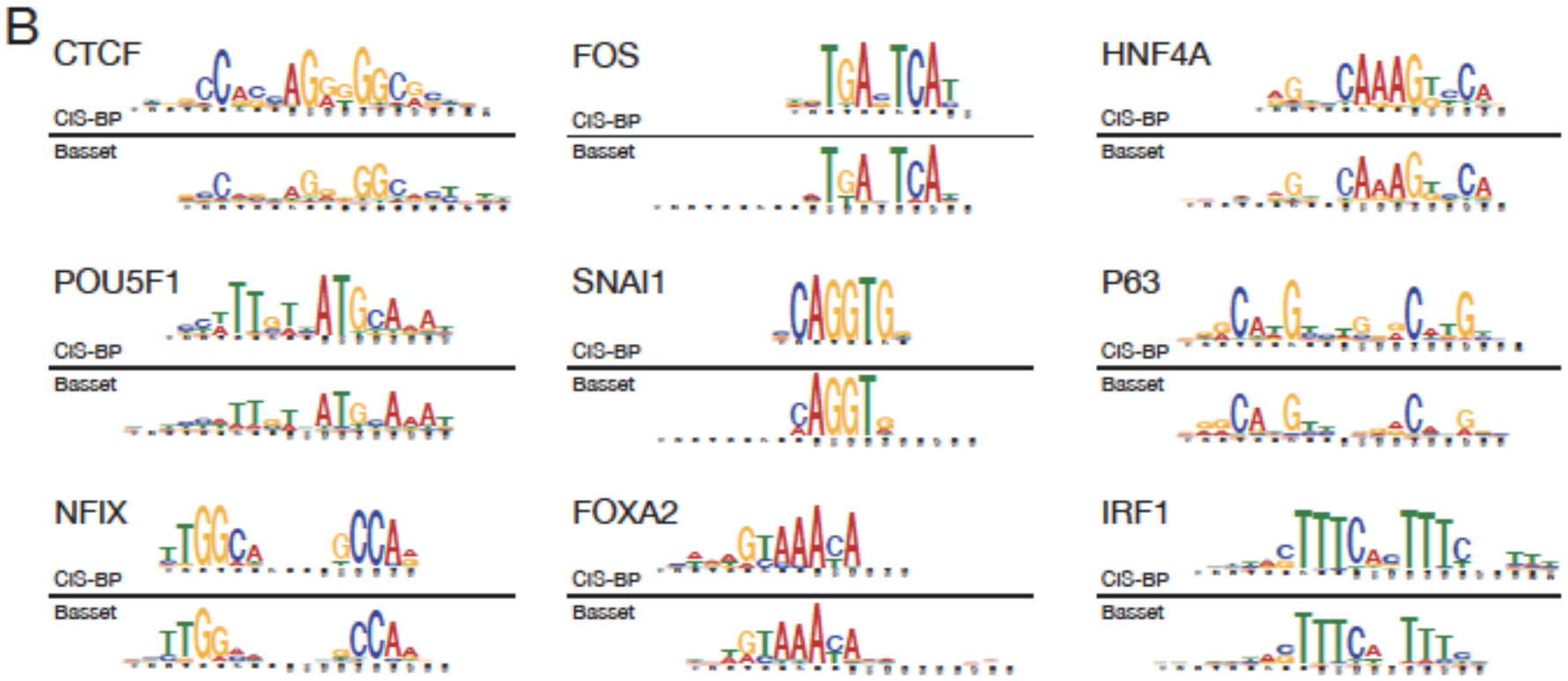
3 fully connected layers

168 outputs
(1 per cell type)

Bassett AUC performance vs. gkm-SVM

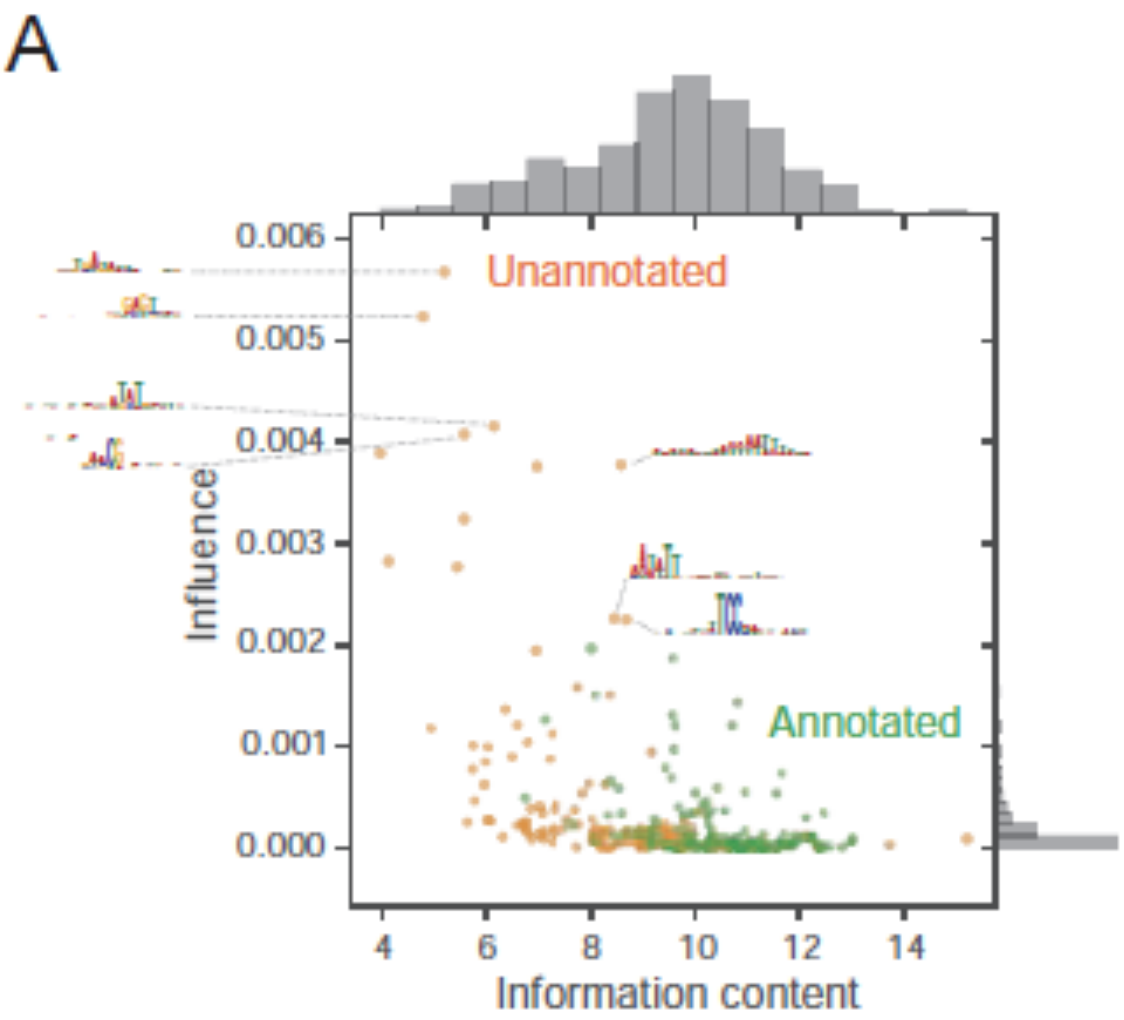


45% of filter derived motifs are found in the CIS-BP database

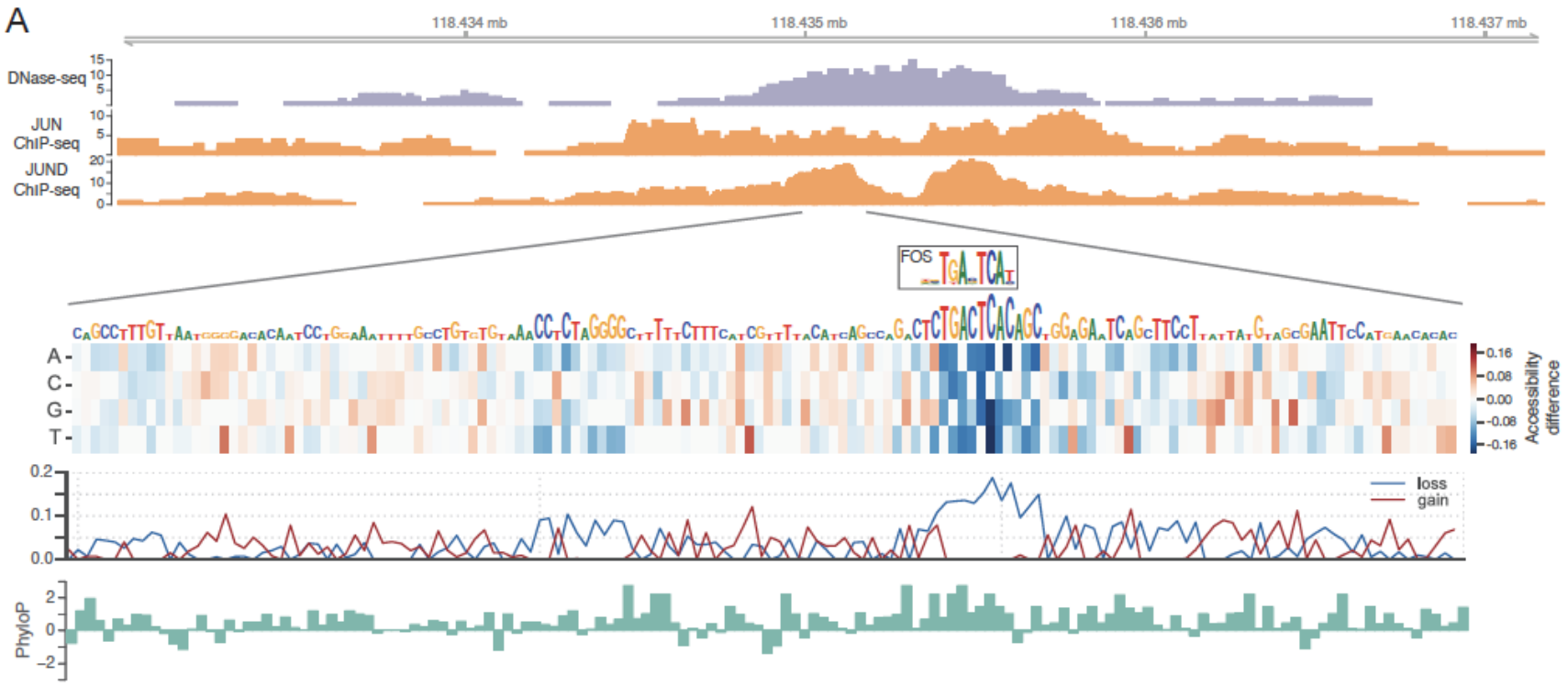


Motifs created by clustering matching input sequences and computing PWM

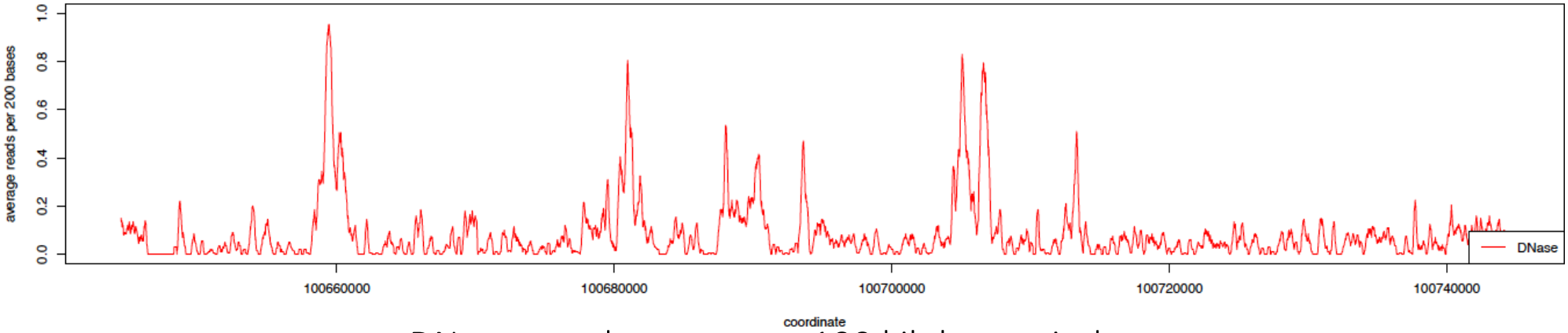
Motif derived from filters with more information tend to be annotated



Computational saturation mutagenesis of an AP-1 site reveals loss of accessibility



Can we predict chromatin accessibility directly from DNA sequence?



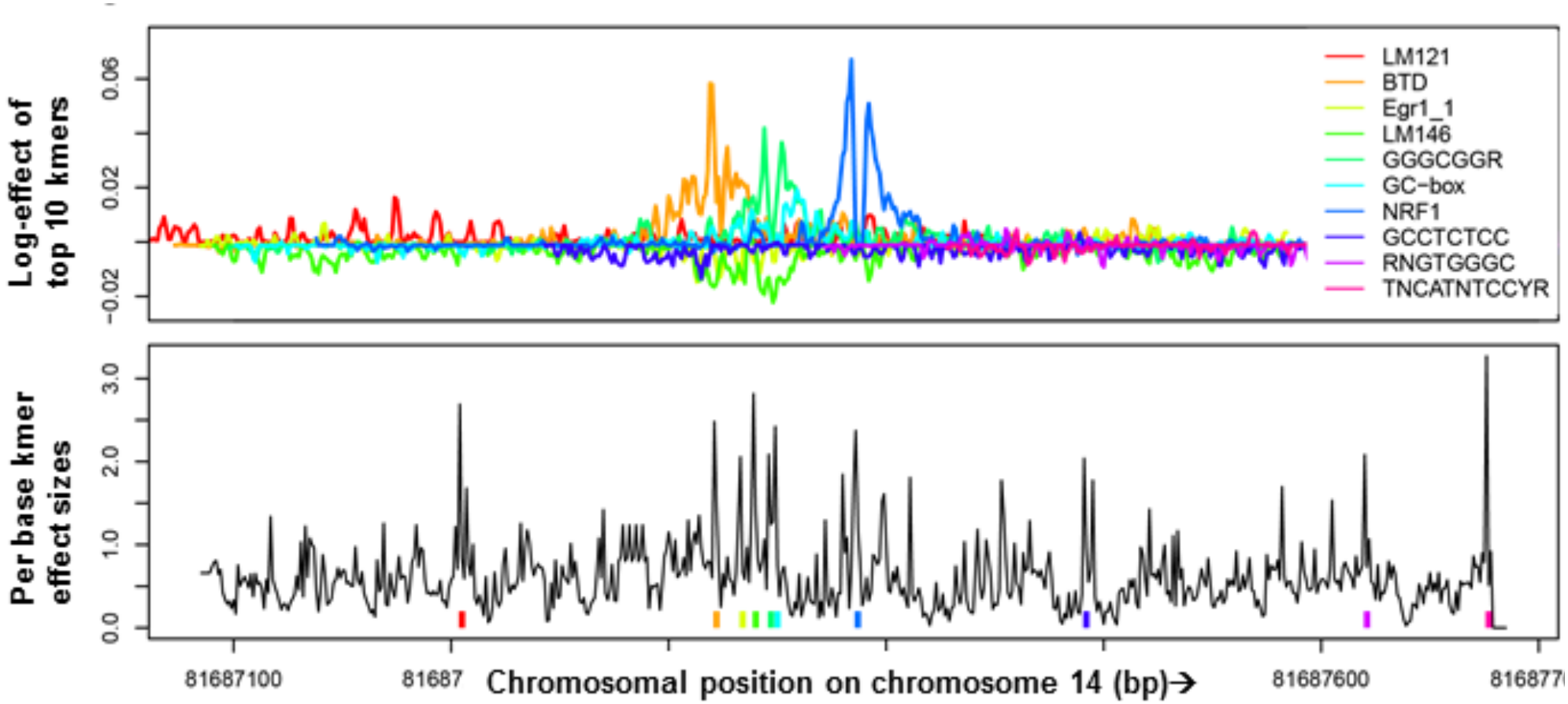
DNase-seq data across a 100 kilobase window
(Chromosome 14 K562 cells)

Hashimoto TB, et al. "**A Synergistic DNA Logic Predicts Genome-wide Chromatin Accessibility"** *Genome Research* 2016

Can we discover DNA “code words” encoding chromatin accessibility?

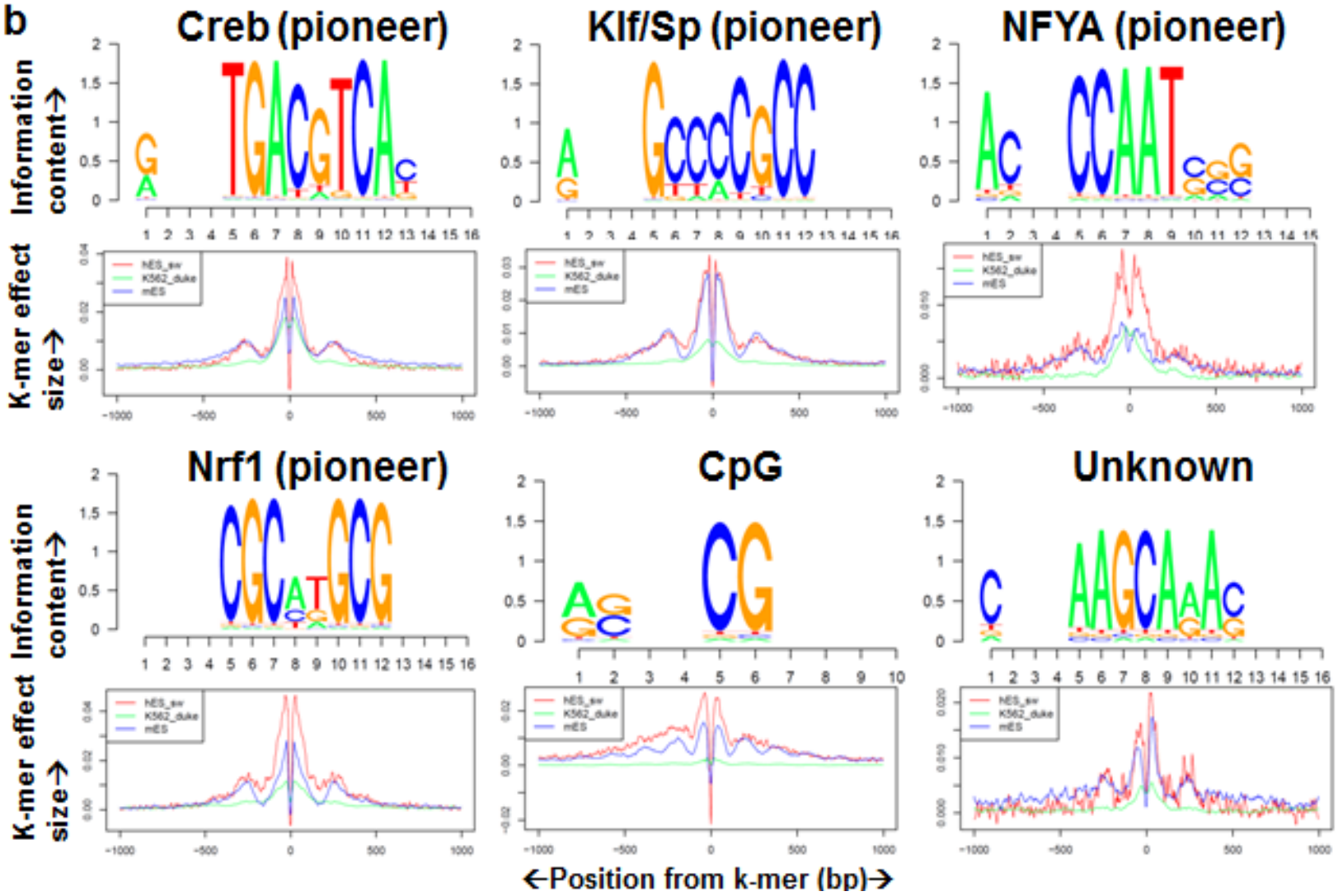
- The DNA “code words” encoding chromatin accessibility can be represented by k-mers (k <= 8)
- K-mers affect chromatin accessibility locally within +/- 1 kb with a fixed spatial profile
- A particular k-mer produces the same effect wherever it occurs

The Synergistic Chromatin Model (SCM) is a K-mer model

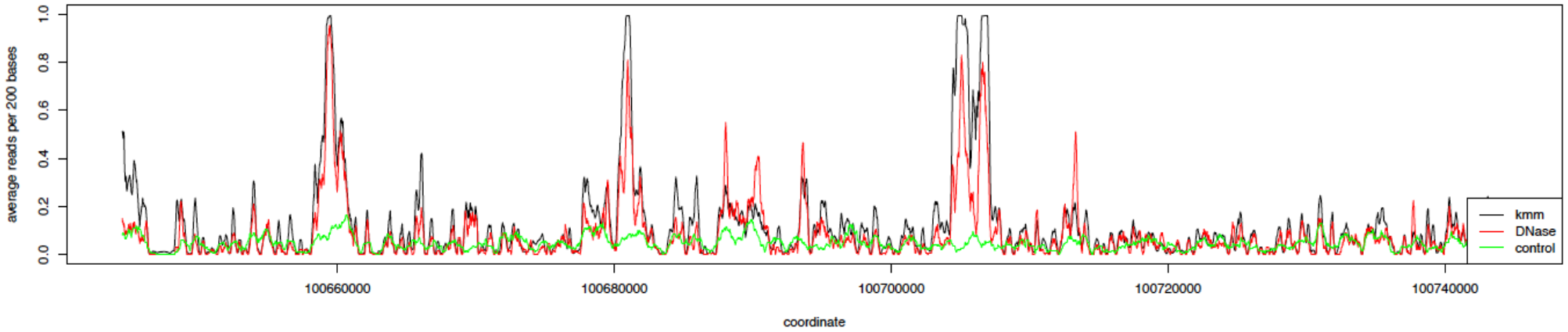


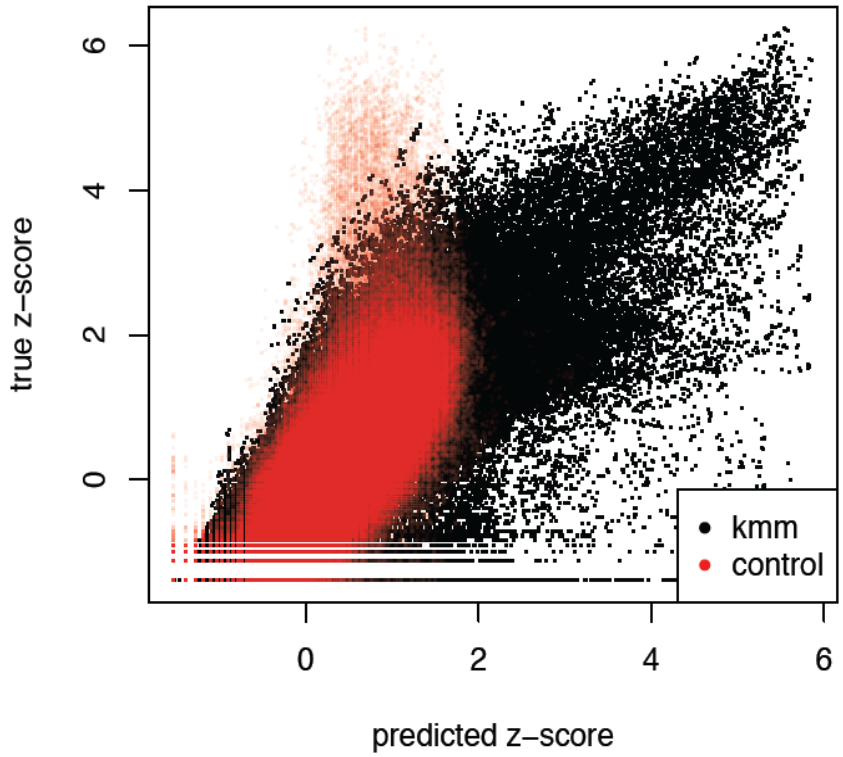
~40,000 K-mers in model
~5,000,000 parameters
543 iterations * 360 seconds / iteration * 40 cores
= ~ 90 days

Chromatin accessibility arises from interactions, largely among pioneer TFs



Training on K562 DNase-seq data from chromosomes 1 – 13 predicts chromosome 14 (black line)

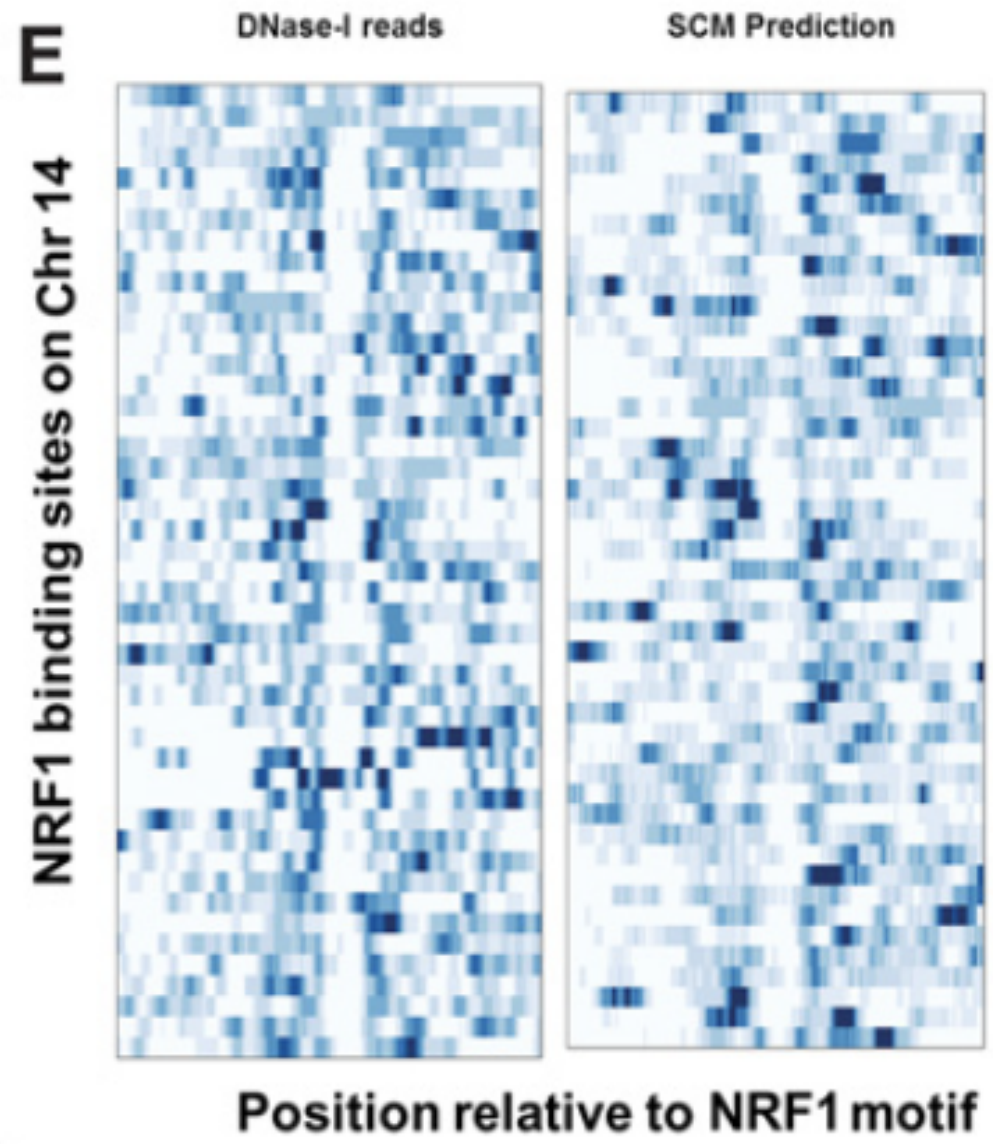




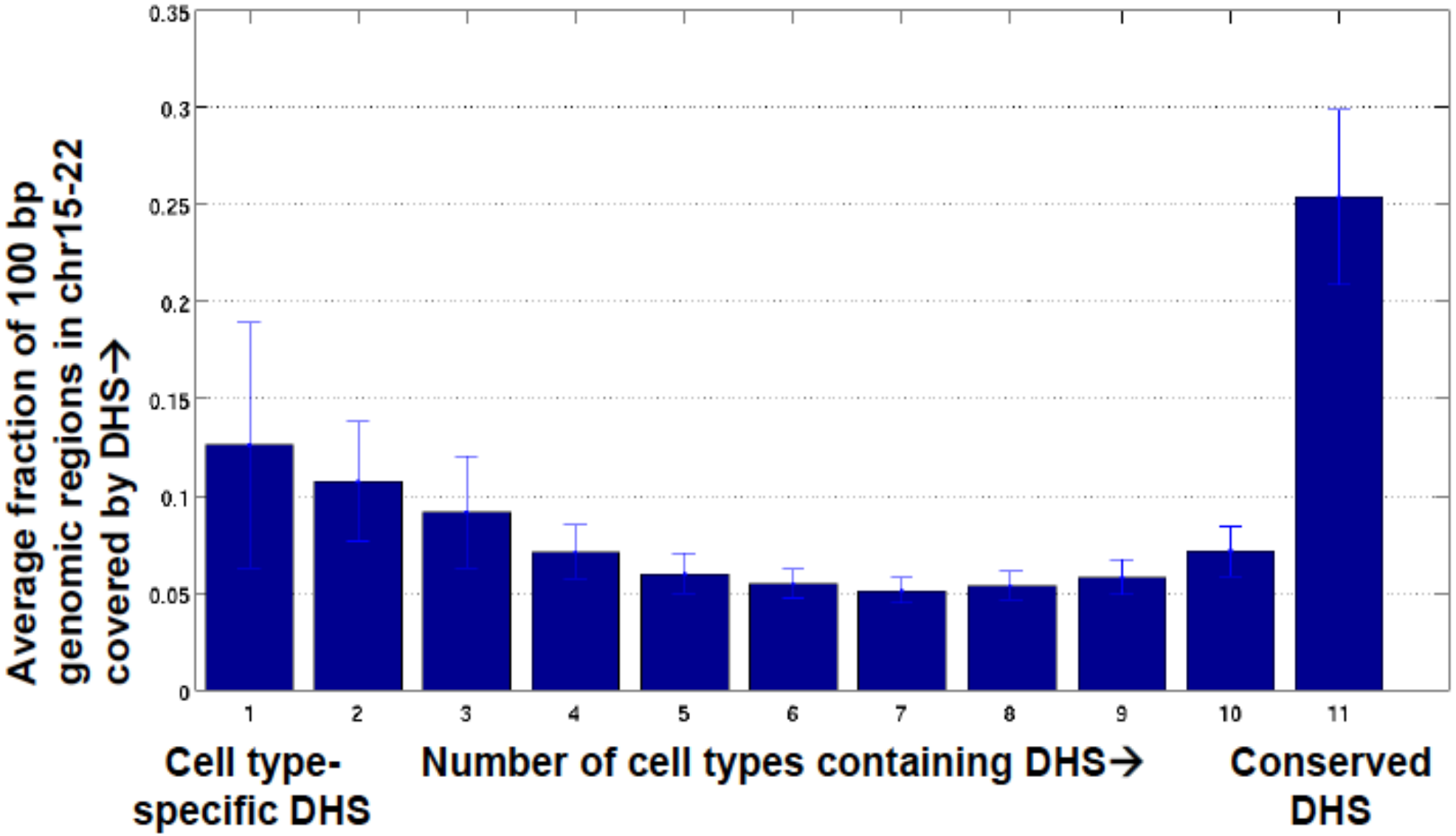
KMM R^2 0.80

Control R^2 0.47

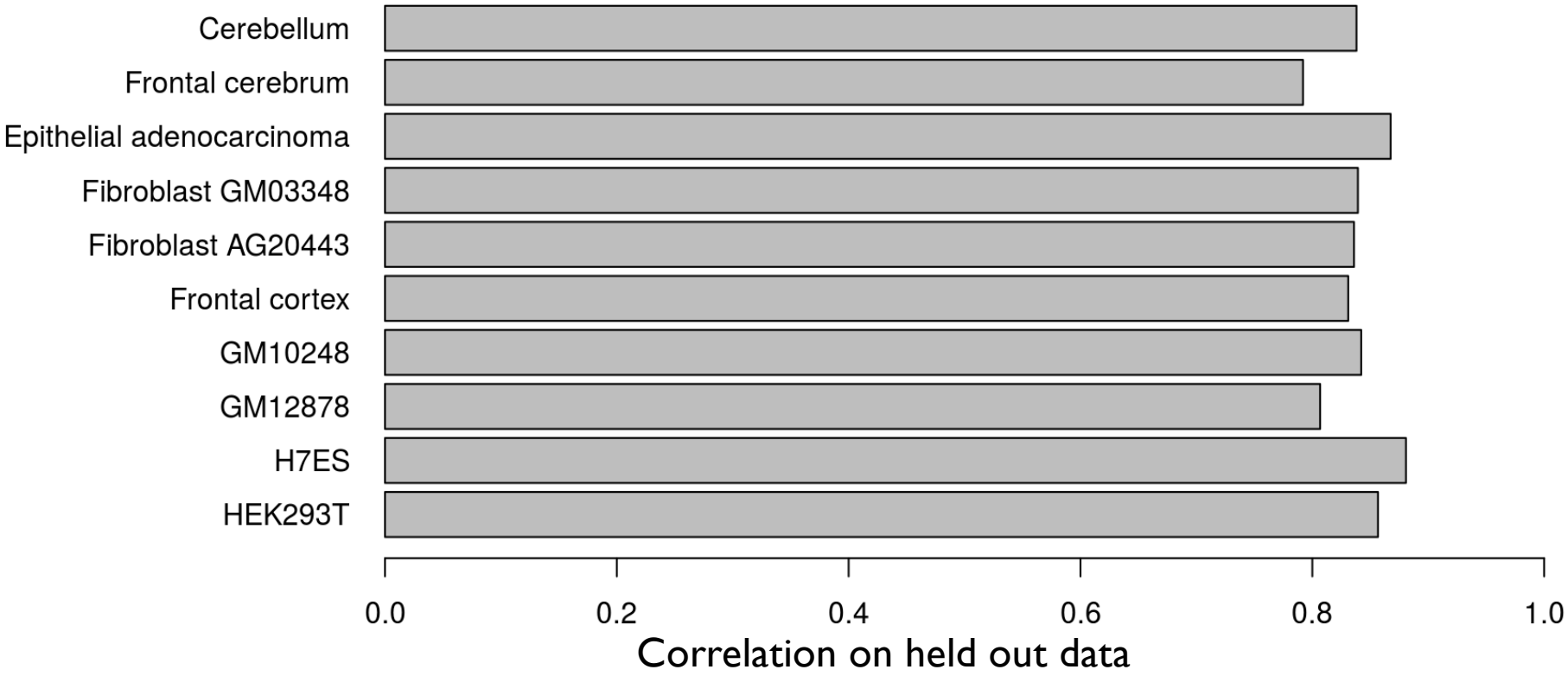
SCM predicts accessibility data from a NRF1 binding site



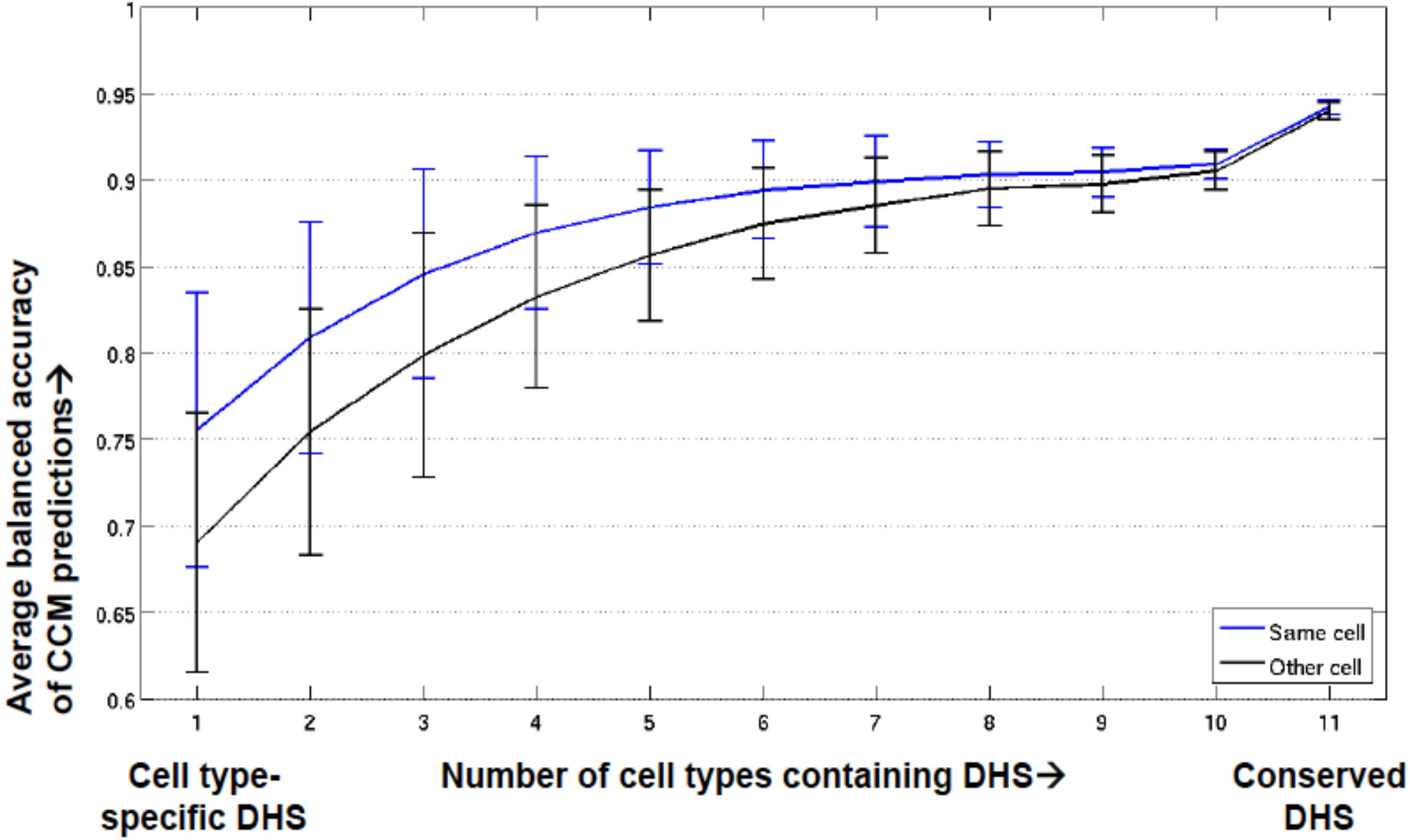
Accessibility contains cell type specific and cell type independent components (11 cell types, Chr 15-22)



SCM models have similar predictive power for other cell types

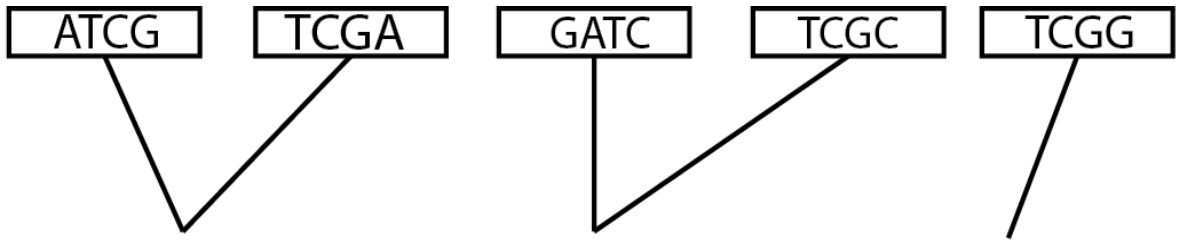


SCM model trained on ES data performs better on shared DNase hot spots (Chr 15 – 22)

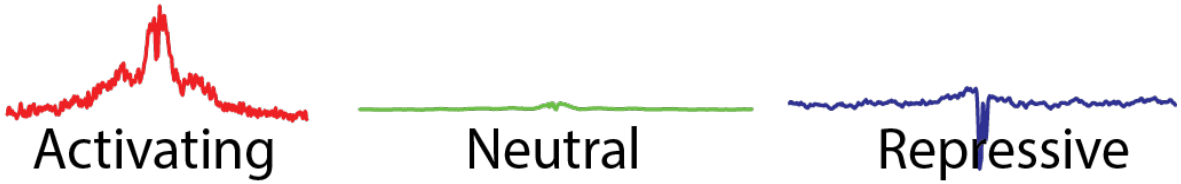


We created synthetic “phrases” each of which contains k-mers that are similar in chromatin opening score

All K-mers



Sort into classes

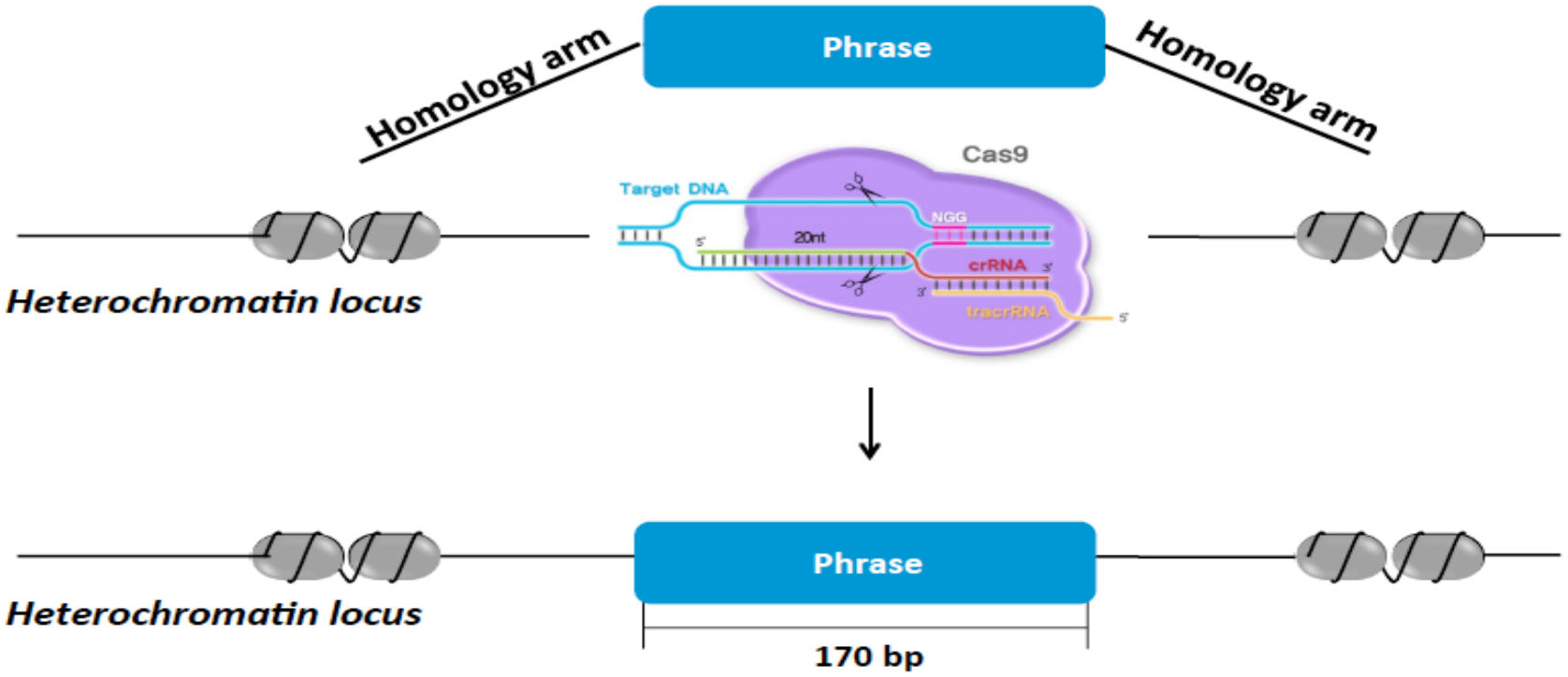


Construct Debrujin Graph

Biased random walk

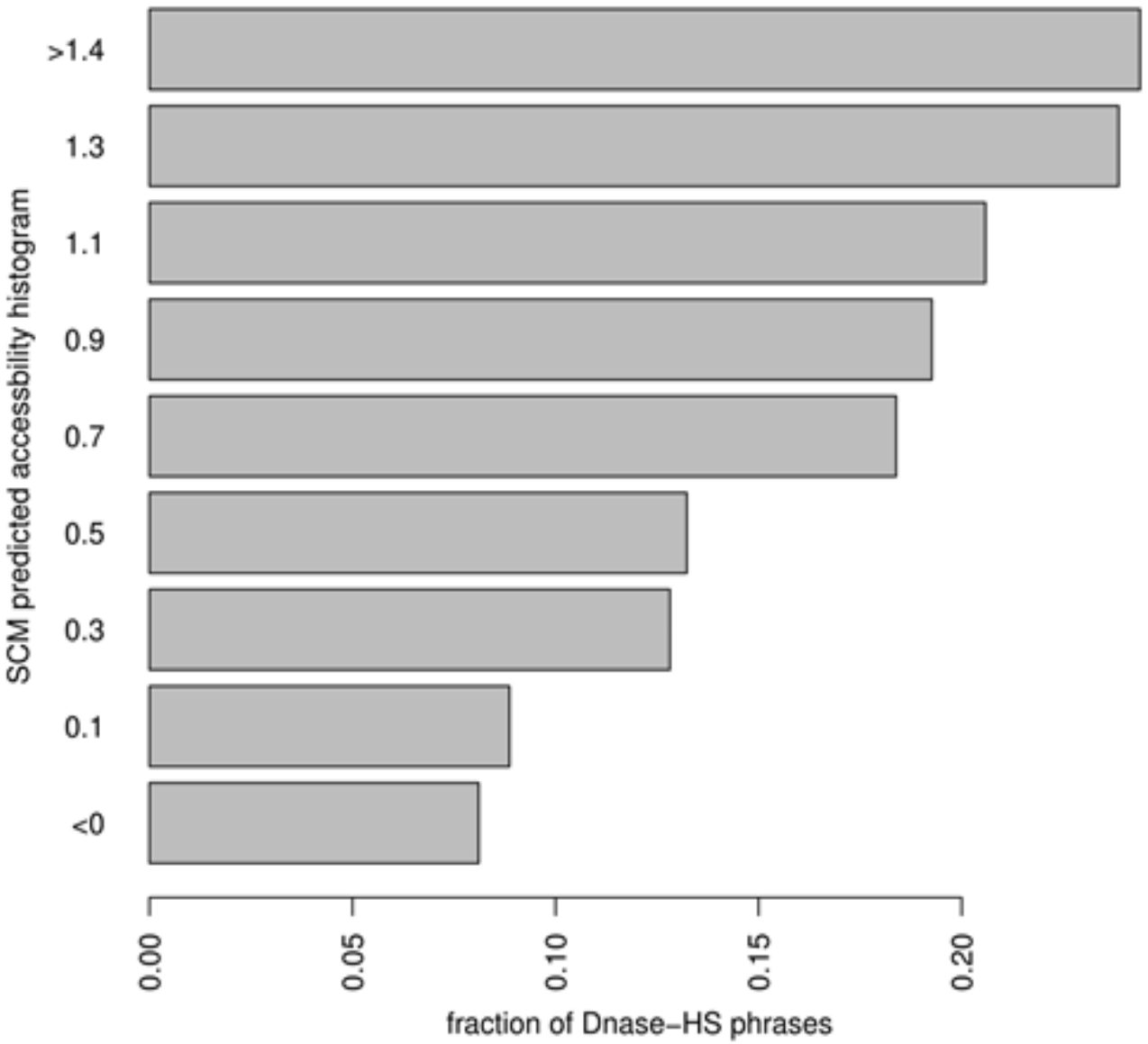
1. GATCGC
2. GATCGA

Single Locus Oligonucleotide Transfer
>6,000 designed phrases into a chromosomal locus



Heterochromatin locus	A	B	C
% alleles with phrase integration	35	15	15
# unique integrations	350,000	150,000	150,000

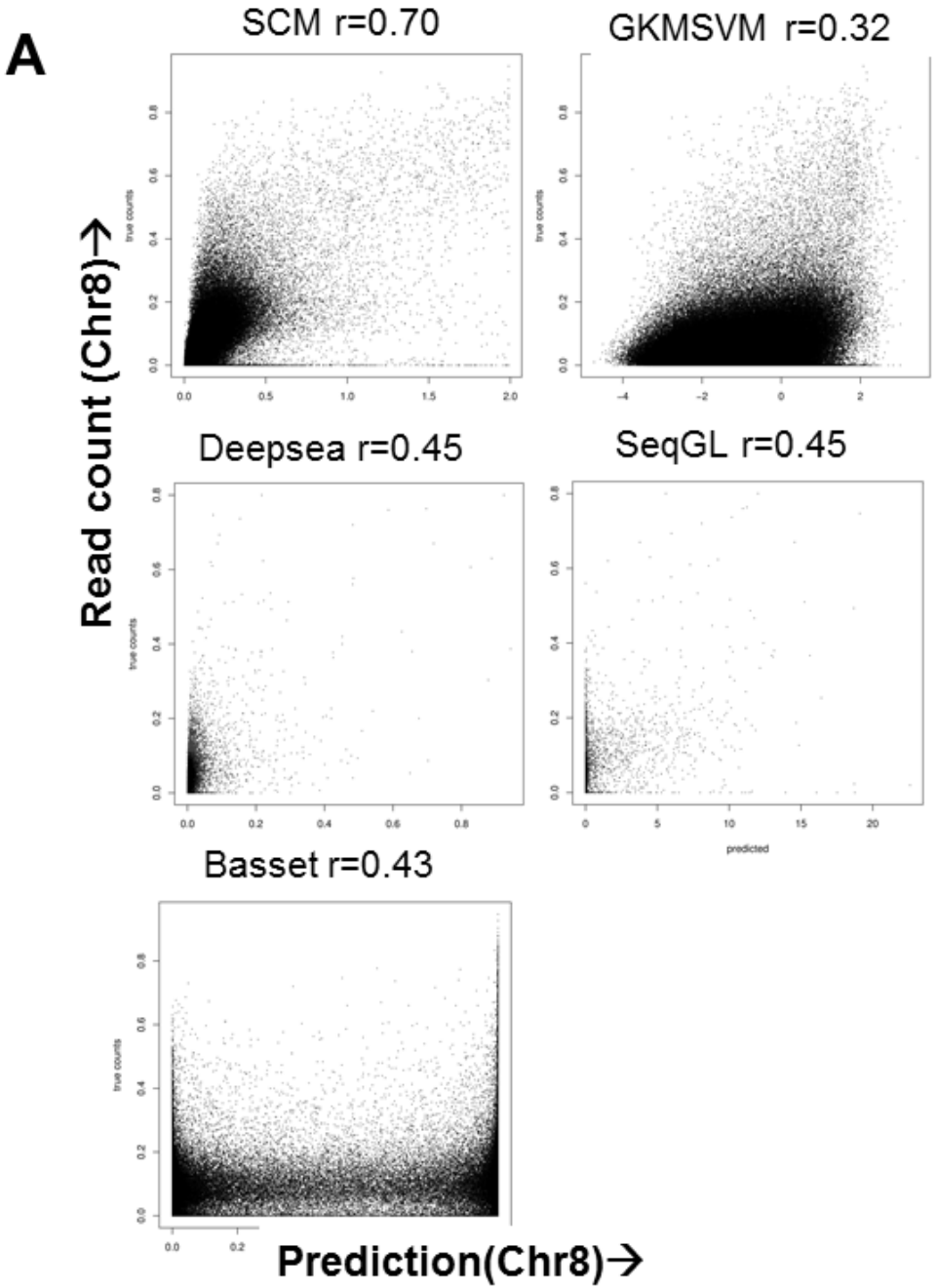
Predicted accessibility matches measured accessibility

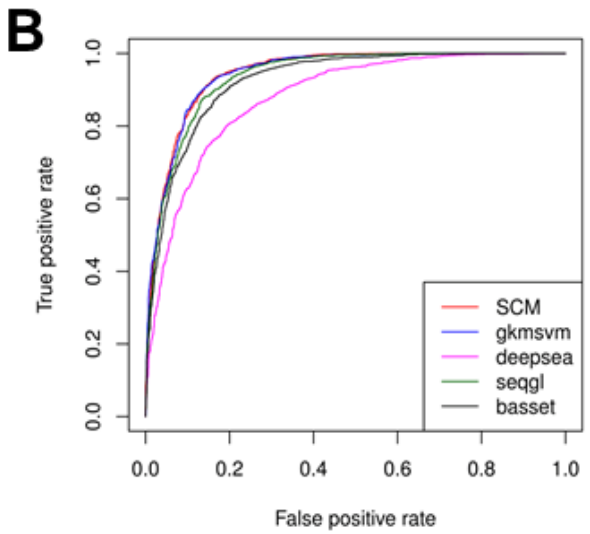


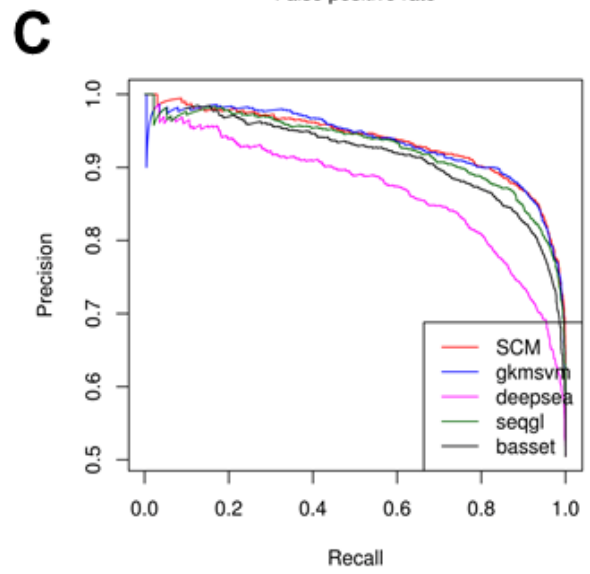
Which is the better model?

- SCM
  - 1bp resolution
  - Regression model – predicts observed read counts
  - Different model per cell type
  - Interpretable effect profile for each unique k-mer that it finds significant (up to 40,000)
- Bassett
  - 600bp resolution
  - Classification model– “open” or ”closed”
  - 168 experiments with one model
  - 300 filters maximum

SCM outperforms contemporary models at predicting chromatin accessibility from sequence (K562)







Making models estimate their uncertainty

What's on tap today!

- The prediction of uncertainty an its importance
  - Aleatoric – inherent observational noise
  - Epistemic – model uncertainty
- How to predict uncertainty
  - Gaussian Processes
  - Ensembles
- Using uncertainty
  - Bayesian optimization
  - Experiment Design

Uncertainty estimates identify where a model should not be trusted

- In self-driving cars, if model is uncertain about predictions from visual data, other sensors may be used to improve situational awareness
- In healthcare, if an AI system is uncertain about a decision, one may want to transfer control to a human doctor
- If a model is very sure about a particular drug helping with a condition and less sure about others, you want to go with the first drug

Model uncertainty enables experiment planning

- High model uncertainty for an input can identify out of training distribution test examples (“out of distribution” input).
- Experiment planning can use uncertainty metrics to design new experiments and observations to fill in training data gaps to improve predictive performance

An example of experiment design

- We a model f of the binding of a transcription factor to 8-mer DNA sequences.
  - Binding = f(8-mer sequence)
  - We train f on: $\{ (s_1, b_1), (s_2, b_2) \ldots (s_n, b_n) \}$
  - Goal is to discover s_{best} = argmax f(s)
  - Need excellent model for f but we have not observed binding for all sequences
  - What the next sequence s_x we should ask to observe?
- What is a principled way to choose s_x ?

Experiment design explores the space where a model is uncertain

- *Explore* the space more to *improve* your model as well (in addition to exploiting existing guesses)
- You want to explore the space where your model is not confident about being right – hence uncertainty quantification.
- We can quantify uncertainty with probability for discrete outputs or a standard deviation for continuous outputs
  - P(label | features) — Classification
  - (μ, σ^2) = f(input) — Regression – Normal distribution parameters

One metric of uncertainty for a given input is entropy for categorical labels

- Suppose we have a multiclass classification problem
- We already have an indication of uncertainty as the model directly outputs class probability
- Intuitively, the more uniformly distributed the predicted probability over the different classes, the more uncertain the prediction
- Formally we can use *information entropy* to quantify uncertainty

$$S = -\sum_i P_i \log P_i$$

There are two types of uncertainty

- Aleatoric (experimental) uncertainty
- Epistemic (model) uncertainty

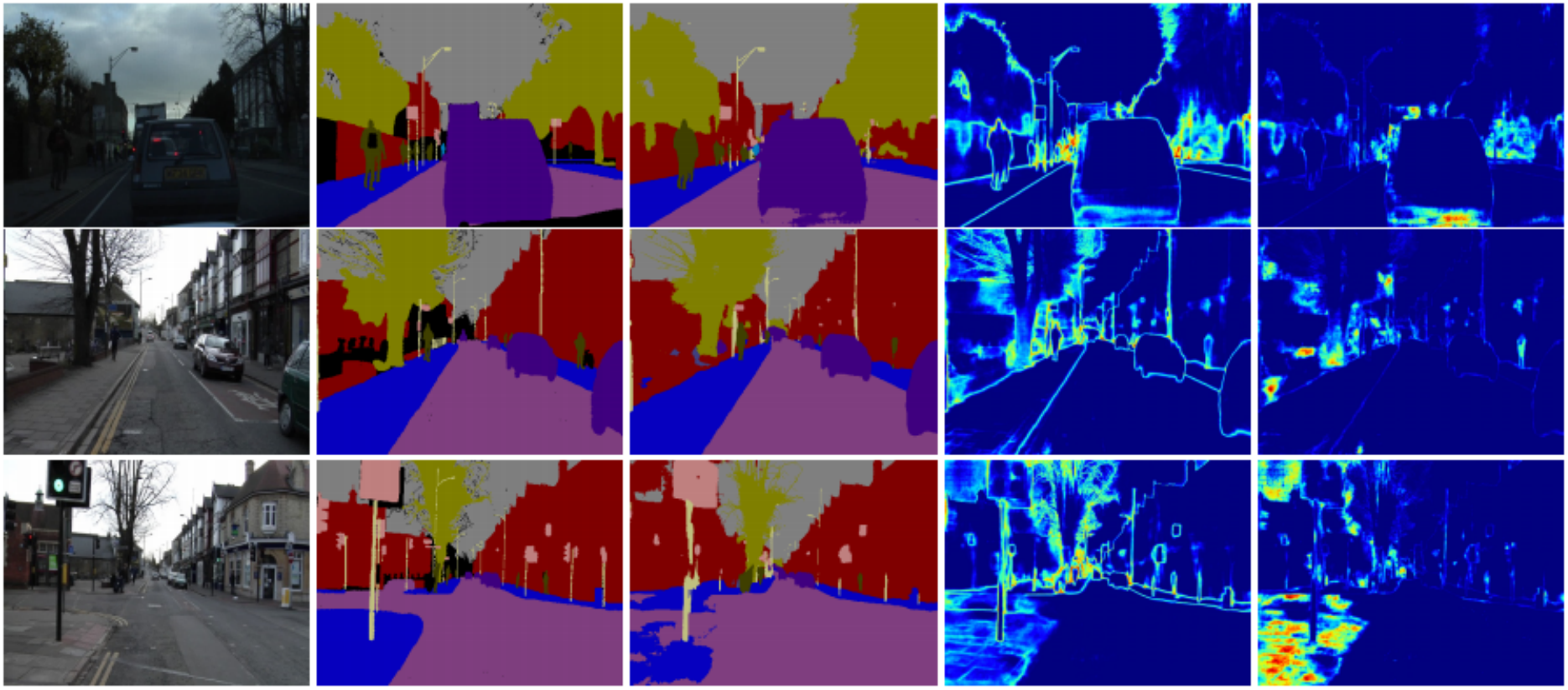
Aleatoric (experimental) uncertainty

- Examples
  - Human error in labeling image categories
  - Noise in biological systems – TF binding to DNA is stochastic
- Source is the *unmeasured* unknowns that can change every time we repeat an experiment
- More training data can better calibrate this noise, not eliminate it

Epistemic (model) uncertainty

- Examples
  - Different hypothesis for why sun moves in the sky (geocentric vs heliocentric)
  - Uncertainty about which features to use in a model
  - Uncertainty about the best model architecture (number of filters, depth of network, number of internal nodes)
- Epistemic uncertainty results from different models that fit the training data equally well but generalize differently
- More training data can reduce epistemic uncertainty

In vision aleatoric uncertainty is seen at edges; epistemic in objects



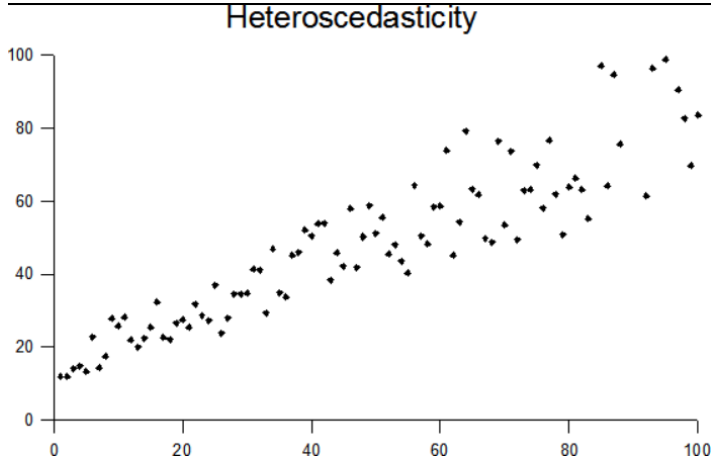
(a) Input Image (b) Ground Truth (c) Semantic Segmentation (d) Aleatoric Uncertainty (e) Epistemic Uncertainty

For (d), (e), Dark blue is lower uncertainty, lighter blue is higher uncertainty, and yellow -> red is the highest uncertainty

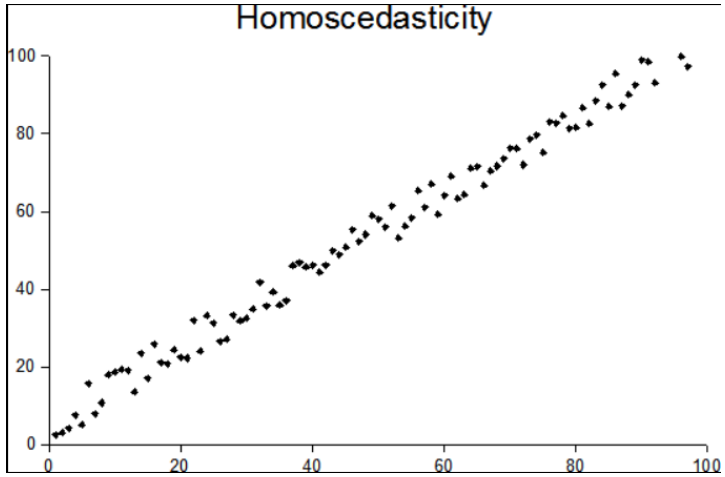
Modeling aleatoric uncertainty

Aleatoric uncertainty can be constant or change with the label value

- Heteroscedastic noise
  - Changes with the feature value



- Homoscedastic noise
  - Does not change with the feature value



Modeling aleatoric uncertainty

$$y = f(x) + \epsilon$$

- Homoscedastic noise

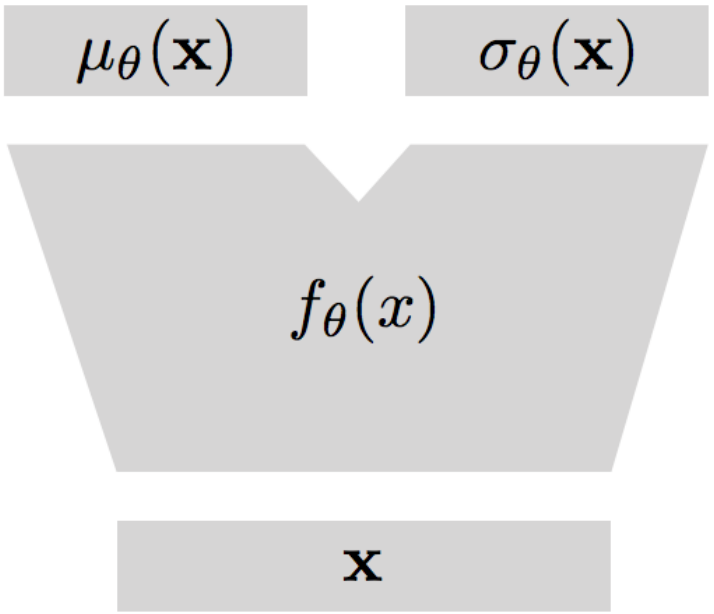
$$\epsilon \sim N(0, 1)$$

- Heteroscedastic noise

$$\epsilon \sim N(0, g(x))$$

- Other popular noise distributions – Poisson, Laplace, Negative Binomial, Gamma, etc.

A “two headed” network can predict aleatoric uncertainty



$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log \mathcal{N}(y_i; \mu_\theta(x_i), \sigma_\theta^2(x_i))$$

$$\mathcal{L}_{BNN}(\theta) = \frac{1}{D}\sum_i \frac{1}{2}\exp(-s_i)||\mathbf{y}_i - \hat{\mathbf{y}}_i||^2 + \frac{1}{2}s_i.$$

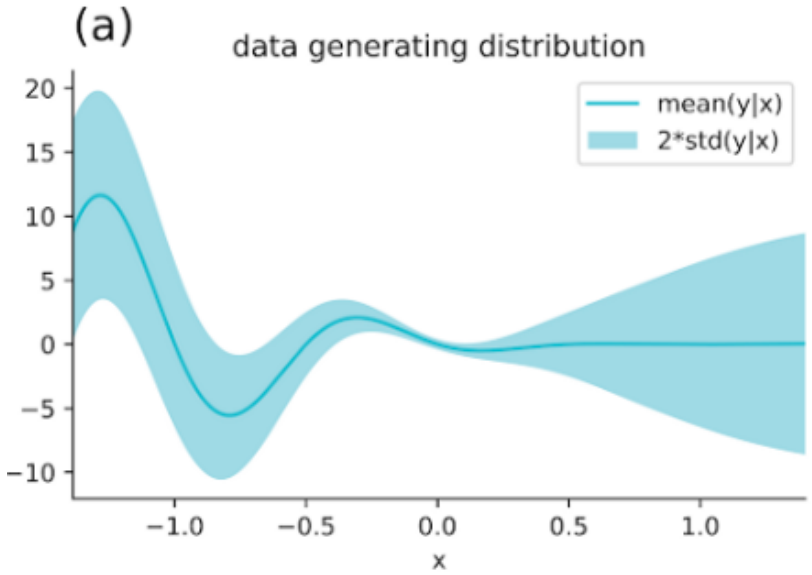
Predict $s_i = \log(\sigma^2)$ to avoid divide by zero issues

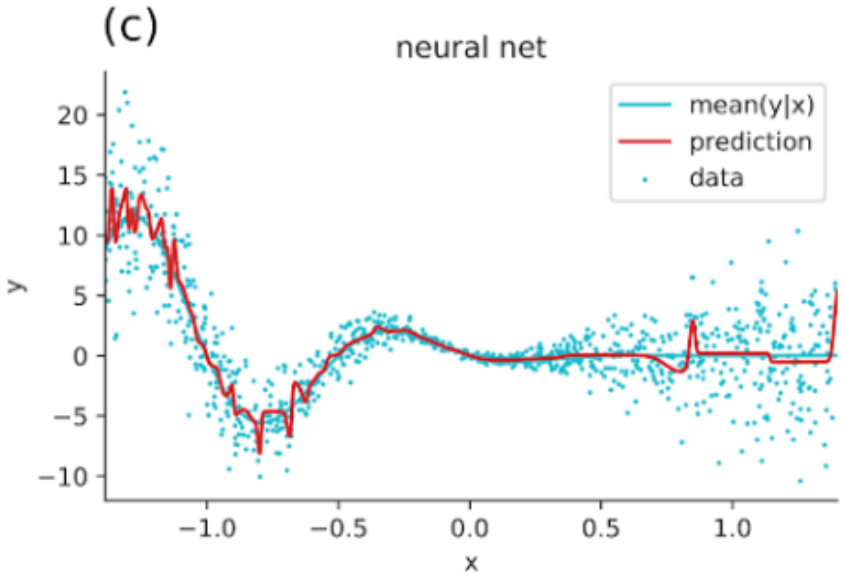
Confidence intervals

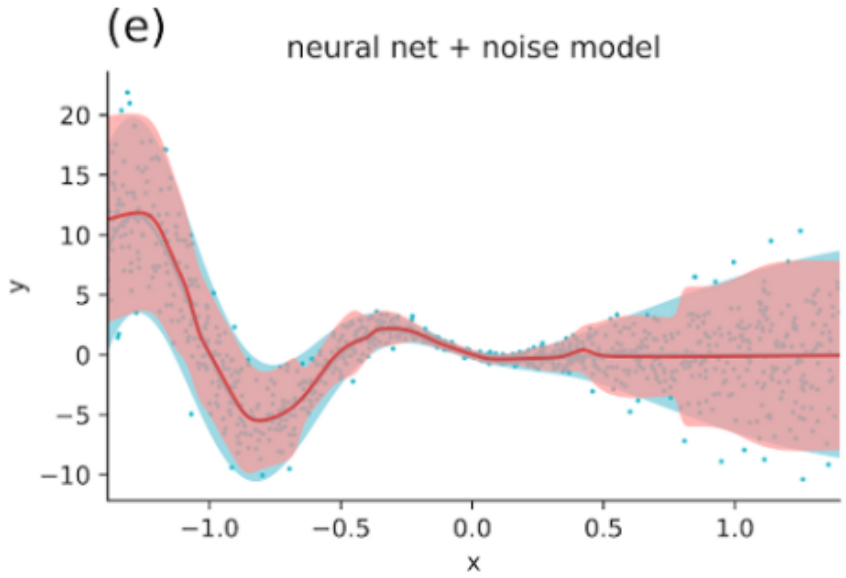
- Intuitively, an interval around the prediction that could contain the true label.

- An X% confidence interval means that for independent and identically distributed (IID) data, X% of the future samples will fall within the interval.

Visualizing uncertainty quantification



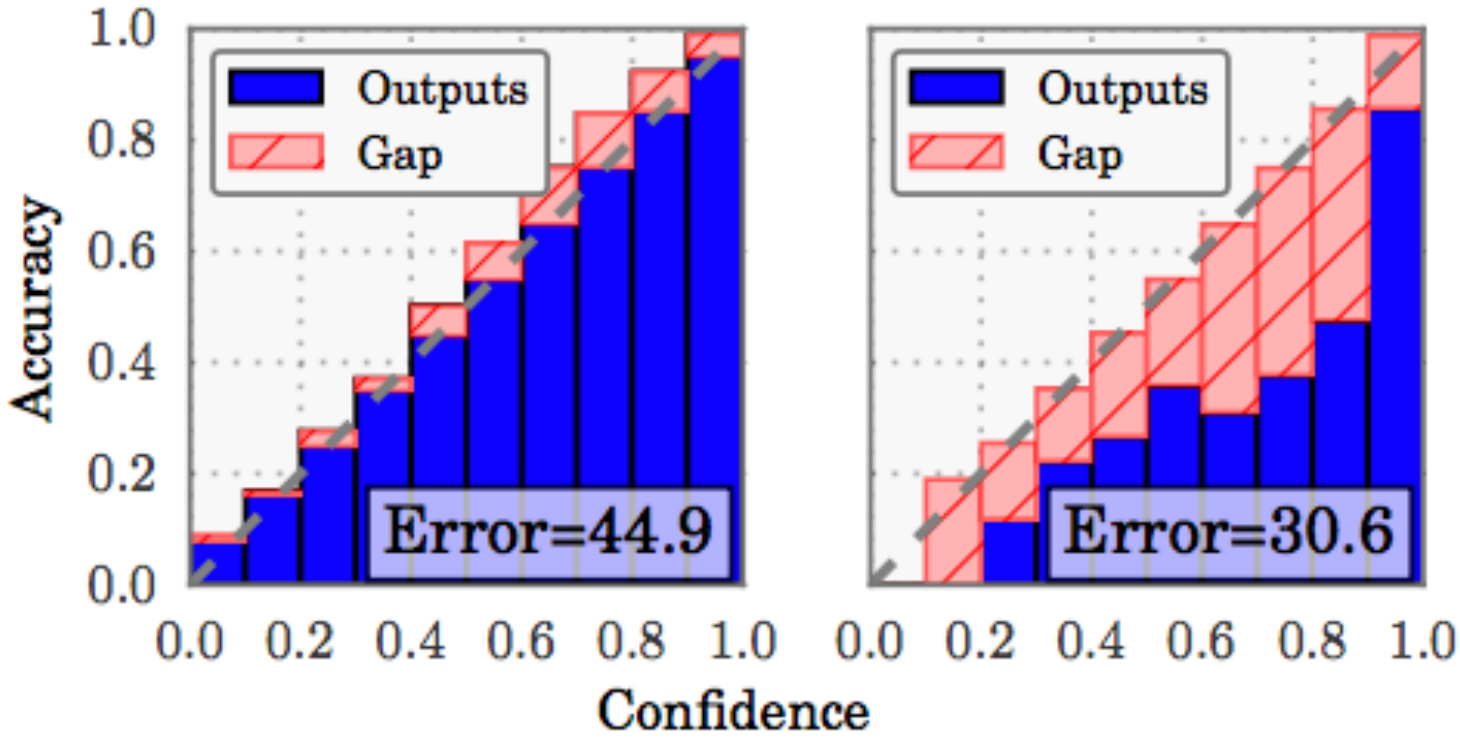




https://medium.com/capital-one-tech/reasonable-doubt-get-onto-the-top-35-mnist-leaderboard-by-quantifying-aleatoric-uncertainty-a8503f134497

A *well-calibrated* model produces uncertainty predictions that match held out data

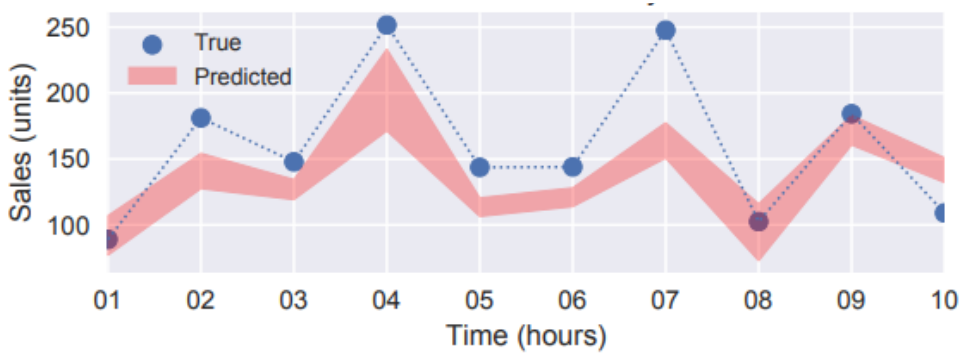
- Classification
  - If we only look at predictions where the probability of a class is 0.3, they should be correct 30% of the time

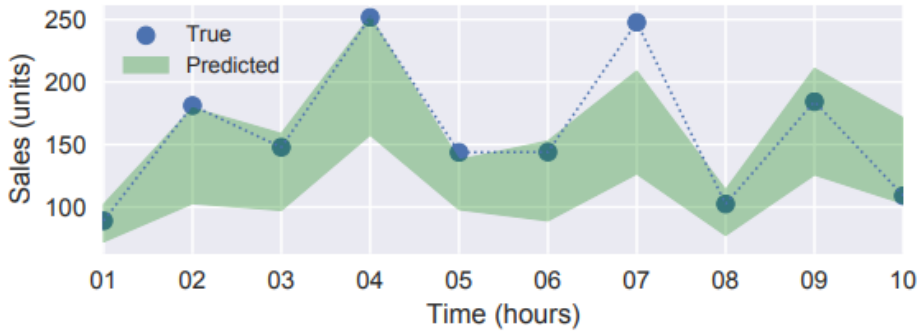


Error indicates the overall network accuracy

A *well-calibrated* model produces uncertainty predictions that match held out data

- Regression
  - Compute *confidence intervals* for each input
  - For inputs with a *confidence interval* of 90% then 90% of predictions should fall within the interval





Overfit models can have uncalibrated uncertainty

- Recall that the loss function includes both accuracy and uncertainty terms
- Once a model gets close to 100% accuracy on predicting mean values the models are incentivized to reduce their uncertainty

$$\mathcal{L}_{BNN}(\theta) = \frac{1}{D}\sum_{i}\frac{1}{2}\exp(-s_i)||\mathbf{y}_i - \hat{\mathbf{y}}_i||^2 + \frac{1}{2}s_i.$$

Recalibration

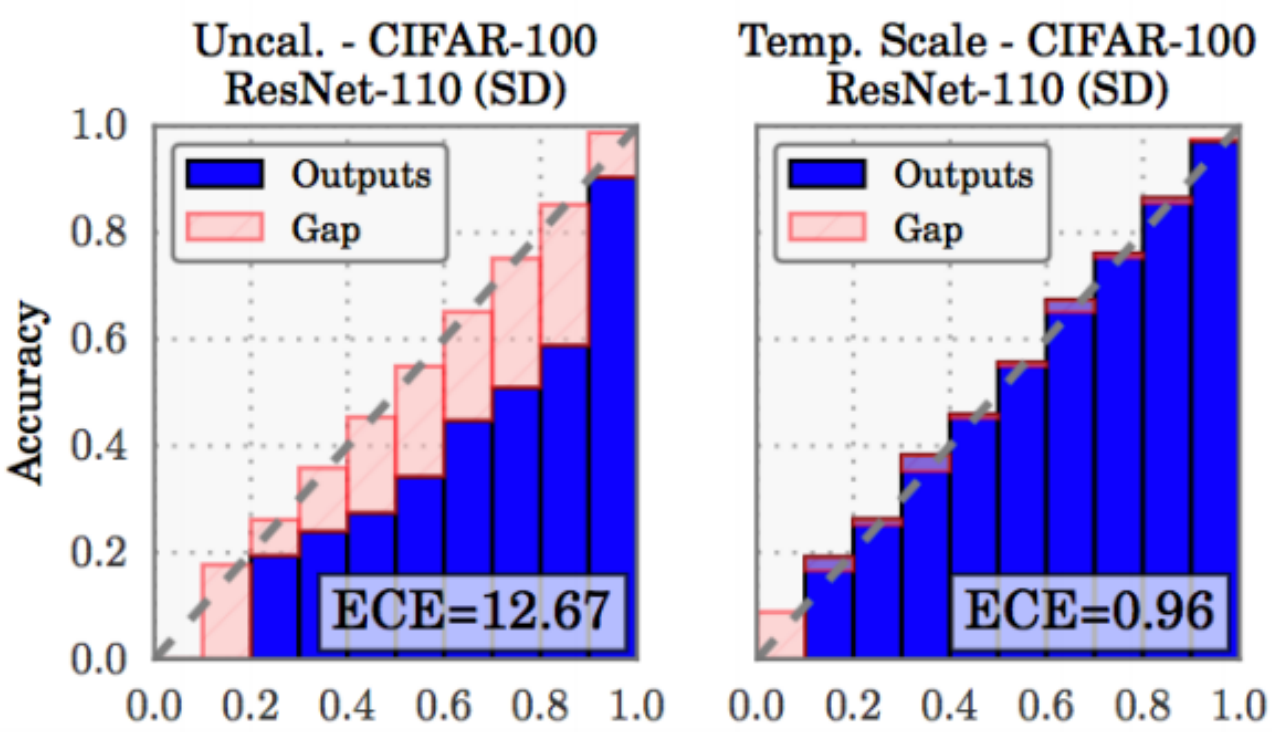
- Temperature scaling is a simple idea to recalibrate softmax based classifiers
- We scale the logits used for the softmax function by a scalar $1/T$
- As $T \Rightarrow \infty$ the probabilities come closer together
- Thus temperature scaling increases model uncertainty when $T > 1$
- Note that temperature scaling does not change the most probable class

Temperature scaling

$$\phi_{\text{softmax}}(\boldsymbol{z})_i = \frac{e^{z_i/T}}{\sum_{j=1}^{|\boldsymbol{z}|} e^{z_j/T}}$$

Recalibration

- ECE – Expected calibration error – area between calibration curve (line formed by blue histograms) and diagonal
  - Before and after temperature scaling:

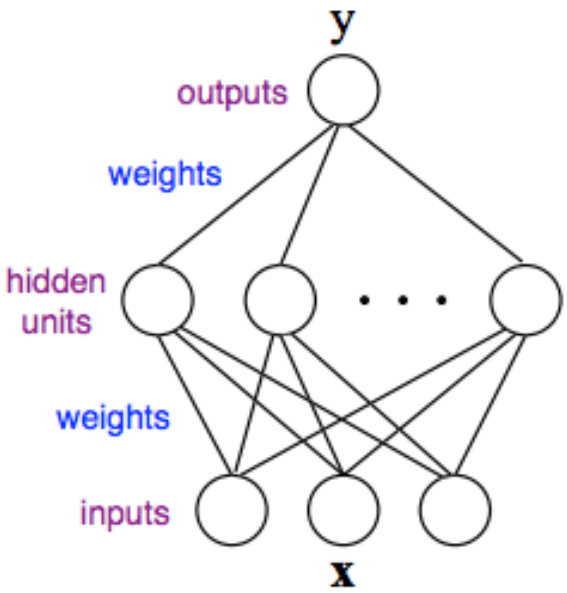


Modeling epistemic uncertainty

Modeling epistemic uncertainty

- Define a space of models – called a *hypothesis* space and assign probabilities to each model
- *Bayesian* modeling is a principled way to assign probabilities to models in a hypothesis space
- *Ensembles* sample different models
- *Dropout* samples different models
- *Gaussian processes* represent many different models

Uncertainty can be produced in a single network by making parameters uncertain – Bayesian Neural Nets



Bayesian neural network

Data: $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N} = (X, \mathbf{y})$

Parameters $\boldsymbol{\theta}$ are weights of neural net

prior $p(\boldsymbol{\theta}|\boldsymbol{\alpha})$

posterior $p(\boldsymbol{\theta}|\boldsymbol{\alpha}, \mathcal{D}) \propto p(\mathbf{y}|X, \boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\alpha})$

prediction $p(y'|\mathcal{D}, \mathbf{x}', \boldsymbol{\alpha}) = \int p(y'|\mathbf{x}', \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\alpha})\, d\boldsymbol{\theta}$

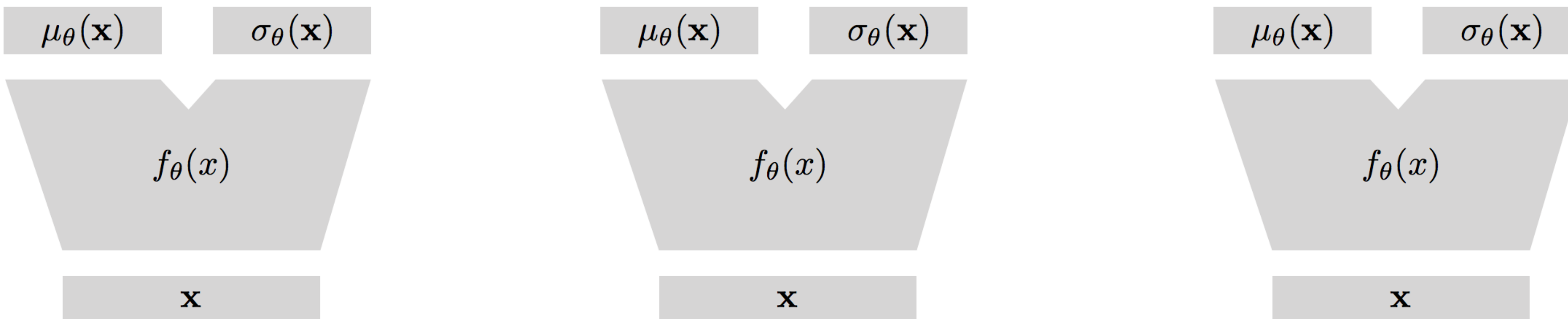
Bayesian NN Advantages/Disadvantages

- Advantages
  - Principled Bayesian approach for deep neural networks
- Disadvantages
  - Tend to be overconfident
  - Common approaches to do inference are expensive
  - While in principle, arbitrary aleatoric noise distributions can be used, in practice, that makes inference even more expensive

Dropout samples different models by randomly dropping nodes

- Randomly drop some fraction of the neurons at prediction time
- Gives an empirical distribution over predictions
- The empirical standard deviation (proportional to entropy in case of Gaussian distribution) is a measure of uncertainty
- Tends to be extremely overconfident

Epistemic uncertainty quantification with an ensemble of different networks or networks trained on different data



$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log \mathcal{N}(y_i; \mu_\theta(x_i), \sigma_\theta^2(x_i))$$

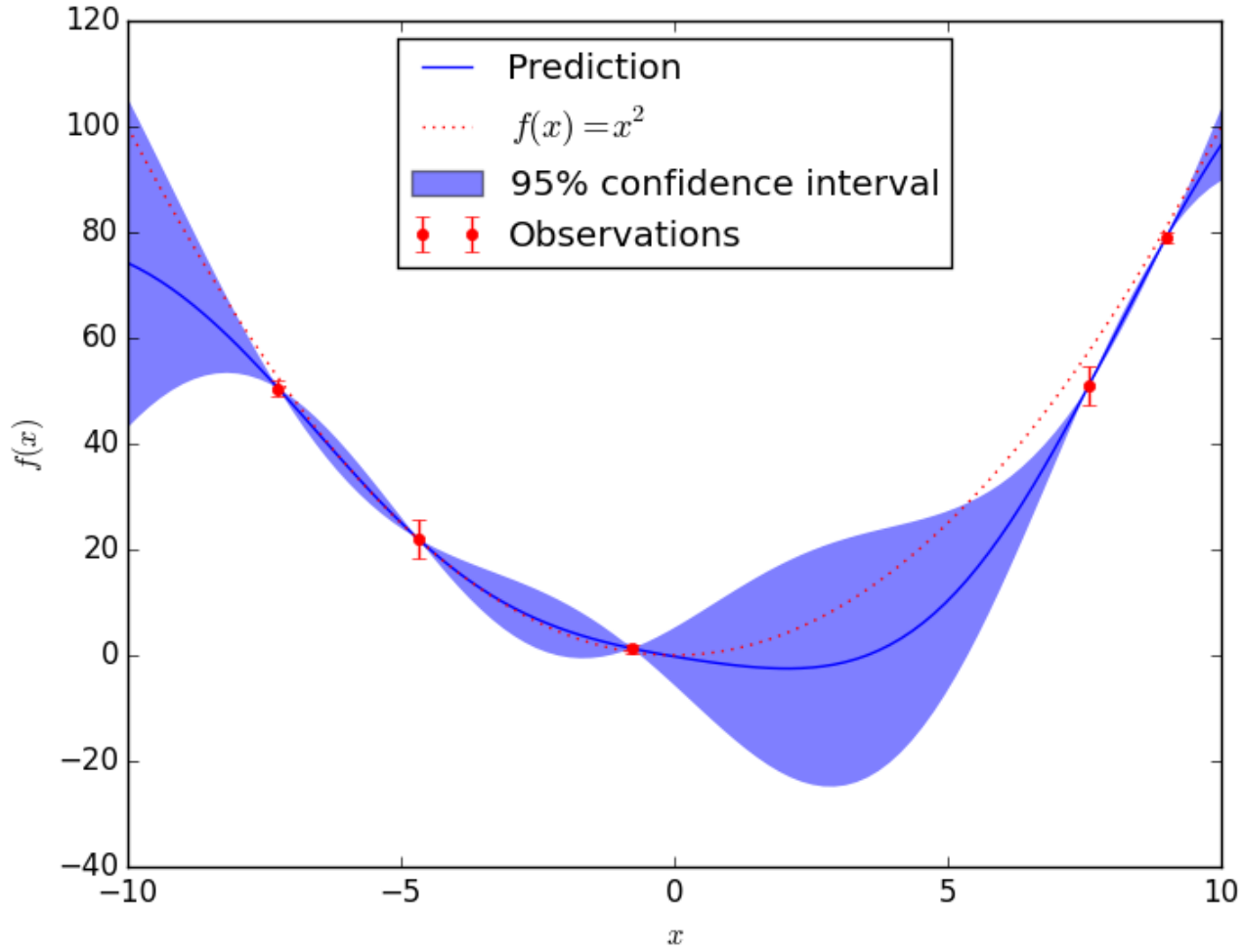
$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log \mathcal{N}(y_i; \mu_\theta(x_i), \sigma_\theta^2(x_i))$$

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log \mathcal{N}(y_i; \mu_\theta(x_i), \sigma_\theta^2(x_i))$$

Epistemic uncertainty - $Var(\mu_{\theta_i}((x)))$

Gaussian processes are predictive models that represent uncertainty with a closed form solution; f* is a set of predictive functions

Prediction f* is represented by a multivariate normal



The squared exponential is a common covariance function

Commonly used covariance function:

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \alpha exp\{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma_1^2}\}$$

Hyperparameters:

- $\sigma_1 =$ Characteristic lengthscale
- $\alpha =$ Signal variance

At the core of a Gaussian Process is the Covariance matrix (Similarity matrix)

Think of it as the *similarity* matrix between two points

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$$

$$\mathsf{K}_{xx} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_1, x_1) & \dots & k(x_2, x_n) \\ \dots & \dots & \dots & \dots \\ k(x_n, x_1) & k(x_n, x_2) & \dots & k(x_n, x_n) \end{bmatrix}$$

Prediction using a Gaussian Process

Given observations X, Y, the posterior distribution of a Gaussian process at a point x is

$$f(x) \sim \mathcal{N}(m(x), \sigma^2(x)) \tag{1}$$

$$m(x) = \mu(x) + k(X,x)^T \left[K + \sigma_n^2 I\right]^{-1} (Y - \mu(x)) \tag{2}$$

$$\sigma^2(x) = k(x,x) - k(X,x)^T \left[K + \sigma_n^2 I\right]^{-1} k(X,x) \tag{3}$$

where $K_{ij} = k(x_i, x_j)$, $k(X,x) = (k(x,x_1), \ldots, k(x,x_n))$, and σ_n^2 is the variance of the noise (assumed additive Gaussian).

Gaussian Processes - Advantages/Disadvantages

- Advantages
  - Closed form for the posterior distribution
  - Can easily adapt to new training data
  - Uncertainties are usually well calibrated
- Disadvantages
  - Scale cubically with the number of training points (though a lot of recent work trying to that down to a linear scaling)
  - Closed form limited to gaussian output noise
  - Can be adapted for classification but not easy to train as there is no closed form either for that case
  - Need a lot of data to cover the entire input space well

Experiment design using uncertainty

An example of experiment design

- We use a model f of the binding of a transcription factor to 8-mer DNA sequences.
  - Binding = f(8-mer sequence)
  - We train f on: $\{ (s_1, b_1), (s_2, b_2) \ldots (s_n, b_n) \}$
  - Goal is to discover s_{best} = argmax f(s)
  - Need excellent model for f but we have not observed binding for all sequences
  - What the next sequence s_x we should ask to observe?
- What is a principled way to choose s_x ?

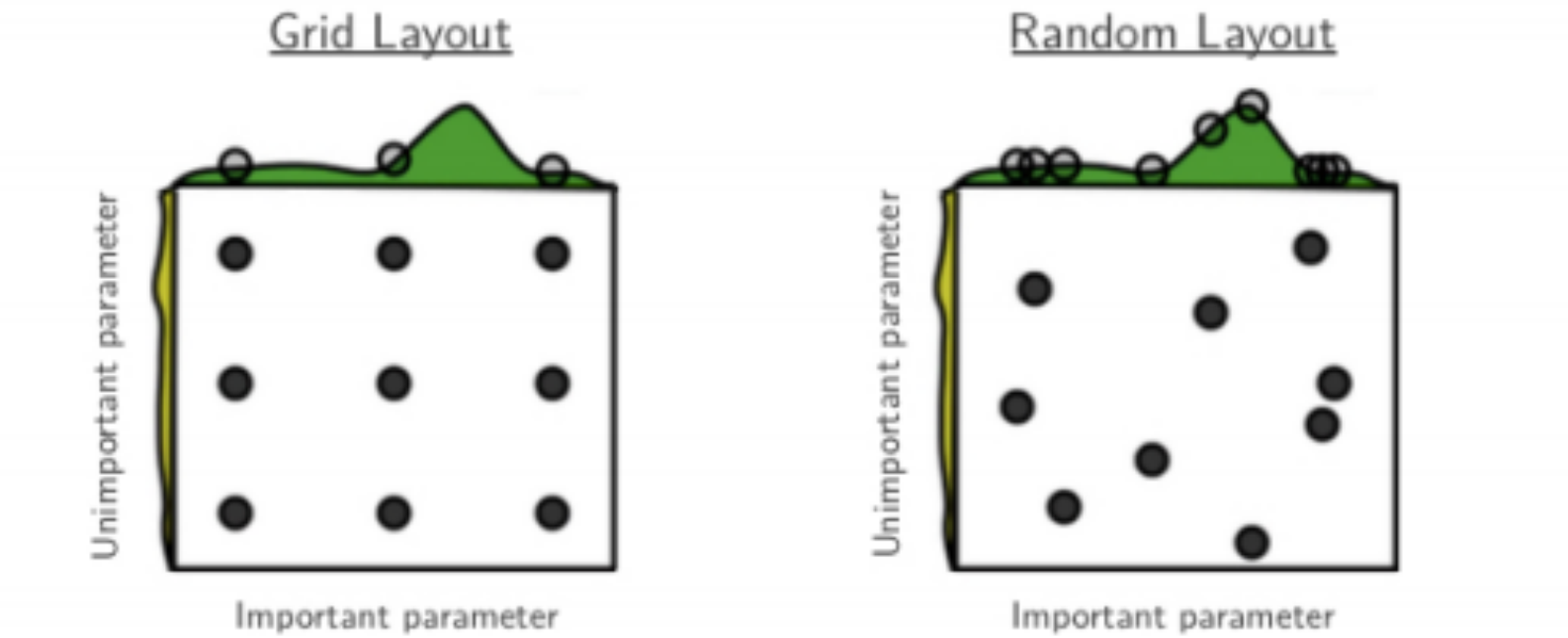
Other example of optimization

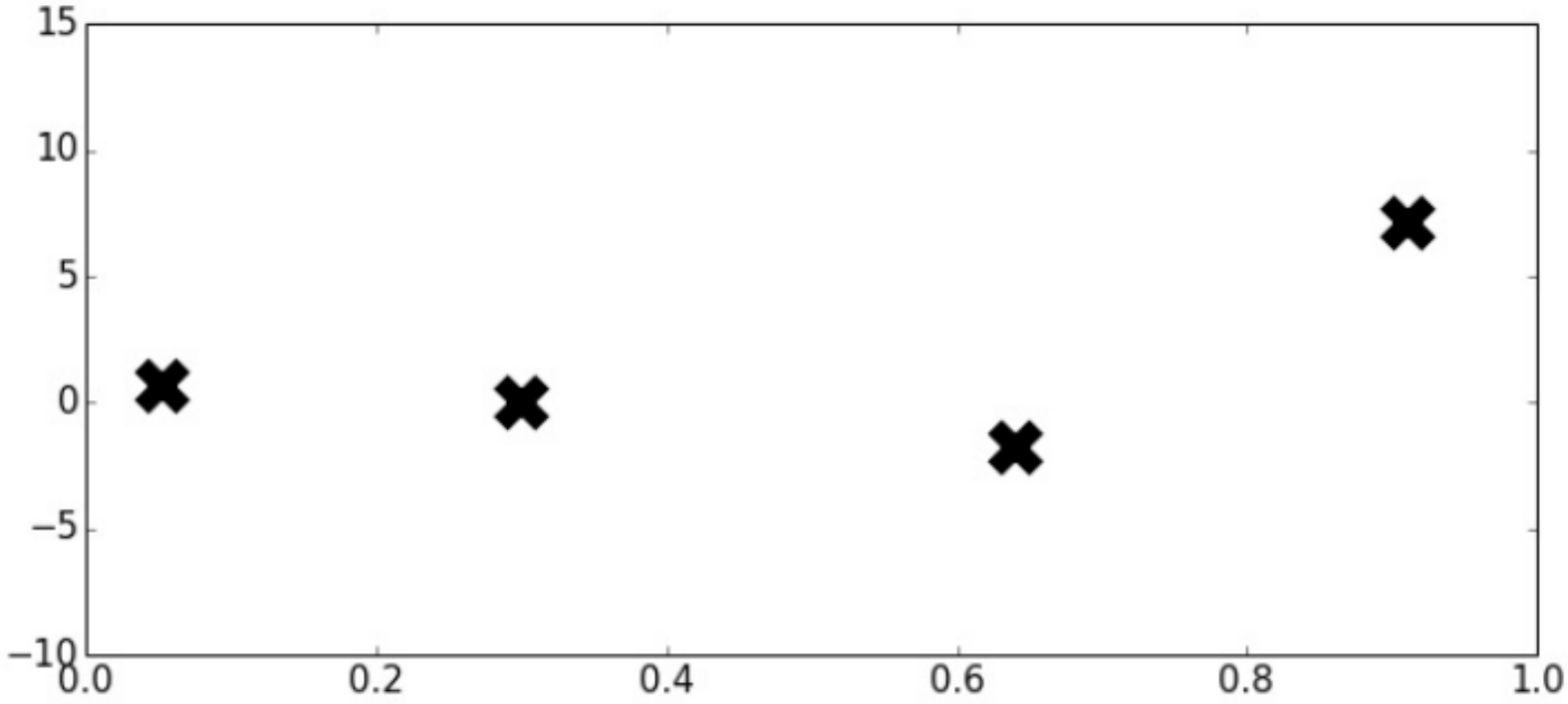
- Find a sequence that best binds to a TF
- Find an airplane wing design that gives the most lift
- How to tune hyperparameters of a neural network automatically
- Optimize web design to maximize purchases
- Find an antibody that best binds to a target

How do we choose the next feature values to observe?

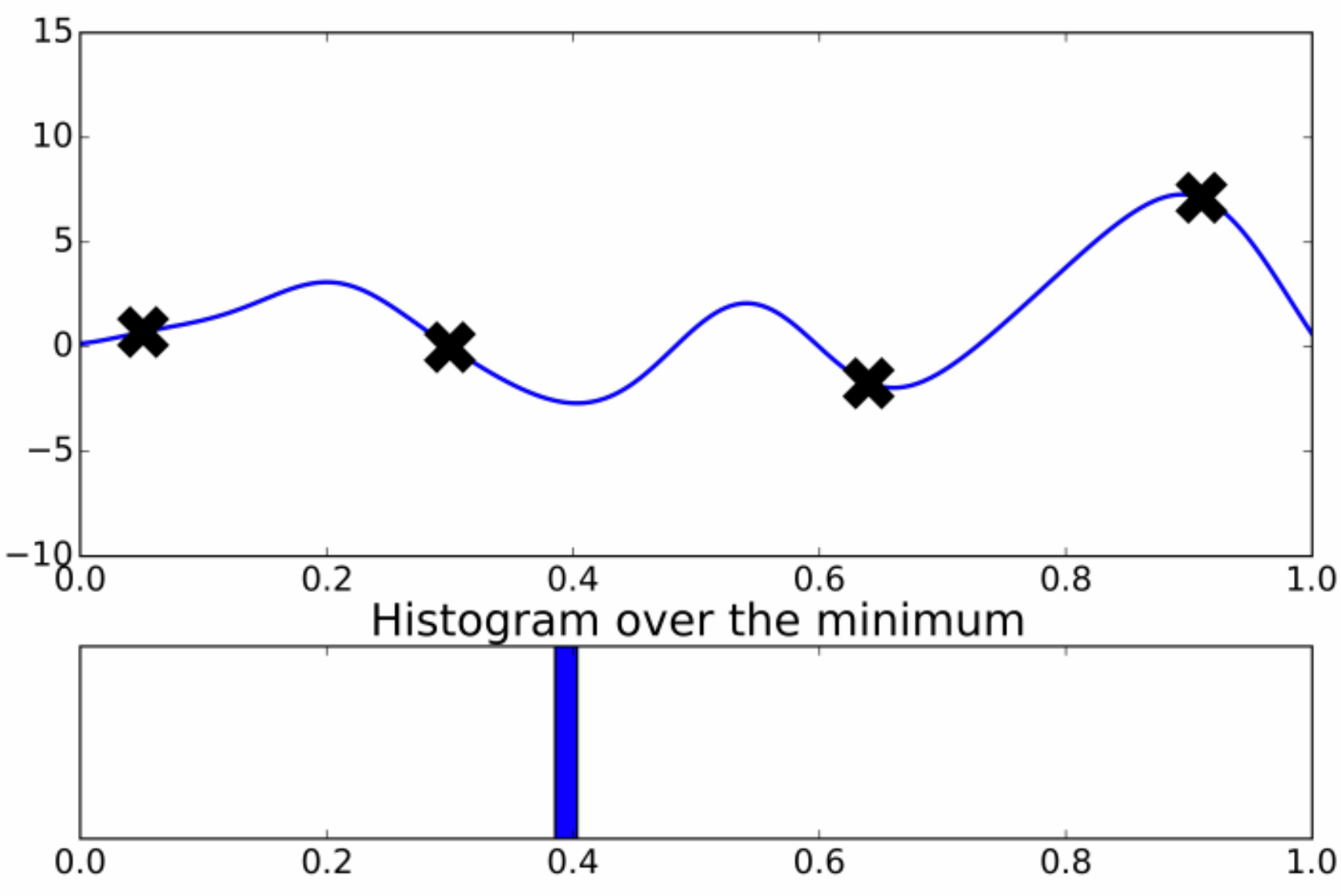
- Prior knowledge
  - Largely used in Biology
- Grid search
  - Expensive
- Grid search is still used when the number of parameters is small. One example is tuning neural network hyperparameters

Randomized grid search has advantages over uniform grid search



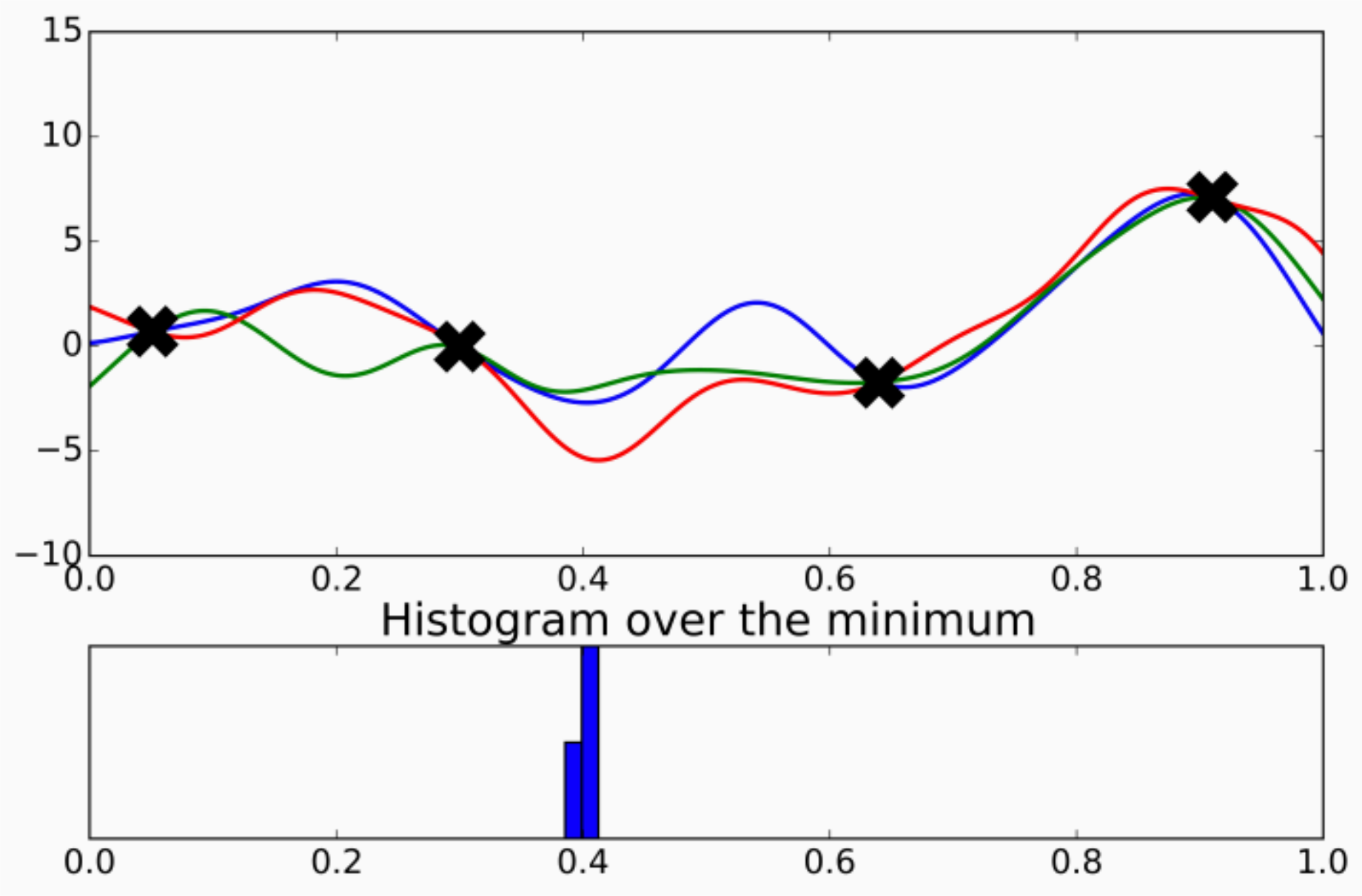


Where is the minimum of f?
Where should the take the next evaluation?


15
10
5
0
−5
−10
0.0
0.2
0.4
0.6
0.8
1.0
Histogram over the minimum
0.0
0.2
0.4
0.6
0.8
1.0

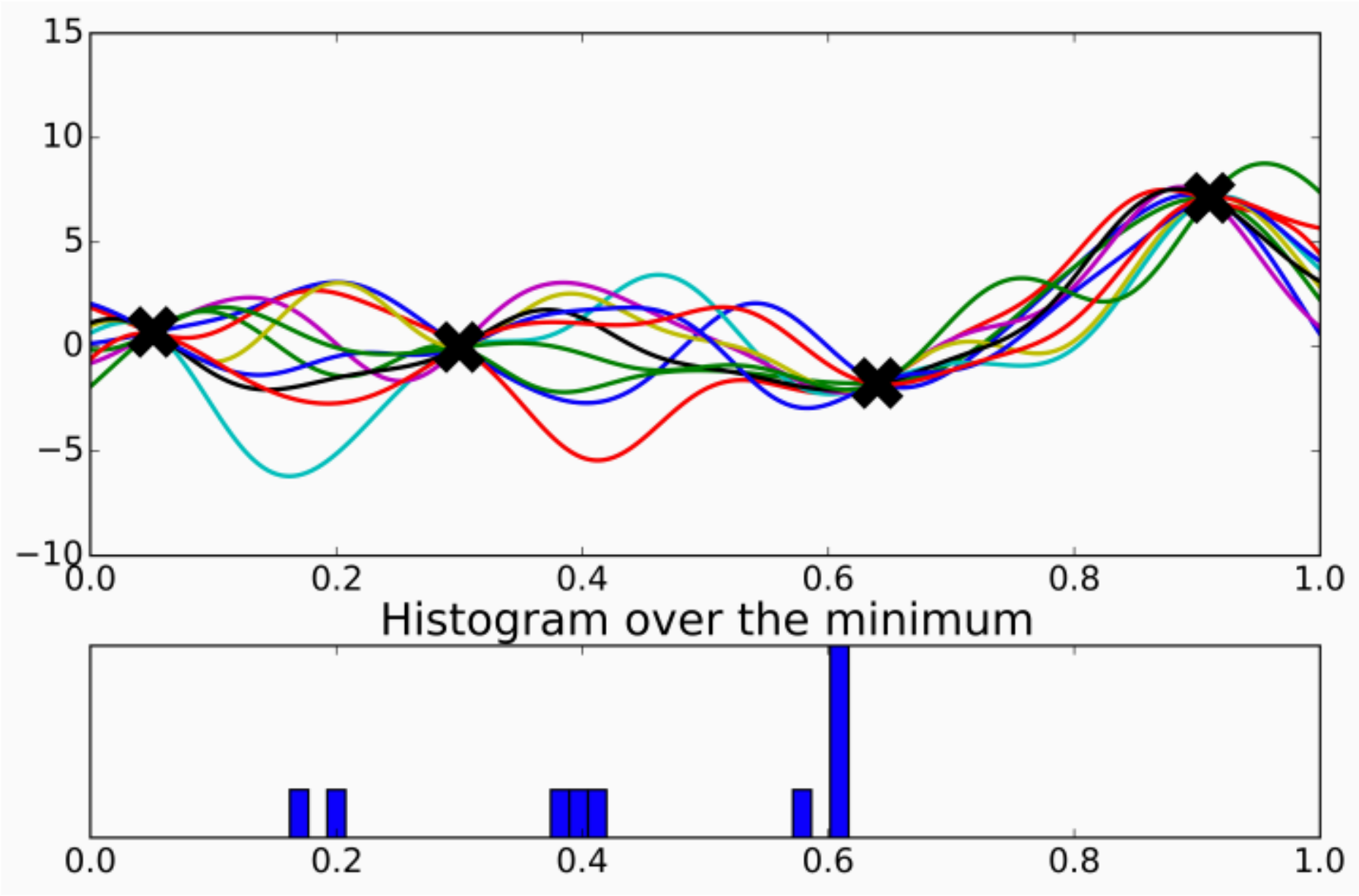
Intuitive solution

Three curves



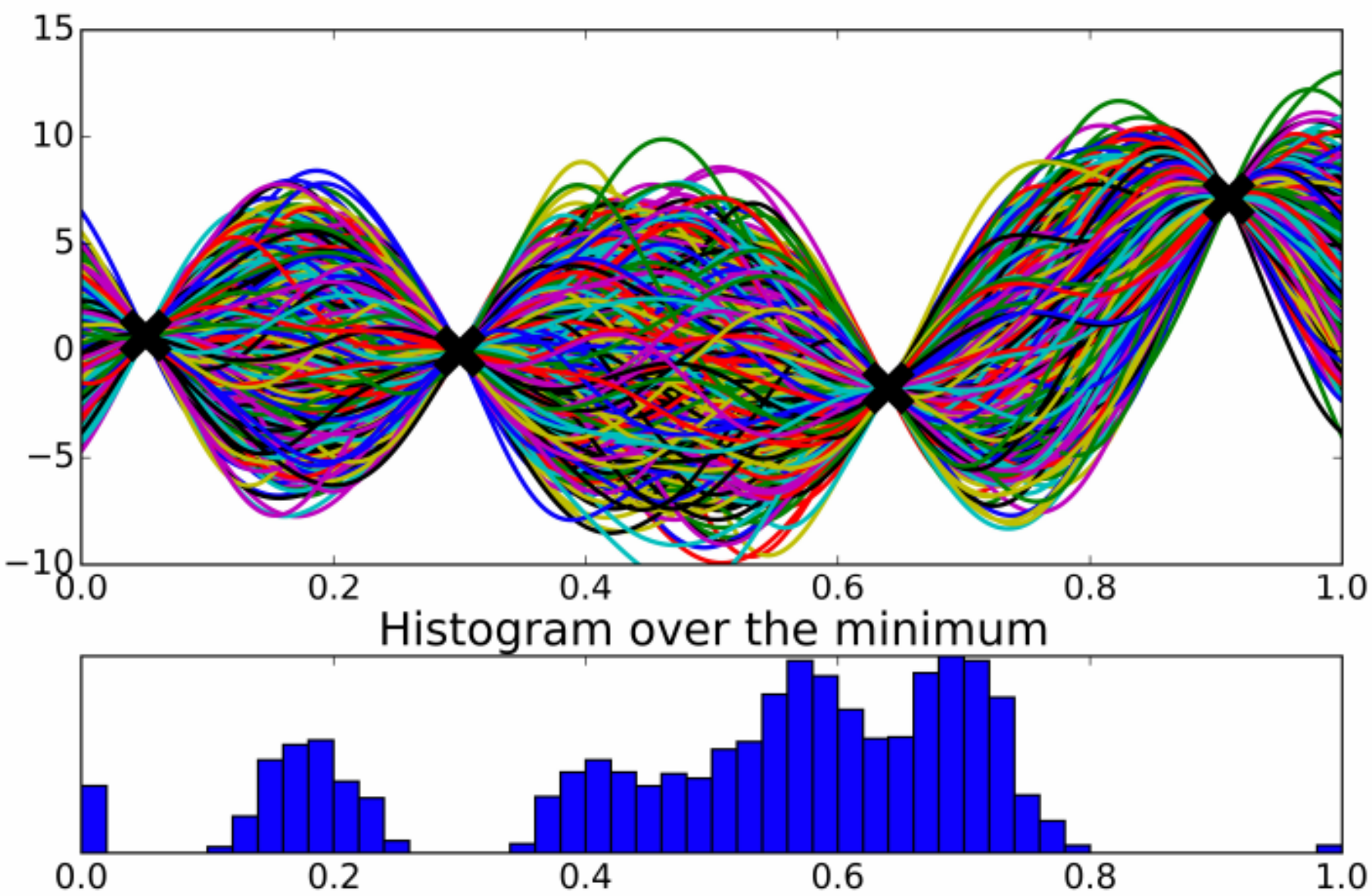
Intuitive solution

Ten curves



Intuitive solution

Many curves



An ***acquisition function*** tells us where to look next

Algorithm 1: Bayesian Optimization

Input: f: function to be optimized
T: number of iterations
$\mathcal{D}$: initial dataset $\{(x_1, y_1), \ldots, (x_n, y_n)\}$

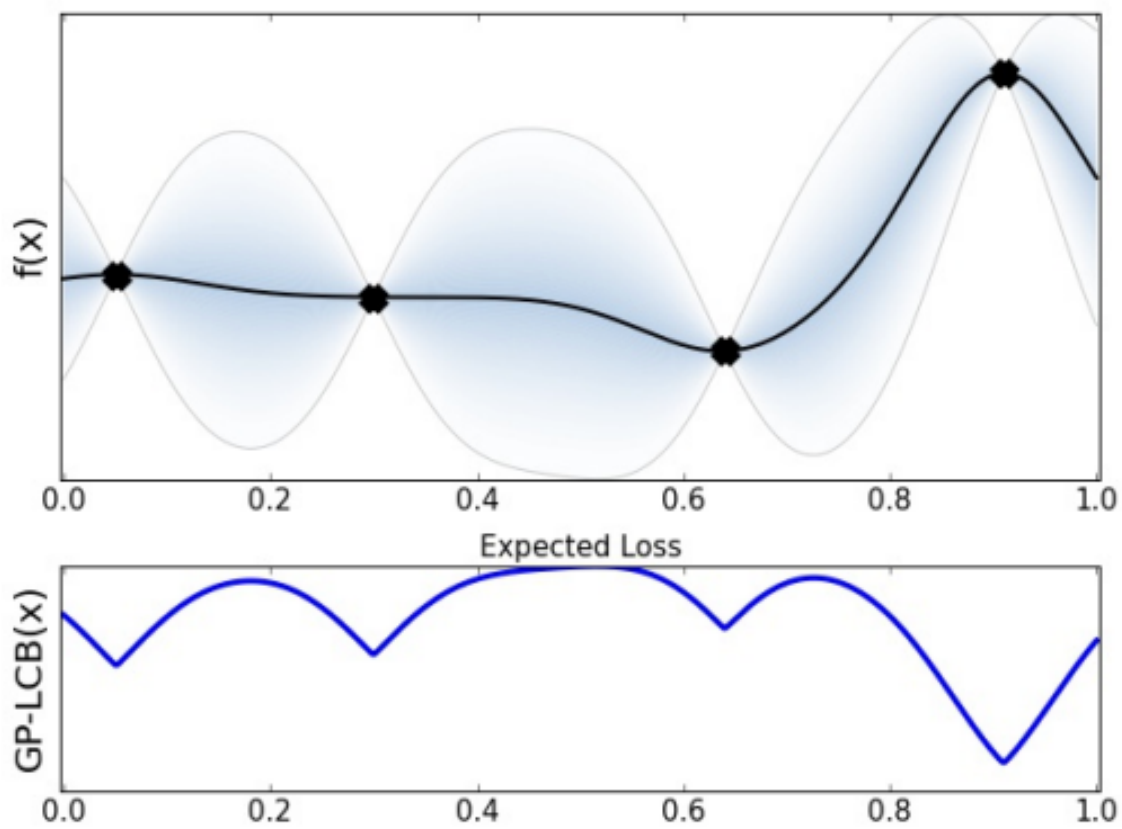
model = create_model($\mathcal{D}$)
for *t in 1, . . . ,* T **:**
  x = acquisition_function(model, $\mathcal{D}$)
  $y = f(x)$ Expensive
  model.update(x, y)
  $\mathcal{D} = \mathcal{D} \cup \{(x, y)\}$
return argmax$_{x \in \mathcal{X}}$ model.expected_value(x)

An ***acquisition function*** has to balance exploitation (next choice is the optimal) vs. exploration (make sure we have have explored the input space)

Lower confidence bound (LCB) acquisition function (here function minimization)

Direct balance between exploration and exploitation:

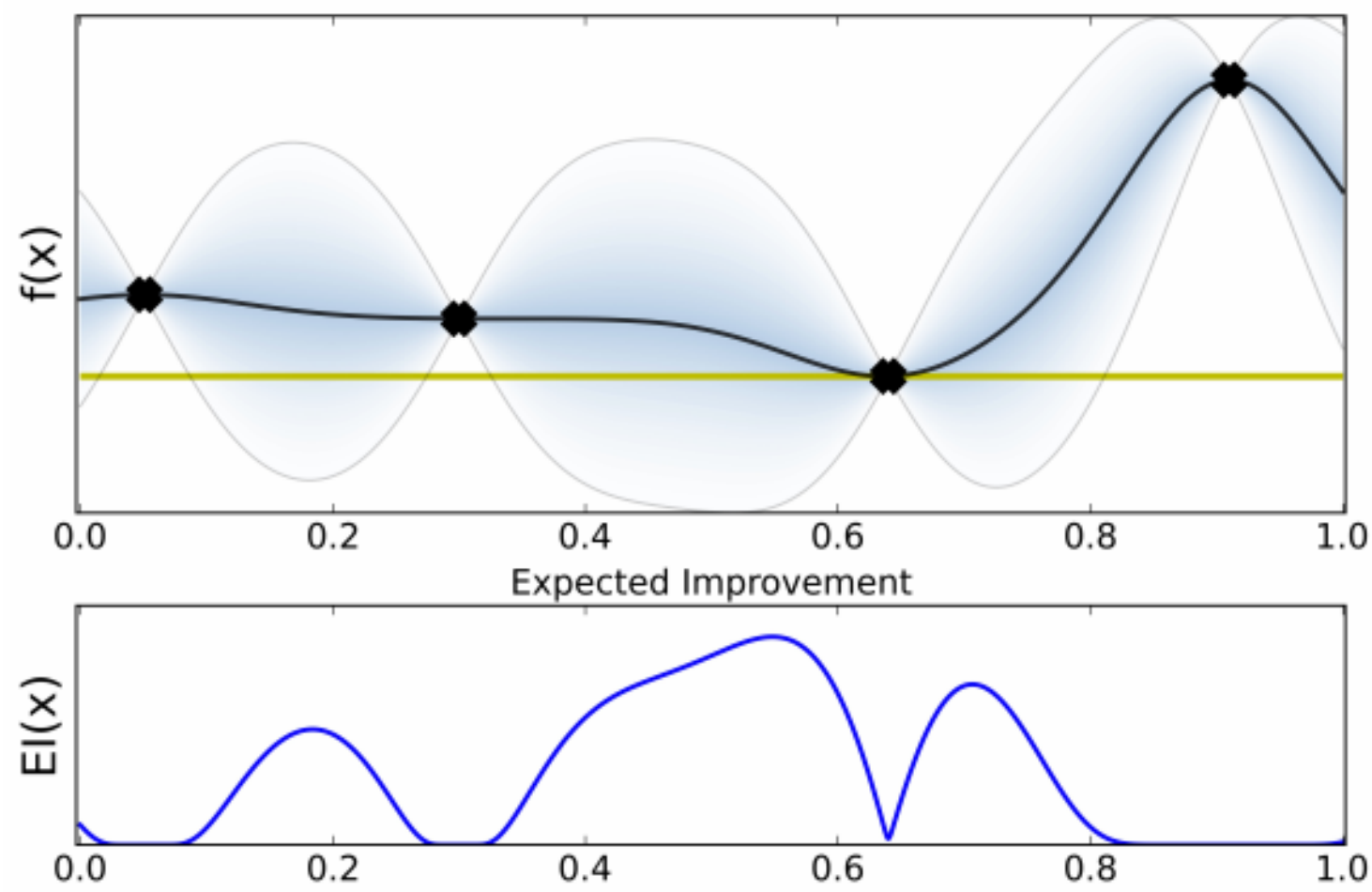
$$\alpha_{LCB}(\mathbf{x};\theta,\mathcal{D}) = -\mu(\mathbf{x};\theta,\mathcal{D}) + \beta_t \sigma(\mathbf{x};\theta,\mathcal{D})$$



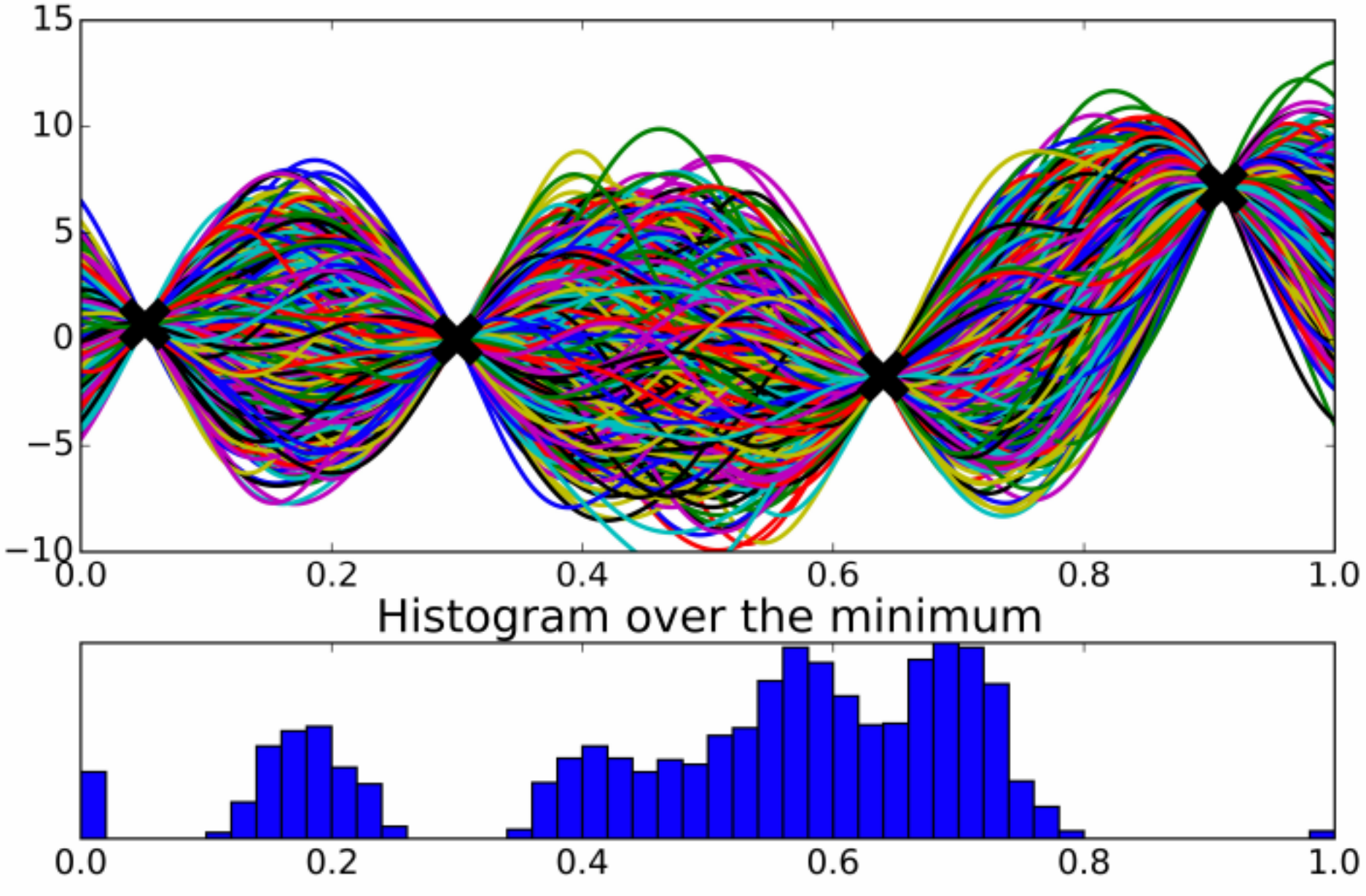
Expected Improvement (EI) acquisition function (here function minimization)

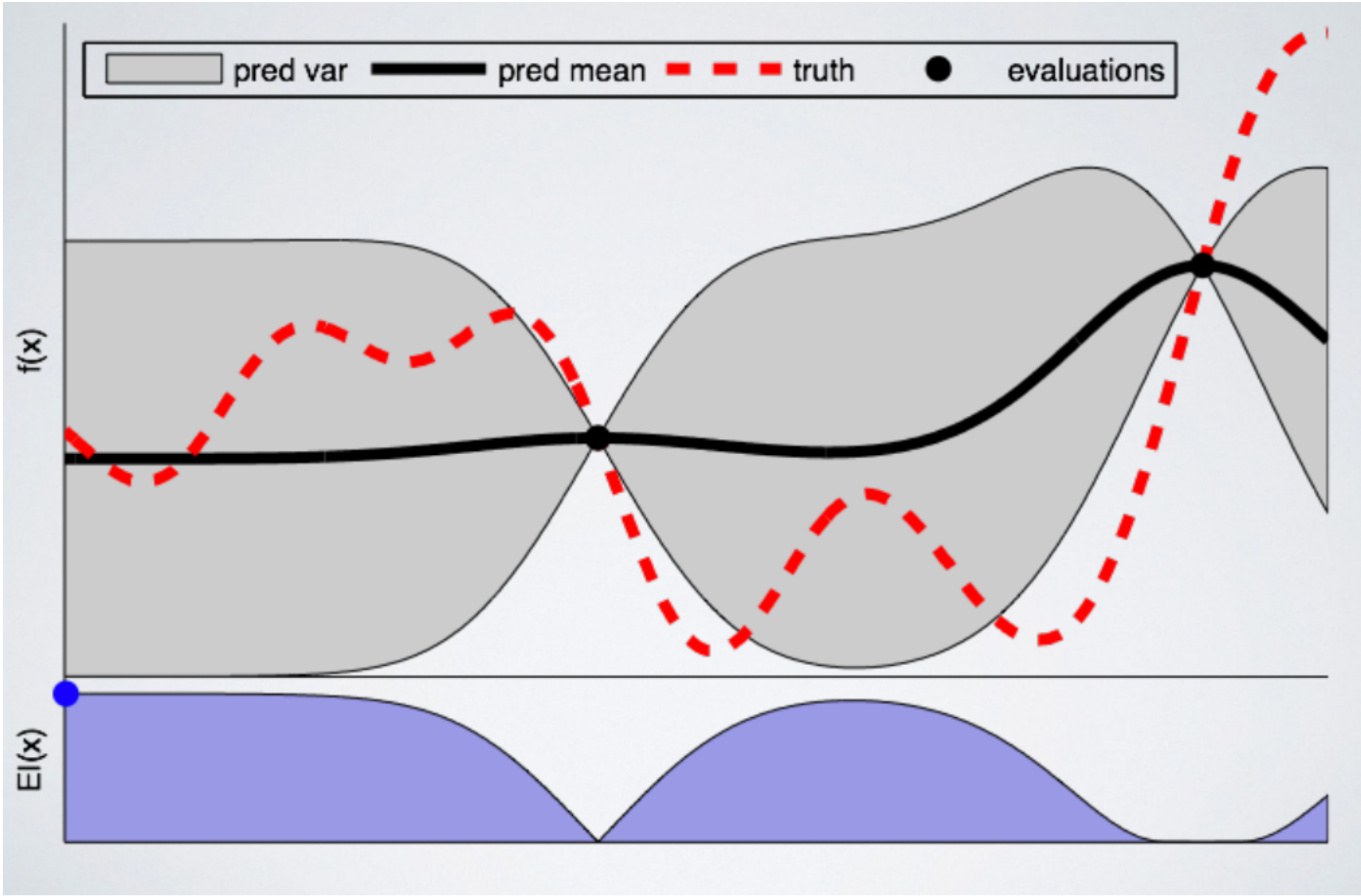
$$\alpha_{EI}(\mathbf{x};\theta,\mathcal{D}) = \int_y \max(0, y_{best} - y) p(y|\mathbf{x};\theta,\mathcal{D}) dy$$

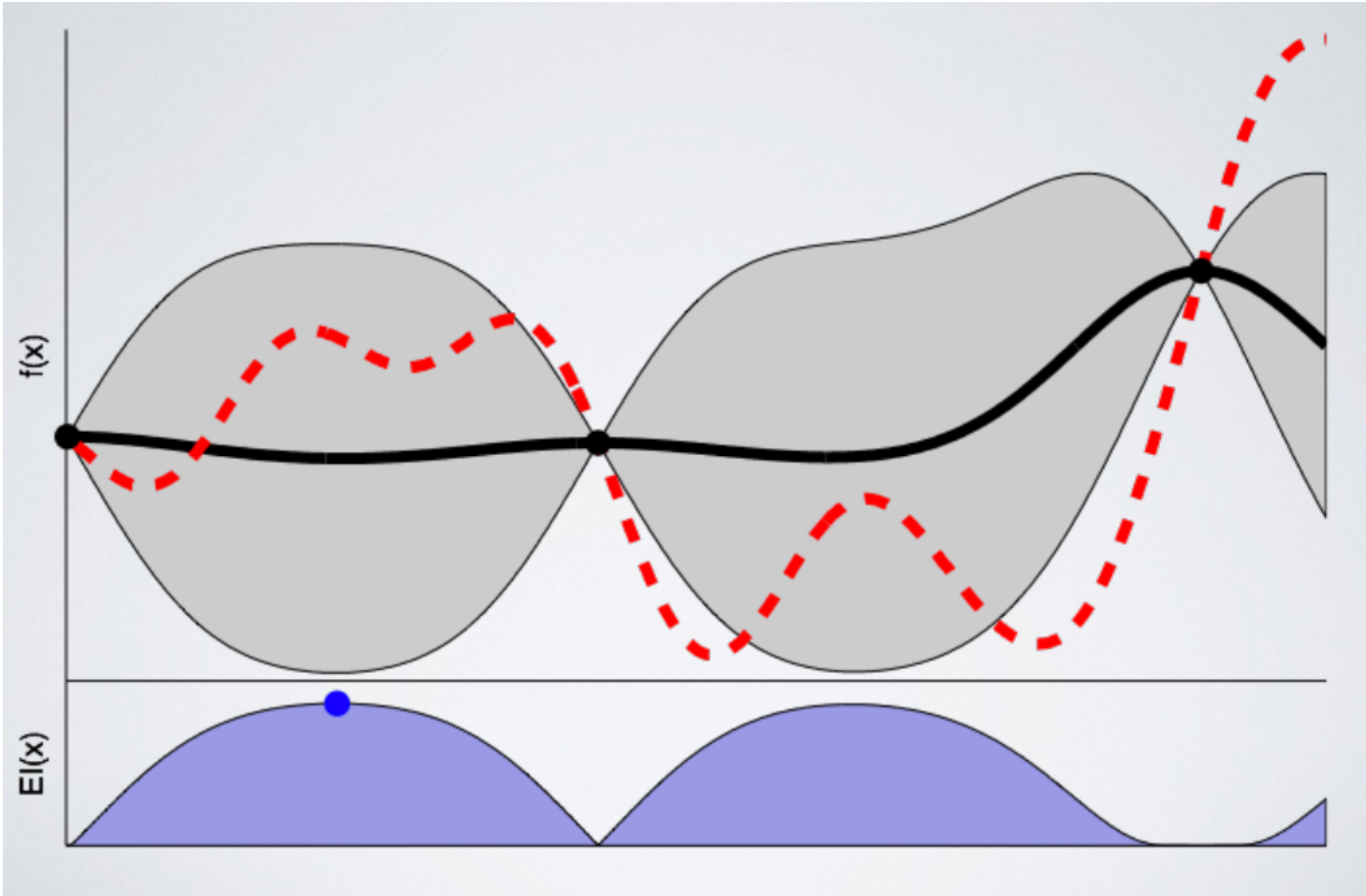
- Commonly used acquisition function
- Explicit form available for Gaussian processes

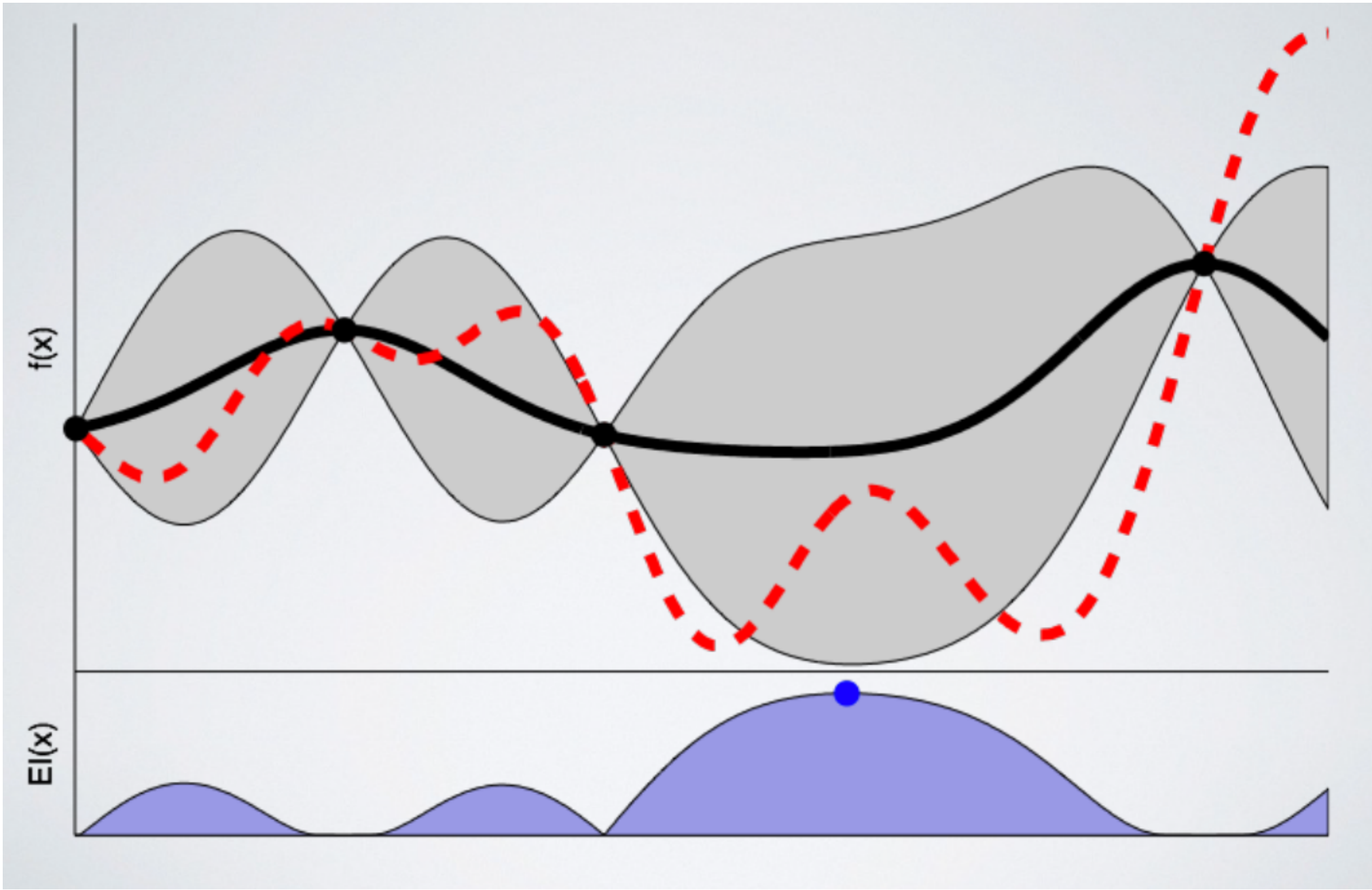


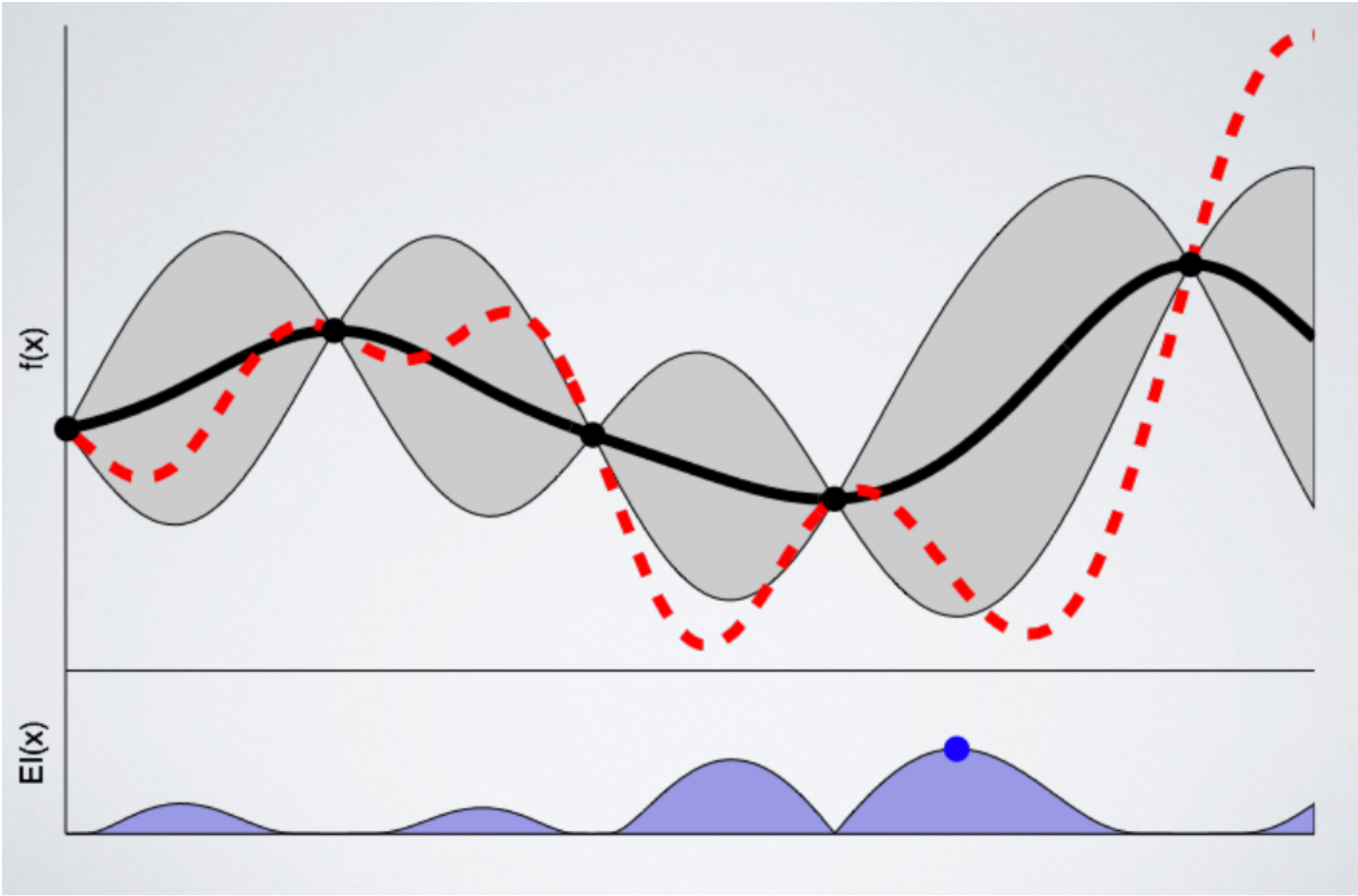
Goal – minimize function
Recall family of functions to define uncertainty

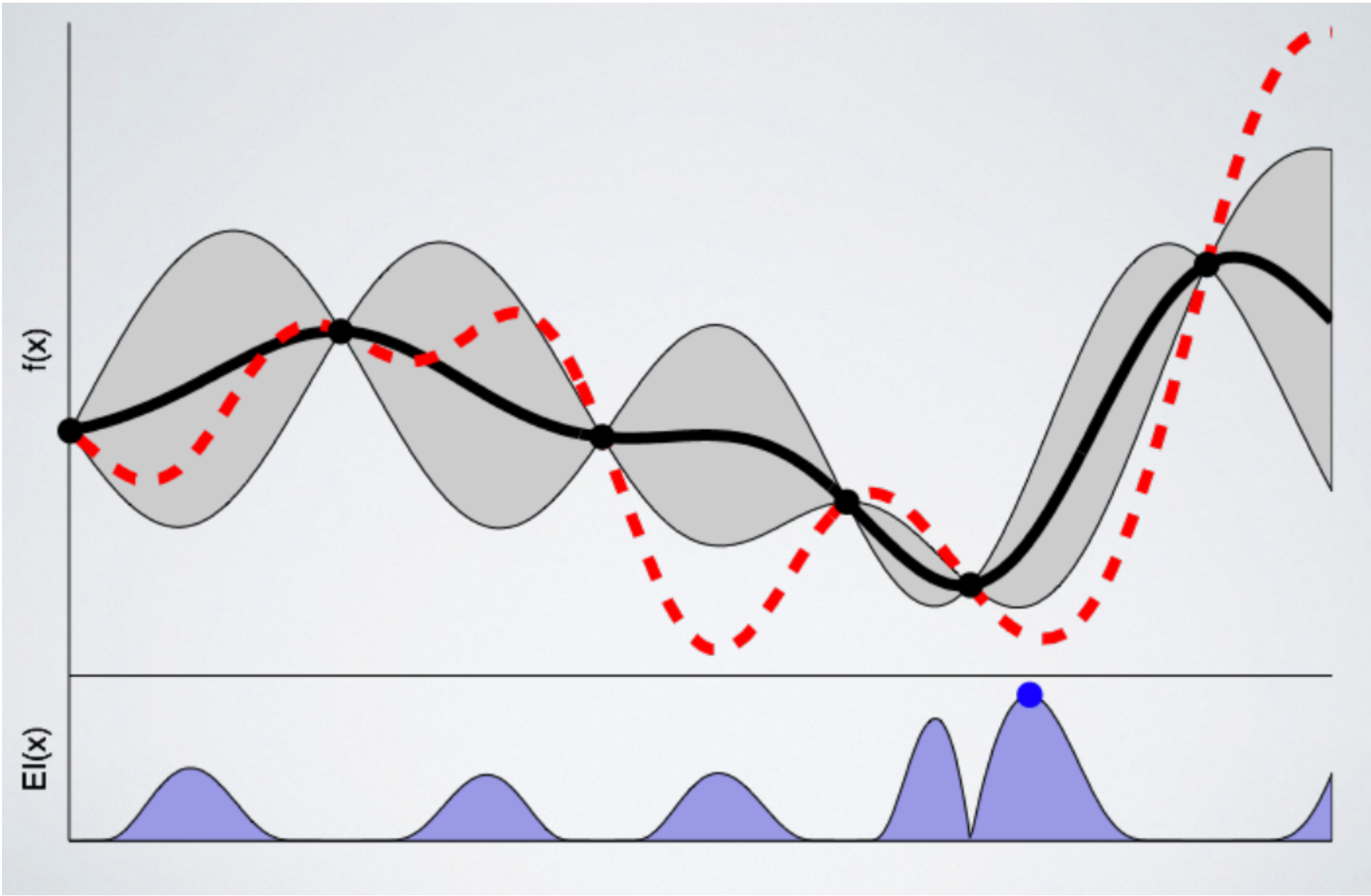


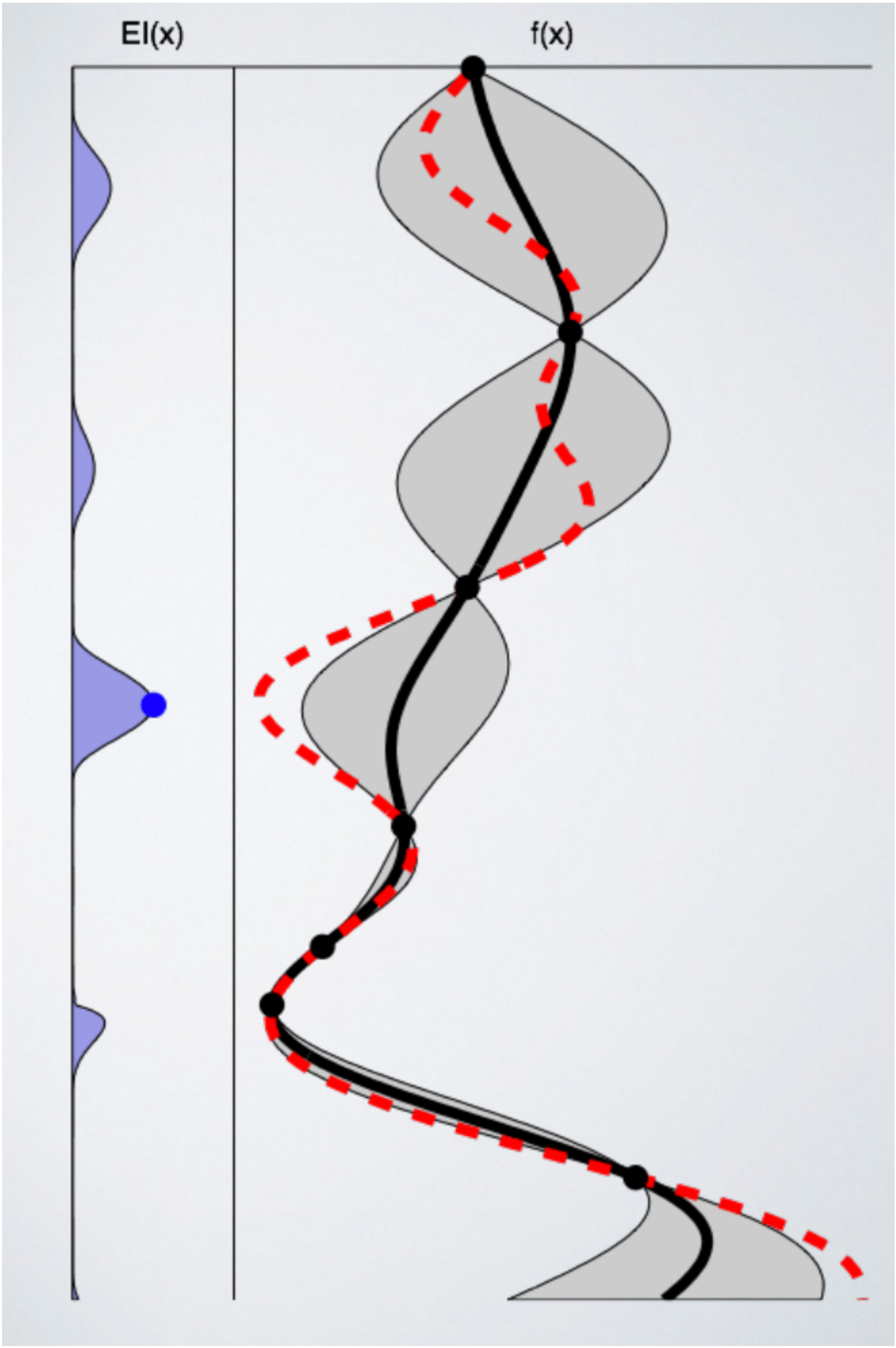

pred var
pred mean
truth
evaluations
f(x)
EI(x)


f(x)
EI(x)


f(x)
EI(x)


f(x)
EI(x)


f(x)
EI(x)


EI(x)
f(x)

Why is Bayesian Optimization not widely used?

- Fragility and poor default choices.
  - Getting the function model wrong can be catastrophic.
- There is not standard software available.
  - Tricky to build from scratch
- Experiments are run sequentially
  - Want to use parallel computing
- Gaussian Processes have limited ability to scale
  - Need alternative models of uncertainty

(In part, Bryan Adams)

Experiment design using deep ensembles

How can we use Bayesian optimization to design our next k-mer experiment?

- We a model f of the binding of a transcription factor to 8-mer DNA sequences.
  - Binding = f(8-mer sequence)
  - Assume given training data { (s_1, b_1), (s_2, b_2) … (s_n, b_n) } to train f
  - Goal is to discover s_{best} = argmax f(s)
  - Need best model f; what the next next s_x we should ask to observe?
- What is a principled way to choose s_x ?

How can we use Bayesian optimization to design our next k-mer experiment?

- Let's use Deep ensembles for Bayesian optimization
  - (Though note that ensembles are not "Bayesian")
- But ensembles can be uncalibrated
- One way to improve calibration…

MOD (Maximizing Overall Diversity) can improve uncertainty calibration

- In a nutshell – maximize variance on the uniform distribution over all possible inputs as part of the loss
- Equivalent to maximizing entropy for Gaussian distributions

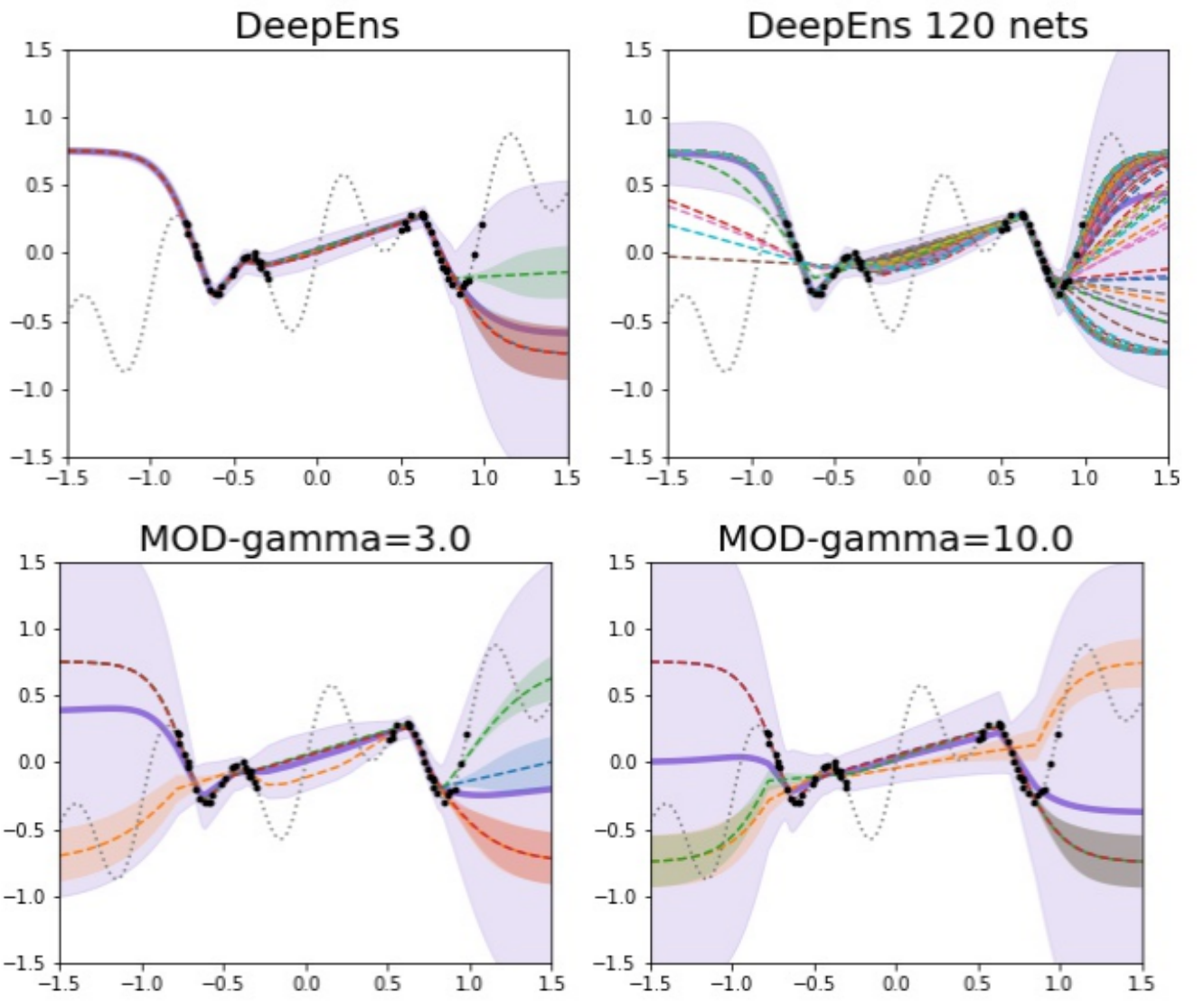
$$\min_{\theta_1,\ldots,\theta_M} \; L_{in} - \gamma L_{out}$$

$$L_{in} = \frac{1}{M} \sum_{m=1}^{M} E_{P_{in}}[L(\theta_m, x, y)]$$

$$L_{out} = \frac{1}{Z} \int_{\mathcal{X}} \sigma^2_{mod}(x) \, \mathrm{d}x$$

Ensembles can provide reasonable uncertainty estimates

$$y = 0.3x + 0.3\sin(2\pi x) + 0.3\sin(4\pi x) + \epsilon$$

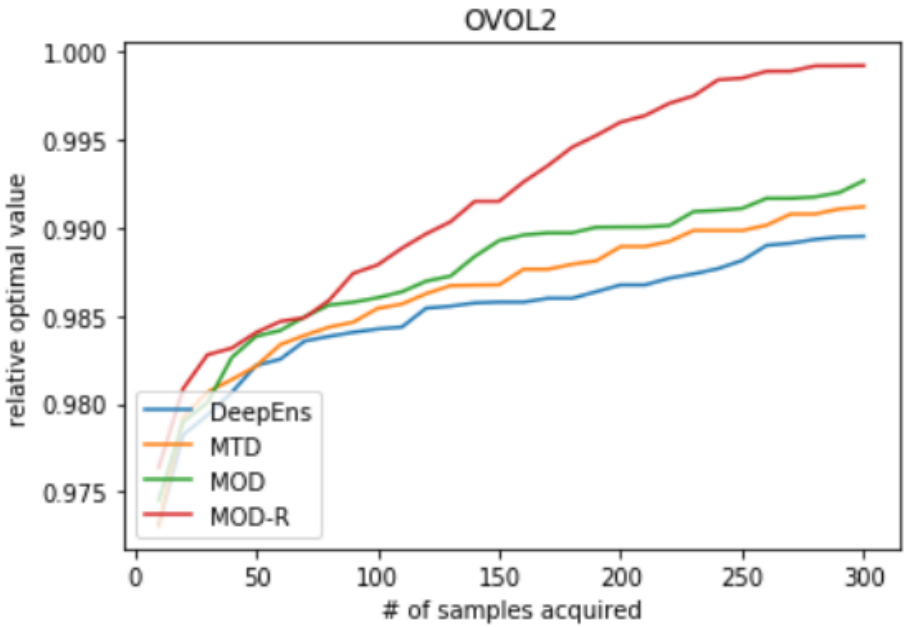


Ensembles can provide uncertainty estimates for choosing the next k-mer to observe

- Protein-DNA binding
  - 38 different TFs, 8-mer binding data derived from PBMs
  - Neural network architecture with a single hidden layer
- Ensemble size 4 throughout

Better uncertainty metrics converge on best value with fewer new experiments (samples)

- Acquisition function – UCB
- 30 rounds of acquisition of size 10 each
- 10% held out data used for hyperparameter selection
- MTD is a control where you maximize variance on *training* inputs



FIN - Thank You