# CSC 2541: Neural Net Training Dynamics

## Lecture 8 - Implicit Regularization and Bayesian Inference

Roger Grosse

University of Toronto, Winter 2022

# Today

- Last week: the curse of stochasticity (slower convergence)
- This week: the blesssing of stochasticity (implicit regularization)
- Agenda
  - "Sharp and flat minima"
  - Disentangling implicit and explicit regularization
  - Explicit Bayesian inference
  - Implicit Bayesian inference
- This lecture will have lots of ideas that connect to each other, but no unifying message. This general topic is still messy!

# Explicit and Implicit Regularization

How I'll use the terms in this course:

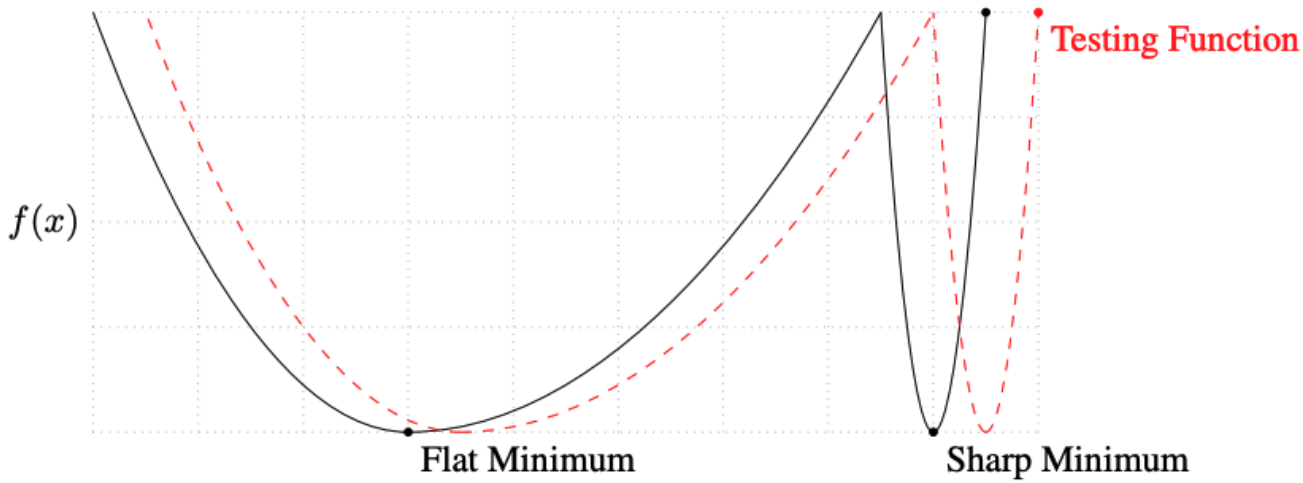
- Explicit regularization imposes an explicit penalty on the parameters of the model (typically some measure of complexity or sensitivity)
  - $\ell_2$ regularization (traditional view — weight decay is more complicated, as we saw in Lecture 5)
  - dropout
  - batch norm (noisy statistics)
  - variational Bayes / MDL
- Implicit regularization occurs when the dynamics of training lead to certain minima rather than others
  - minimum norm solutions in linear regression (Lecture 1)
  - NTK dynamics (Lecture 6)
  - effects of gradient noise (still poorly understood)

# Sharp and Flat Minima

# Sharp and Flat Minima

Keskar et al., 2017. "On large batch training for machine learning: Generalization gap and sharp minima"

- Common observation: training with larger batch sizes often leads to worse test error, even if the model can fit all the training data
- **Claims:**
  - Training with larger batches leads to sharper minima
    - **Caution:** not necessarily "local minima". Could be a continuous manifold of minimizers.
  - Sharper minima generalize worse
- Sharpness quantified using the largest eigenvalues of $\mathbf{H}$

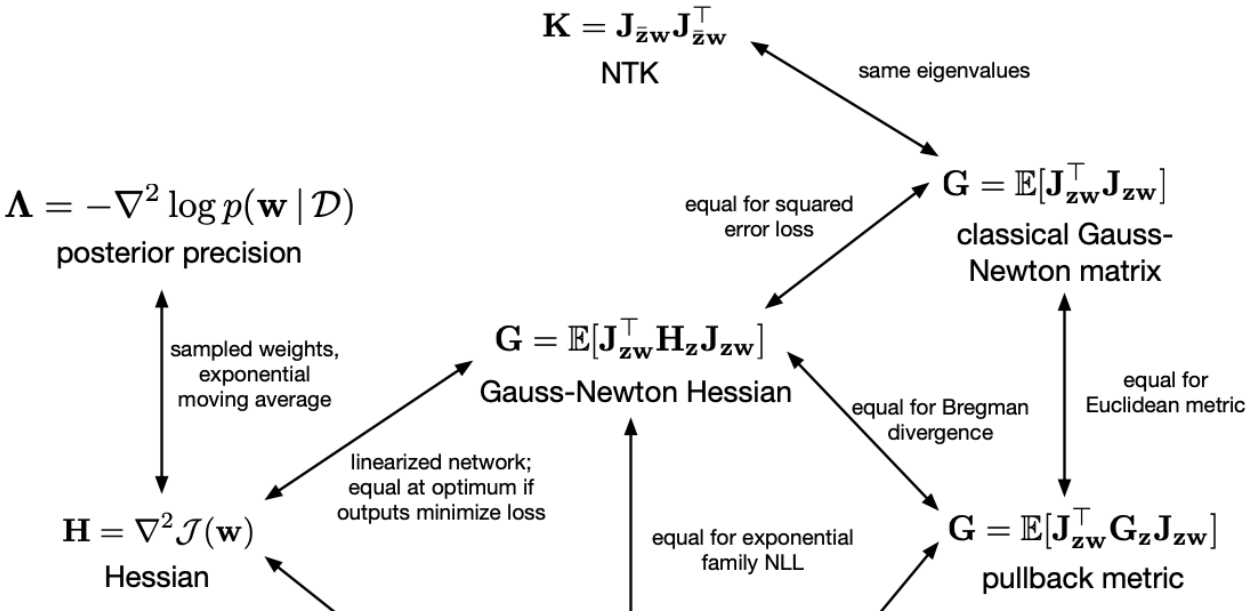


# Sharp and Flat Minima

- The idea of sharp and flat minima is intuitive, but it's notoriously hard to fit everything together rigorously
- Dinh et al., 2017. "Sharp minima can generalize for deep nets"
  - We can easily reparameterize the network to an equivalent one with larger or smaller $\mathbf{H}$, so the size of $\mathbf{H}$ can't explain generalization
  - Note that some notions of sharpness don't have this problem
- Shallue et al., 2018. "Measuring the effects of data parallelism on neural network training"
  - Their experiments considered validation loss for a wide variety of tasks and architectures, and they never saw any degradation from large batches
  - They identify 2 frequent confounds in the literature
    - Batch norm creates an *explicit* regularizer which is stronger for smaller batch sizes
    - Some papers fix the number of *epochs*, so models with larger batches were trained for fewer iterations

# Sharp and Flat Minima

- Another complication: $\mathbf{H}$ is related to lots of other matrices, so any apparent correlation between $\mathbf{H}$ and generalization might mean the *other* matrix explains generalization




- **Speculation:** the largest eigenvalue of $\mathbf{G} = \mathbb{E}[\mathbf{J}^\top \mathbf{J}]$ is the largest singular value of $\mathbf{J}$.
  - Smaller $\|\mathbf{J}\|_2$ implies less sensitivity to input perturbations (also consider adversarial robustness)
  - Bounded $\|\mathbf{J}\|_2$ implies a Lipschitz function, which was used in classical generalization bounds

# Dropout

# Dropout

- For a network to overfit, its computations need to be really precise. This suggests regularizing them by injecting noise into the computations, a strategy known as stochastic regularization.
- Dropout is a stochastic regularizer which randomly deactivates a subset of the units (i.e. sets their activations to zero).

$$h_i = \begin{cases} \phi(z_i) & \text{with probability } 1-\rho \\ 0 & \text{with probability } \rho, \end{cases}$$

where $\rho$ is a hyperparameter.

- Equivalently,

$$h_i = m_i \cdot \phi(z_i),$$

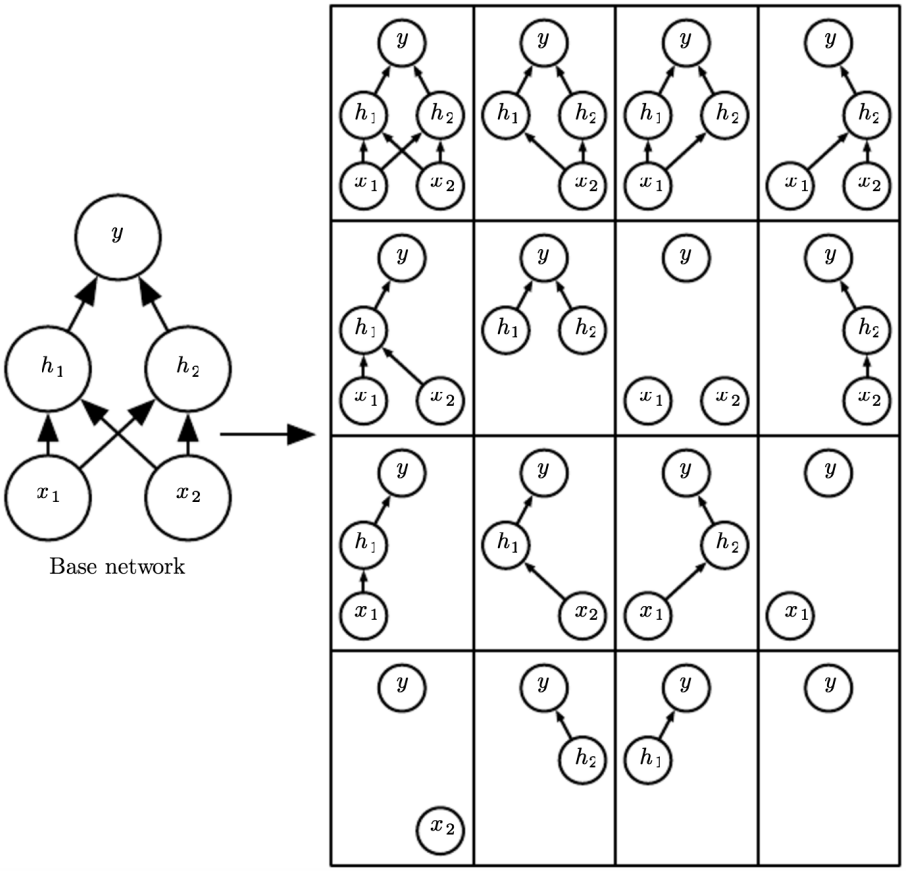
where $m_i$ is a Bernoulli random variable, independent for each hidden unit.

- Backprop rule:

$$\overline{z_i} = \overline{h_i} \cdot m_i \cdot \phi'(z_i)$$

# Dropout

- Dropout can be seen as training an ensemble of $2^D$ different architectures with shared weights (where $D$ is the number of units):




— Goodfellow et al., *Deep Learning*

# Dropout

Dropout at test time:

- Most principled thing to do: run the network lots of times independently with different dropout masks, and average the predictions.
  - Individual predictions are stochastic and may have high variance, but the averaging fixes this.
- In practice: don't do dropout at test time, but multiply the weights by $1 - \rho$
  - Since the weights are on $1 - \rho$ fraction of the time, this matches their expectation.

# Dropout

- Dropout is traditionally viewed as an explicit regularizer. If the cost is the expectation with respect to the dropout mask, then:

$$\begin{aligned}
\mathcal{J}_{\text{reg}}(\mathbf{w}) &= \mathcal{J}(\mathbf{w}) + \mathcal{R}(\mathbf{w}) \\
\mathcal{J}(\mathbf{w}) &= \mathbb{E}_{\mathbf{x},\mathbf{t}}[\mathcal{L}(\mathbf{t}, f(\mathbf{x}, \mathbf{w}))] \\
\mathcal{R}(\mathbf{w}) &= \mathbb{E}_{\mathbf{x},\mathbf{t}}\left[\mathbb{E}_{\mathbf{m}}[\mathcal{L}(\mathbf{t}, f_{\text{drop}}(\mathbf{x}, \mathbf{w}; \mathbf{m}))] - \mathcal{L}(\mathbf{t}, f(\mathbf{x}, \mathbf{w}))\right]
\end{aligned}$$

- Here, $\mathbf{w}$ represents the "test time weights" (i.e. after multiplying by $1 - \rho$)
- $\mathcal{R}$ is the amount the training predictions were hurt as a result of the stochasticity
- The original dropout paper (Srivastava et al., 2014) derived $\mathcal{R}$ explicitly for linear regression. It's equivalent to an $\ell_2$ penalty reweighted by the feature variance.

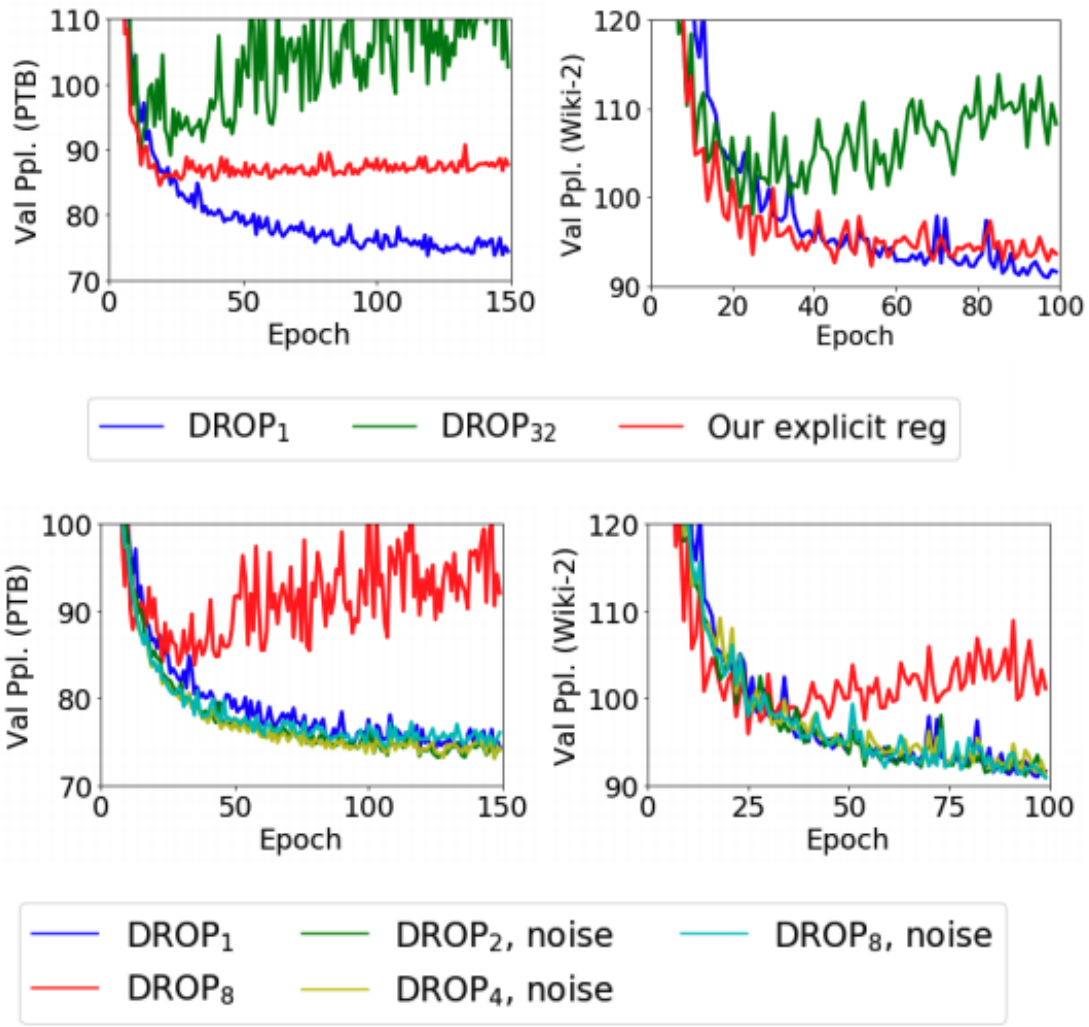
# Dropout

Wei, Kakade, and Ma, 2020, "The implicit and explicit regularization effects of dropout"

- Since dropout samples the mask stochastically, it adds noise to the gradients
- This noise might have an implicit regularization effect, in addition to the explicit regularizer
- Disentangling the implicit and explicit effects
  - Sample $K$ independent dropout masks for each gradient computation
  - This is still doing SGD on the same explicit objective
  - $K = 1$ is ordinary dropout training, and the dropout-induced gradient noise decays as $1/K$. So the implicit regularization effect vanishes as $K \to \infty$
- A further "knock-in" manipulation: compute the amount of gradient noise that was removed, inject it back in, and see if the algorithm behaves like ordinary dropout

# Dropout

Wei et al., 2020



# Variational Bayes and MDL

# BNN Posterior Inference

- The cheapest and simplest way of training Bayesian neural nets is probably maximum a-posteriori (MAP) inference, which simply maximizes the posterior probability:

$$\begin{aligned}\mathbf{w}_{\text{MAP}} &= \arg\max_{\mathbf{w}} \log p(\mathbf{w} \mid \mathcal{D}) \\ &= \arg\max_{\mathbf{w}} \log p(\mathbf{w}, \mathcal{D}) \\ &= \arg\max_{\mathbf{w}} \overbrace{\sum_i \log p(\mathbf{t}^{(i)} \mid \mathbf{w}, \mathbf{x}^{(i)})}^{\text{likelihood}} + \overbrace{\log p(\mathbf{w})}^{\text{prior}}\end{aligned}$$

- With an i.i.d. Gaussian prior on the weights, the prior term is equivalent to $\ell_2$ regularization (weight decay)
- To get uncertainty estimates, we can use the Laplace approximation to the posterior, which takes the second-order Taylor approximation to the log-likelihood around $\mathbf{w}_{\text{MAP}}$:

$$p(\mathbf{w} \mid \mathcal{D}) \approx \mathcal{N}(\mathbf{w}; \mathbf{w}_{\text{MAP}}, \bar{\mathbf{H}}^{-1})$$
$$\bar{\mathbf{H}} = N\mathbf{H} = -\nabla^2 \log p(\mathbf{w} \mid \mathcal{D})$$

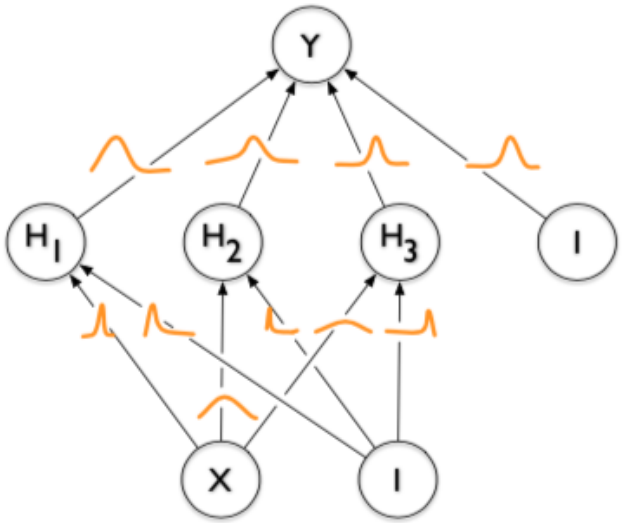
- This is like the Hessian of the training cost, up to a factor of $N$

# Variational BNNs

- The problem with the Laplace approximation is that uncertainty isn't considered until training is finished
- Variational Bayes is a more accurate posterior inference method which accounts for uncertainty during training
- We approximate a complicated posterior distribution with a simpler variational approximation. E.g., assume a Gaussian posterior with diagonal covariance (i.e. fully factorized Gaussian):

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$$
$$= \prod_{j=1}^{D} \mathcal{N}(\theta_j; \mu_j, \sigma_j)$$

- This means each weight of the network has its own mean and variance.




— Blundell et al., Weight uncertainty for neural networks

# Posterior Inference: Variational Bayes

- The marginal likelihood is the probability of the observed data (targets given inputs), with all possible weights marginalized out:

$$\begin{aligned} p(\mathcal{D}) &= \int p(\mathbf{w})\, p(\mathcal{D} \mid \mathbf{w})\, \mathrm{d}\mathbf{w} \\ &= \int p(\mathbf{w})\, p(\{t^{(i)}\} \mid \{\mathbf{x}^{(i)}\}, \mathbf{w})\, \mathrm{d}\mathbf{w}. \end{aligned}$$

- Analogously to VAEs, we define a variational lower bound:

$$\log p(\mathcal{D}) \geq \mathcal{F}(q) = \mathbb{E}_{q(\mathbf{w})}[\log p(\mathcal{D} \mid \mathbf{w})] - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{w}) \,\|\, p(\mathbf{w}))$$

- Unlike with VAEs, $p(\mathcal{D})$ is fixed, and we are *only* maximizing $\mathcal{F}(q)$ with respect to the variational posterior $q$ (i.e. a mean and standard deviation for each weight).

# Posterior Inference: Variational Bayes

$$\log p(\mathcal{D}) \geq \mathcal{F}(q) = \mathbb{E}_{q(\mathbf{w})}[\log p(\mathcal{D} \mid \mathbf{w})] - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{w}) \,\|\, p(\mathbf{w}))$$

- Same as for VAEs, the gap equals the KL divergence from the true posterior:

$$\mathcal{F}(q) = \log p(\mathcal{D}) - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{w}) \,\|\, p(\mathbf{w} \mid \mathcal{D})).$$

  Hence, maximizing $\mathcal{F}(q)$ is equivalent to approximating the posterior.

# Posterior Inference: Variational Bayes

- Likelihood term:

$$\mathbb{E}_{q(\mathbf{w})}[\log p(\mathcal{D} \mid \mathbf{w})] = \mathbb{E}_{q(\mathbf{w})}\left[\sum_{i=1}^{N} \log p(t^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w})\right]$$

  This is just the usual likelihood term (e.g. minus classification cross-entropy), except that $\mathbf{w}$ is sampled from $q$.

- KL term:

$$\mathrm{D}_{\mathrm{KL}}(q(\mathbf{w}) \,\|\, p(\mathbf{w}))$$

  This term encourages $q$ to match the prior, i.e. each dimension to be close to $\mathcal{N}(0, \eta^{1/2})$.

- Without the KL term, the optimal $q$ would be a point mass on $\mathbf{w}_{\mathrm{ML}}$, the maximum likelihood weights.
  - Hence, the KL term encourages $q$ to be more spread out (i.e. more stochasticity in the weights).

# Posterior Inference: Variational Bayes

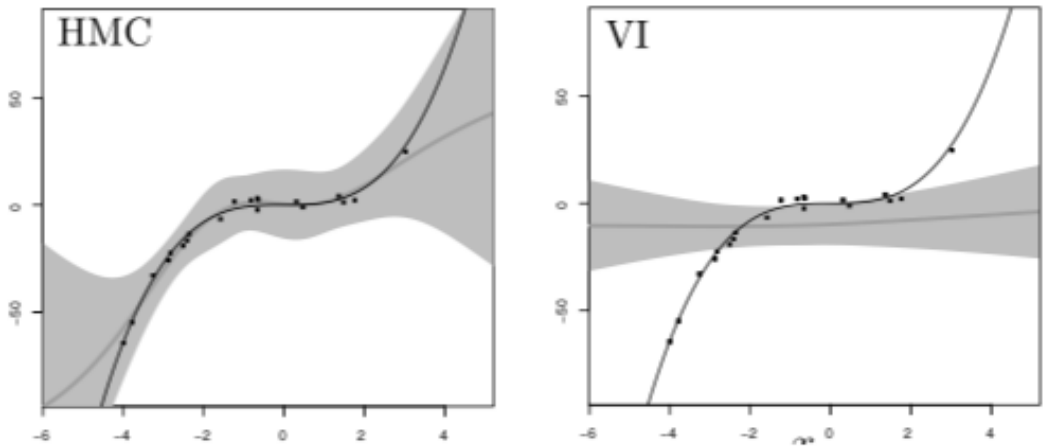
- We can train a variational BNN using the same reparameterization trick as from VAEs.

$$\theta_j = \mu_j + \sigma_j \epsilon_j,$$

  where $\epsilon_j \sim \mathcal{N}(0, 1)$.
- Then the $\epsilon_j$ are sampled at the beginning, independent of the $\mu_j, \sigma_j$, so we have a deterministic computation graph we can do backprop on.
- If all the $\sigma_j$ are 0, then $\theta_j = \mu_j$, and this reduces to ordinary backprop for a deterministic neural net.
- Hence, variational inference injects stochasticity into the computations. This acts like a regularizer, just like with dropout.
  - The difference is that it's stochastic activations, rather than stochastic weights.
  - See Kingma et al., "Variational dropout and the local reparameterization trick", for the precise connections between variational BNNs and dropout.

# Posterior Inference: Variational Bayes

- Bad news: variational BNNs aren't a good match to Bayesian posterior uncertainty.
- The BNN posterior distribution is complicated and high dimensional, and it's really hard to approximate it accurately with fully factorized Gaussians.

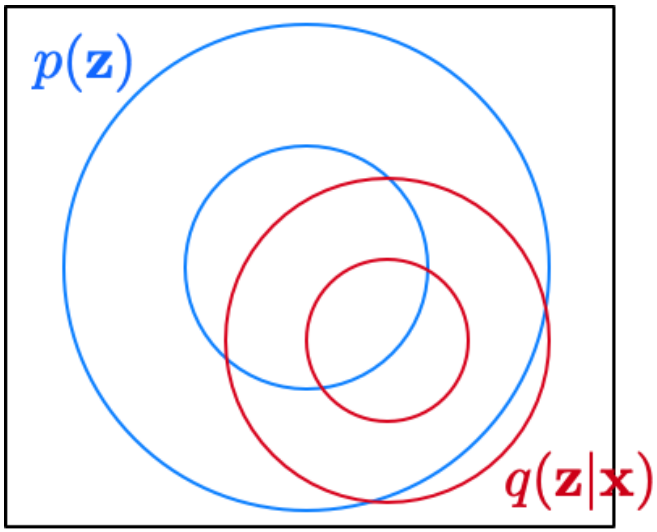



— Hernandez-Lobato et al., Probabilistic Backpropagation

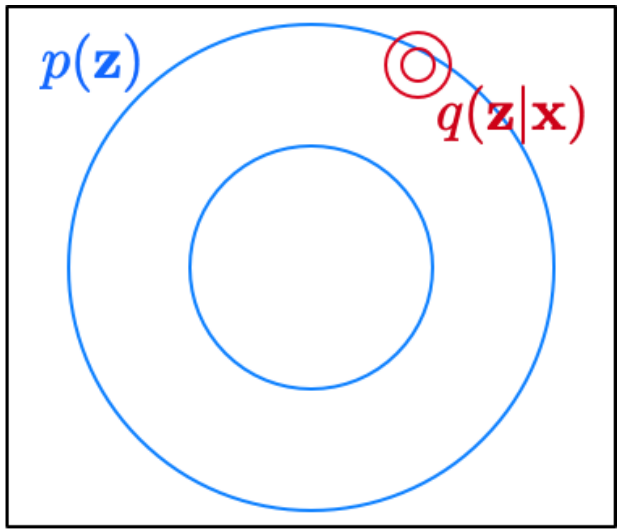
- So what are variational BNNs good for?

# Description Length Regularization

- What variational BNNs are really doing is regularizing the description length of the weights.
- Intuition: the more concentrated the posterior is, the more bits it requires to describe the location of the weights to adequate precision.
- A more concentrated $q$ generally implies a higher KL from the prior.




small KL divergence,
small description length




large KL divergence,
large description length

# Description Length Regularization

- The KL term $\mathrm{D}_{\mathrm{KL}}(q(\mathbf{w}) \,\|\, p(\mathbf{w}))$ can be interpreted as the number of bits required to describe $\mathbf{w}$ to adequate precision.
  - This can be made precise using the bits-back argument. This is beyond the scope of the class, but see here for a great explanation:

    `https://youtu.be/0IoLKnAg6-s`
- A classic result from computational learning theory ("Occam's Razor") bounded the generalization error a learning algorithm that selected from $K$ possible hypotheses.
  - It requires $\log K$ bits to specify the hypothesis.
  - PAC-Bayes gives analogous bounds for the generalization error of variational BNNs, where $\mathrm{D}_{\mathrm{KL}}(q(\mathbf{w}) \,\|\, p(\mathbf{w}))$ behaves analogously to $\log K$.
    - This is one of the few ways we have to prove that neural nets generalize.
    - See Dziugaite et al., "Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data".

# Uses of BNNs

- **Guiding exploration**
  - **Bayesian optimization:** Snoek et al., 2015, "Scalable Bayesian optimization using deep neural networks"
  - **Curriculum learning:** Graves et al., 2017, "Automated curriculum learning for neural networks"
  - **Intrinsic motivation in reinforcement learning:** Houthooft et al., 2016, "Variational information maximizing exploration"
- **Network compression:** Louizos et al., 2017, "Bayesian compression for deep learning"
- **Predicting generalization:** Jiang et al., 2019, "Fantastic generalization measures and where to find them": measures based on PAC-Bayesian/MDL ideas were the most predictive of generalization
- Lots more references in CSC2541, "Scalable and Flexible Models of Uncertainty"
  - `https://csc2541-f17.github.io/`

# Variational Bayes and Curvature

# Variational Bayes and Curvature

- We introduced the Laplace approximation, where uncertainty comes directly from the curvature:

$$p(\mathbf{w} \mid \mathcal{D}) \approx \mathcal{N}(\mathbf{w}; \mathbf{w}_{\text{MAP}}, \mathbf{H}^{-1})$$
$$\mathbf{H} = -\nabla^2 \log p(\mathbf{w} \mid \mathcal{D})$$

- But the Laplace approximation is often a very poor model of uncertainty. Variational Bayes is often better (at least for capturing a single mode).

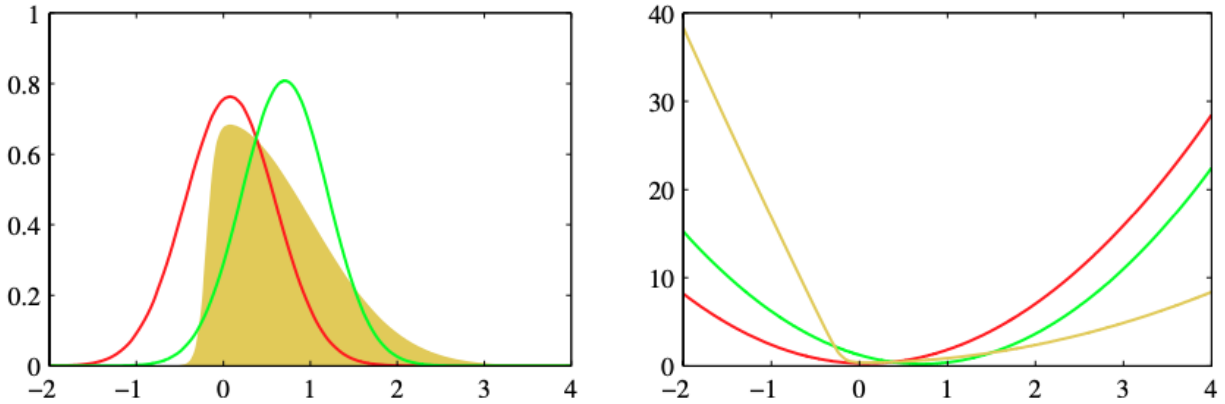



**Figure 10.1** Illustration of the variational approximation for the example considered earlier in Figure 4.14. The left-hand plot shows the original distribution (yellow) along with the Laplace (red) and variational (green) approximations, and the right-hand plot shows the negative logarithms of the corresponding curves.

(Bishop, PRML)

# Noisy Natural Gradient

Zhang et al., 2018, "Noisy natural gradient as variational inference"

- Consider a proximal objective for variational Bayes, over a family of probability distributions (say, multivariate Gaussians):

$$\min_{\boldsymbol{\phi}} -\mathcal{F}(q_{\boldsymbol{\phi}}) + \mathrm{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}} \| q_{\boldsymbol{\phi}_0})$$

- As in Lecture 3, taking the infinitesimal limit gives the natural gradient update, called Natural Gradient for Variational Inference (NGVI):

$$\boldsymbol{\phi}^{(k+1)} = \boldsymbol{\phi}^{(k)} + \alpha \mathbf{F}_{\boldsymbol{\phi}}^{-1} \nabla \mathcal{F}(q_{\boldsymbol{\phi}^{(k)}})$$

- **Note:** $\boldsymbol{\phi}$ are the variational parameters, not the network weights. $\mathbf{F}_{\boldsymbol{\phi}}$ is the Fisher information matrix for the multivariate Gaussian distribution, not the pullback metric for the network.

# Noisy Natural Gradient

(Zhang et al., 2018)

- Parameterize the multivariate Gaussian by mean $\boldsymbol{\mu}$ and precision matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$
- Stochastic NGVI update rule (derivable using the exponential family identities from NNTD Chapter 3):
  - Sample the network weights $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$
  - Update the variational parameters

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \alpha \boldsymbol{\Lambda}^{-1} \left[ \nabla \log p(y \,|\, \mathbf{x}, \mathbf{w}) - \frac{\lambda}{N\eta} \mathbf{w} \right]$$

$$\boldsymbol{\Lambda} \leftarrow \left(1 - \frac{\lambda\beta}{N}\right) \boldsymbol{\Lambda} + \beta \left[ -\nabla^2_{\mathbf{w}} \log p(y \,|\, \mathbf{x}, \mathbf{w}) + \frac{\lambda}{N\eta} \mathbf{I} \right]$$

- Update for $\boldsymbol{\Lambda}$: exponential moving average of the Hessian (justifies the use of moving averages in second-order optimization!)
- Update for $\boldsymbol{\mu}$: stochastic Newton update to the weights

# Noisy Natural Gradient

(Zhang et al., 2018)

- All roads lead to $\mathbf{H}$!
- In practice, we approximate $\mathbf{H}$ with $\mathbf{G}$ so that it's PSD
- Imposing structure on $\mathbf{\Lambda}$ corresponds to imposing structure on the variational posterior
  - Diagonal $\mathbf{\Lambda} \Leftrightarrow$ factorial (independent) Gaussian posterior (noisy Adam)
  - K-FAC approximation for $\mathbf{\Lambda} \Leftrightarrow$ matrix variate Gaussian posterior (noisy K-FAC)
    - Unlike the diagonal approximation, this can capture correlations between different weights. So it's not just about optimization!

# Variational Bayes and Flatness

- Recall: sharp/flat minima claims
  - Gradient noise implicitly regularizes towards flat minima
  - Flat minima generalize better
  - Obvious sharpness measures based on $\mathbf{H}$ aren't invariant to reparameterization
- Variational Bayes
  - Adds noise to the weights
  - KL term encourages flatness (high posterior volume)
  - Strong generalization bounds from PAC-Bayes
  - KL term is invariant to reparameterization
- Is this the right way to think about flatness? I don't know
- Is ordinary SGD training doing anything Bayesian?

# SGD and Implicit Variatonal Bayes

# SGD and Implicit Variational Bayes

Welling and Teh, 2011, "Bayesian learning via stochastic gradient Langevin dynamics"

- Hamiltonian Monte Carlo (HMC), as pioneered by Neal (1993), is still the gold-standard inference method for BNNs. But it doesn't scale to large datasets because it requires full batch computations.
- Stochastic gradient Langevin dynamics (SGLD) is a scalable alternative which uses stochastic gradients.
- Update rule: compute the mini-batch gradient and add noise

$$\mathbf{w}^{(k)} = \mathbf{w}^{(k-1)} - \alpha \mathbf{g}^{(k)} + \eta^{(k)}$$
$$\eta^{(k)} \sim \mathcal{N}(\mathbf{0}, \alpha \mathbf{I}),$$

  where $\mathbf{g}$ is a stochastic estimate of the log-likelihood gradient
- As $\alpha \to 0$, $\eta^{(k)}$ dominates the mini-batch noise, and the stationary distribution approaches the true posterior

# SGD and Implicit Variational Bayes

- In practice, $\alpha$ has to be very small for the gradient noise to be negligible
- Is ordinary SGD doing something like SGLD?
  - Only if the gradient noise has spherical covariance (unlikely!)

# SGD and Implicit Variational Bayes

Mandt, Hoffman, and Blei, 2017, "Stochastic gradient descent as approximate Bayesian inference"

- Analyzes SGD as an Ornstein-Uhlenbeck process, essentially a continuous analogue of the NQM
- Quadratic cost function $\mathcal{J}(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\top \mathbf{H}\mathbf{w}$, gradient $g(t) = \mathbf{H}\mathbf{w}(t)$, gradient covariance $\mathbf{C} = \mathbf{A}\mathbf{A}^\top$
- Bayesian posterior: $\mathcal{N}(0, \frac{1}{N}\mathbf{H}^{-1})$
- Stochastic differential equation for the dynamics:

$$\mathrm{d}\mathbf{w}(t) = -\alpha g(t)\,\mathrm{d}t + \frac{\alpha}{\sqrt{B}}\mathbf{A}\,\mathrm{d}W(t)$$

  where W is a white noise process
- Stationary distribution is $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$, where $\mathbf{\Sigma}$ satisfies

$$\mathbf{\Sigma}\mathbf{H} + \mathbf{H}\mathbf{\Sigma} = \frac{\alpha}{B}\mathbf{C}$$

# SGD and Implicit Variational Bayes

$$\mathbf{\Sigma H} + \mathbf{H\Sigma} = \frac{\alpha}{B}\mathbf{C}$$

(Mandt et al., 2017)

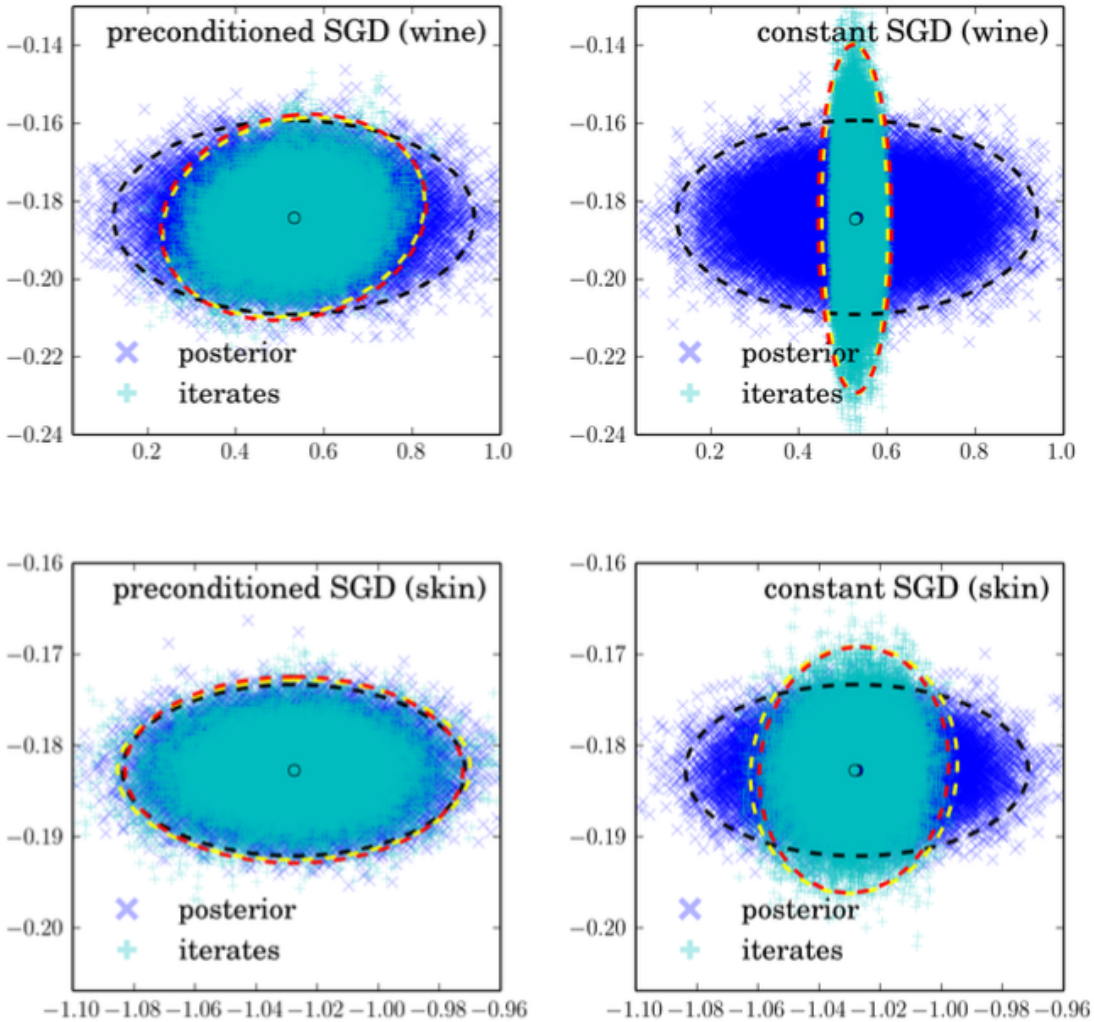
- If $\mathbf{H}$ and $\mathbf{C}$ are codiagonalizable, then is simplifies to:

$$\mathbf{\Sigma} = \frac{\alpha}{2B}\mathbf{H}^{-1}\mathbf{C}$$

- If $\mathbf{H} = \mathbf{C}$ (as in Lecture 7), then we get $\mathbf{\Sigma} = \frac{\alpha}{2B}\mathbf{I}$. This is not Bayesian! It ignores posterior uncertainty!
- The optimal preconditioner from a Bayesian standpoint is $\mathbf{C}^{-1}$. Preconditioned SGD converges to $\mathbf{\Sigma} = \frac{\alpha}{2B}\mathbf{H}^{-1}$.
  - Justifies the use of adaptive gradient methods, sort of (they precondition by $\mathbf{F}_{\text{emp}}^{-1/2}$, and arguably $\mathbf{F}_{\text{emp}} \approx \mathbf{C}$)
  - If $\mathbf{H} = \mathbf{C} = \mathbf{F}$, then natural gradient descent is doing Bayesian inference. (See also Ahn et al., 2012, "Bayesian posterior sampling via stochastic gradient Fisher scoring")

# SGD and Implicit Variational Bayes

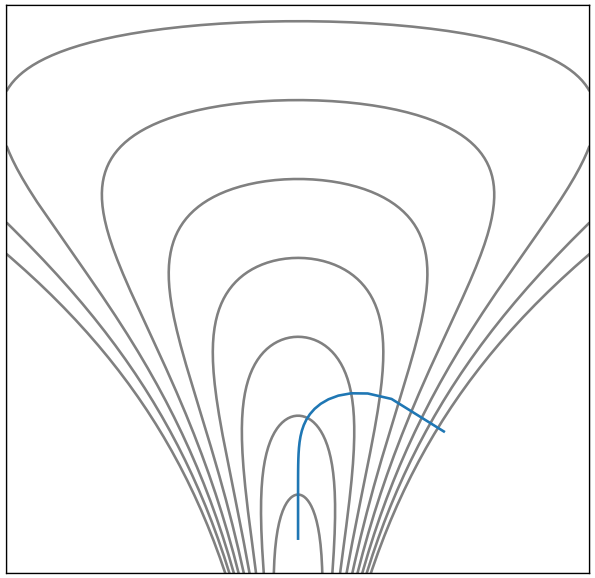
(Mandt et al., 2017)
Empirical results (note: axis scale not uniform)



# Gradient Noise and Flatness

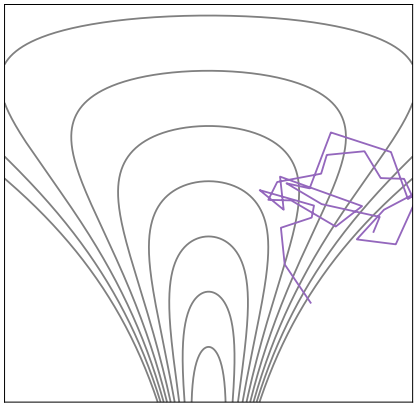
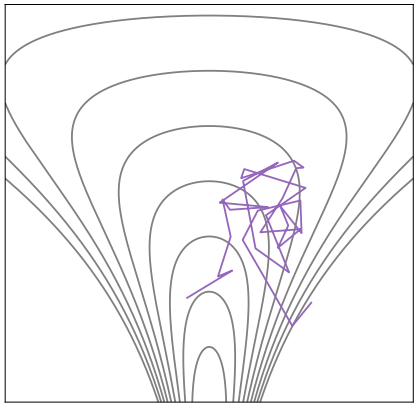
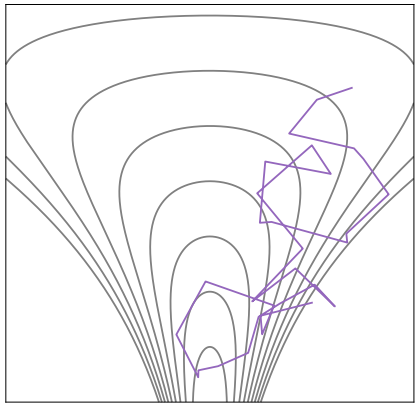
# Gradient Noise and Flatness

- The preceding analysis tells us that if you run SGD for a long time, the spread in the iterates tells us something about flatness or uncertainty.
- But SGD also encourages flatness in a way that isn't captured by that analysis.
- Considering the following funnel-shaped cost function. If we run gradient descent, it moves directly downhill.

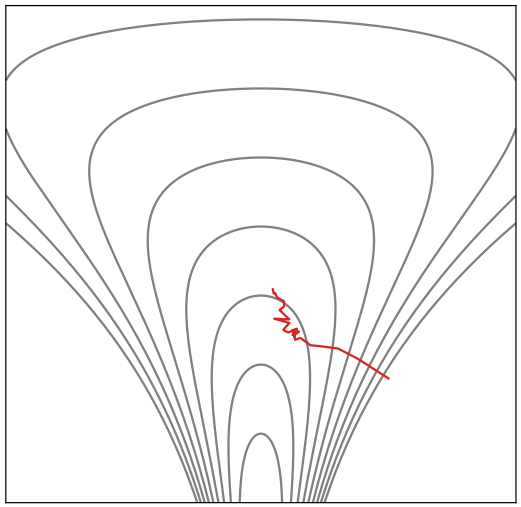


# Gradient Noise and Flatness

- Suppose we use stochastic gradients, where Gaussian noise with variance $\sigma^2$ is added to the gradient. Here are three trajectories:

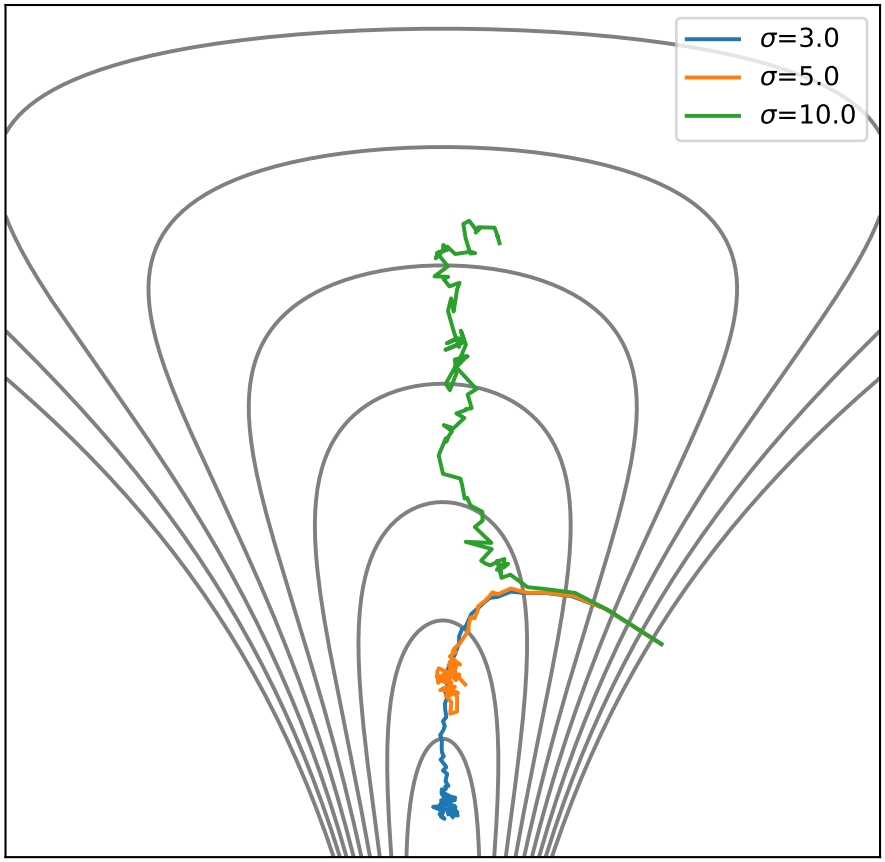




- Averaging over 100 trajectories, we can see a drift towards the flatter region.



# Gradient Noise and Flatness

- Larger amounts of noise create a stronger bias towards flatness.




- Can we characterize the implicit regularization effect of gradient noise?

# Gradient Noise and Flatness

Smith et al., ICLR 2021. "On the origin of implicit regularization in stochastic gradient descent"

- Goal: find an explicit regularizer which can mimic the implicit regularization effect of SGD in full batch mode.
- In the limit of small learning rates and more steps, the SGD trajectories approach a gradient flow:

$$\dot{\mathbf{w}}(t) = -\alpha \nabla \mathcal{J}(\mathbf{w}(t))$$

- Want to find a regularizer $\mathcal{R}(\mathbf{w})$ such that the average SGD trajectory for a typical learning rate is well approximated by:

$$\dot{\mathbf{w}}(t) = -\alpha \nabla [\mathcal{J}(\mathbf{w}(t)) + \mathcal{R}(\mathbf{w}(t))]$$

- This is a strategy known as backwards analysis.

# Gradient Noise and Flatness

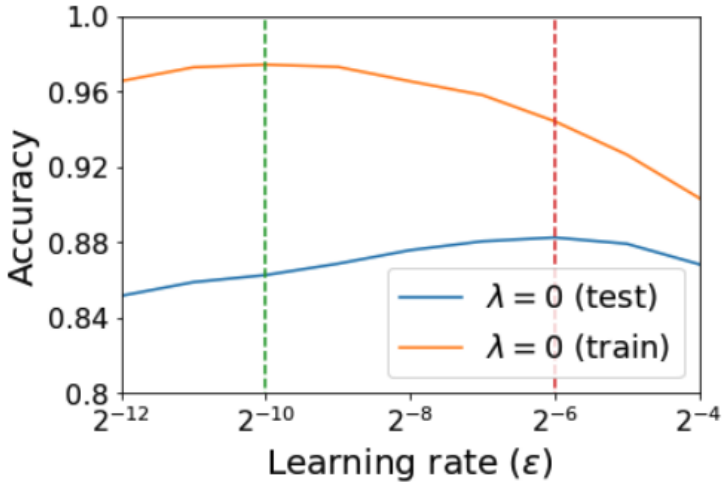
- For SGD with learning rate $\alpha$, batch size $B$, and dataset size $N$, they show that the regularizer is given by:

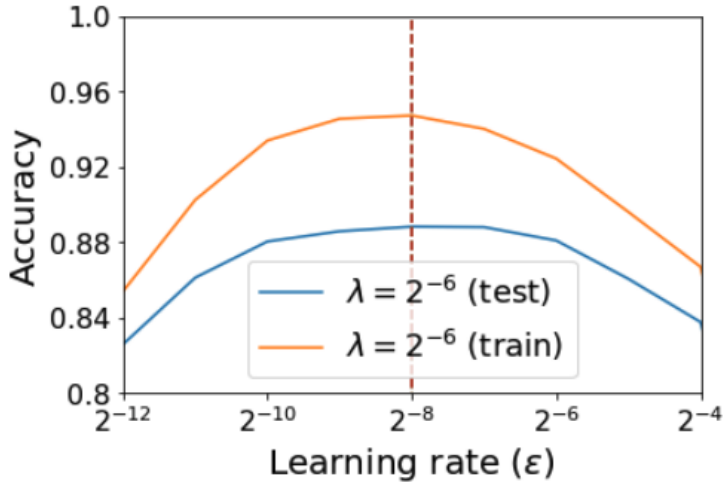
$$\mathcal{R}(\mathbf{w}) = \frac{\alpha}{4}\|\nabla \mathcal{J}(\mathbf{w})\|^2 + \frac{N-B}{N-1}\frac{\alpha}{4B}\operatorname{tr}\mathbf{C}_{\mathbf{w}},$$

  where $\mathbf{C}_{\mathbf{w}}$ is the covariance of the per-example gradients, evaluated at $\mathbf{w}$.
- To the extent that $\mathbf{C}_{\mathbf{w}}$ (similar to the empirical Fisher $\mathbf{F}_{\text{emp}}$) is related to the Hessian $\mathbf{H}$, this can be seen as a flatness regularizer.
- Dependence on $\alpha$ and $B$
  - The second term is stronger for larger $\alpha$ or $B$, as we'd expect.
  - The first term is independent of $B$, so even full-batch GD contributes some sort of implicit bias. I don't know what is the practical significance of this term.

# Gradient Noise and Flatness

- For a Wide-ResNet on CIFAR10, with the original loss **(left)**, the learning rate which maximizes test accuracy is larger than the one that maximizes training accuracy, indicating an implicit regularization effect.
- This goes away if the regularizer is added explicitly **(right)**.



# Gradient Noise and Flatness

- Under the original objective **(left)**, there is a generalization advantage to small batches. This goes away when the explicit regularizer is added **(right)**.